Rapport brief

“**Step By Step :**

**Skills Tree**”

Realisé par:

Mehdi snaiki  
Jamal Eddine Nomane

Khalid askhatir

Encadré par:

Hanae Elwahabi

[**Contexte du projet**](#_xdc41qdh4tin) **3**

[**Etude et conception**](#_3qm6folvgk7j) **4**

[**Realisation:**](#_m1rpu8gt3fkb) **6**

[Base de données](#_tg2oy72fz9ad) 6

[La table user :](#_7ib60glcrw59) 7

[La table competence:](#_egm7b2fmtie5) 7

[**Les triguers et les procedure:**](#_4lhie2o6iujt) **8**

[Trigger ajouter\_competance user](#_an6c4vsdhupq) 8

[PROCEDURE change\_user\_reference](#_qapbztqmmomx) 9

[**Les classes ultils:**](#_wc8ravjba9c2) **10**

[La class User:](#_jsasa5xjfoxz) 10

[La class staff:](#_ebmxzskabii6) 11

[La class apprenant:](#_ywpcp7dey56w) 11

[La class competence](#_8u3zow6olfsn) 12

[La class db skills](#_75yh2hyqkp0j) 13

[**Les interfaces:**](#_qta50ulxe0a1) **17**

[**Methodes utiles**](#_omqvhlsxoco1) **17**

# Contexte du projet

Dans le but d’informatiser les outils pédagogiques du suivi de l’état d’avancement de chaque apprenant, on propose l’utilisation des arbres de compétences sous format numérique. L’idée est de développer une application desktop qui permet la gestion de l’arbre de compétences. Cette dernière offre la possibilité à chaque apprenant de s’autoévaluer et de visualiser son état d’avancement.

Description de l’application :

L’utilisateur de l’application se connecte tant que staff ou apprenant

L’apprenant pourra valider un ou plusieurs niveaux d’une ou plusieurs compétences

Les membres de staff auront la possibilité de sélectionner l’apprenant et de visualiser son état

d’avancement sans pouvoir le modifier

La durée de formation est de deux ans.

Les apprenants de la première année devront valider un référentiel « à préciser !»

Une fois réussi, Les apprenants de la deuxième année devront valider un autre référentiel « à préciser !»

Chaque référentiel est réparti en un nombre fini de compétences et chaque compétence se compose de trois niveaux classés par ordre croissant de difficulté N1, N2 et N3.

La validation de chaque compétence nécessite la validation des trois niveaux

La validation de chaque référentiel nécessite la validation de toutes les compétences

La formation basée sur le principe de la pédagogie active. Chaque apprenant est censé de valider toutes les niveaux de toutes les compétences afin de réussir l’année

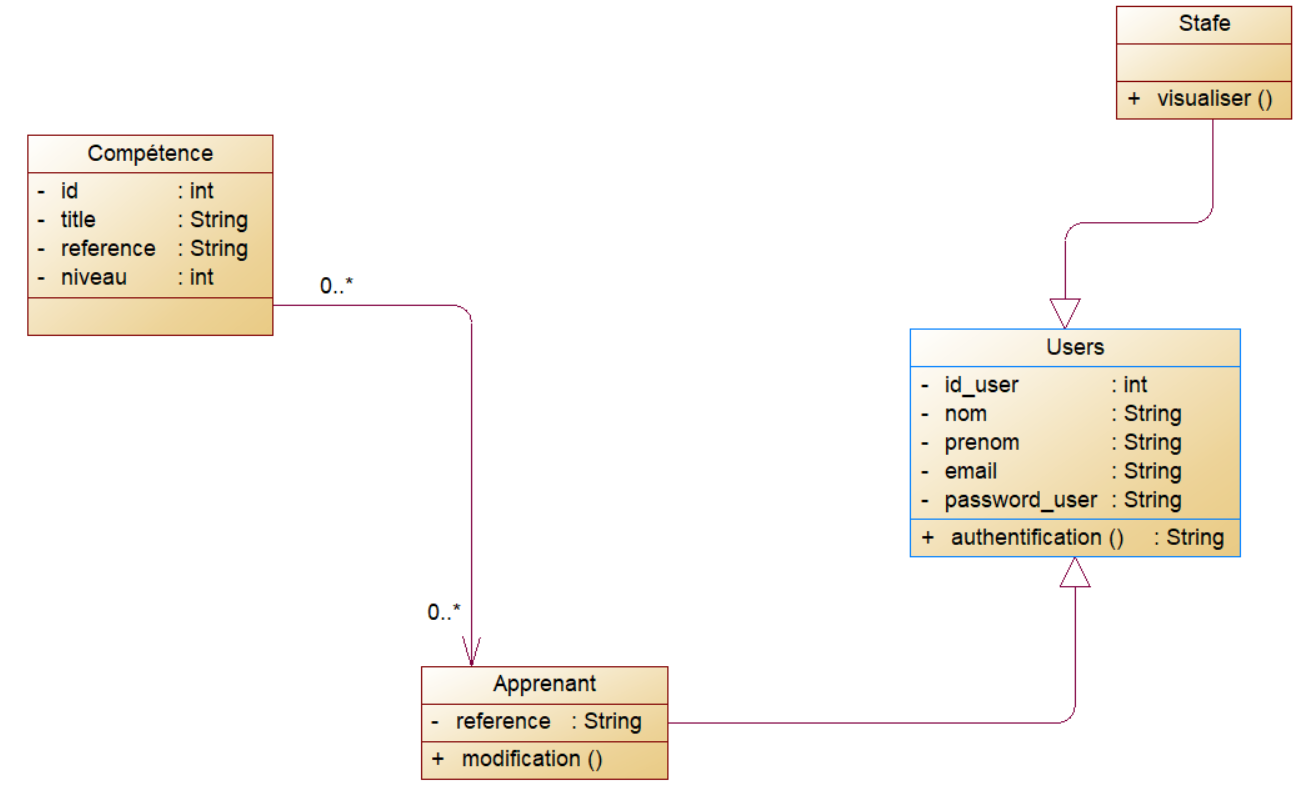
Tasks :

* Proposez un diagramme de classes, un diagramme de cas d’utilisations et un diagramme de séquences
* Créez la base de données
* Créez une application desktop qui implémente la BDD et assure les services exigés

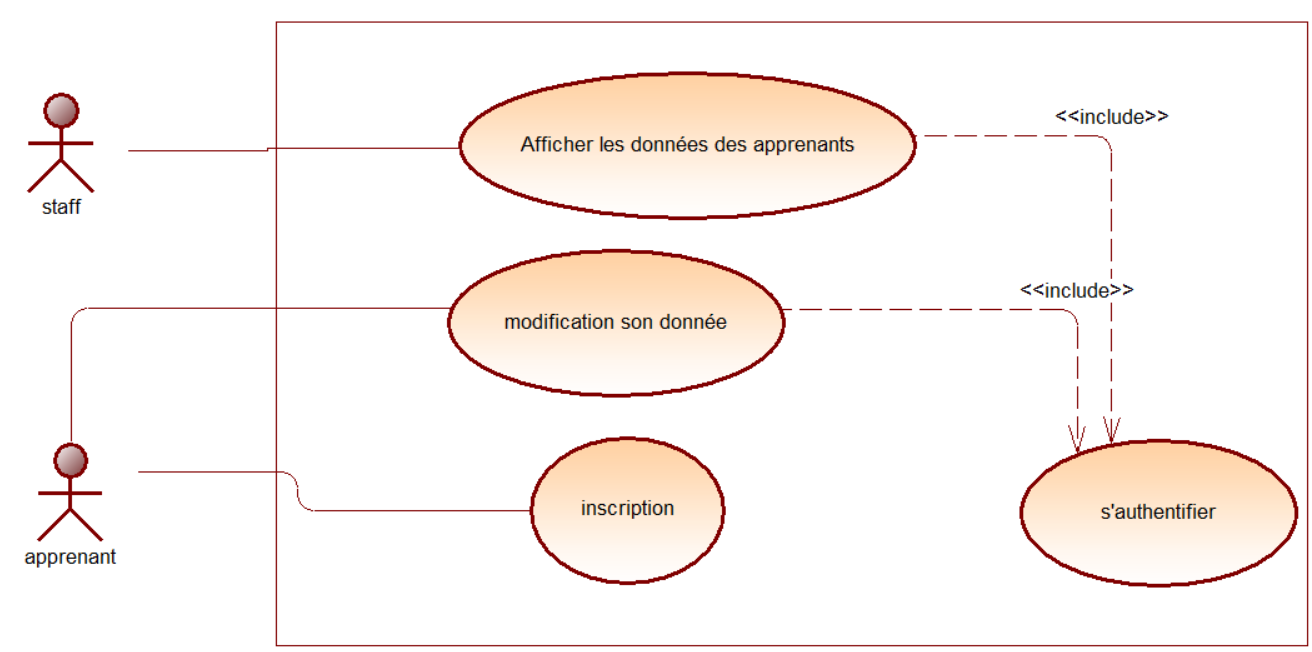
# Etude et conception

Dans le brief nous avon adopté la méthode UML pour avoir la clarté du projet en réalisant

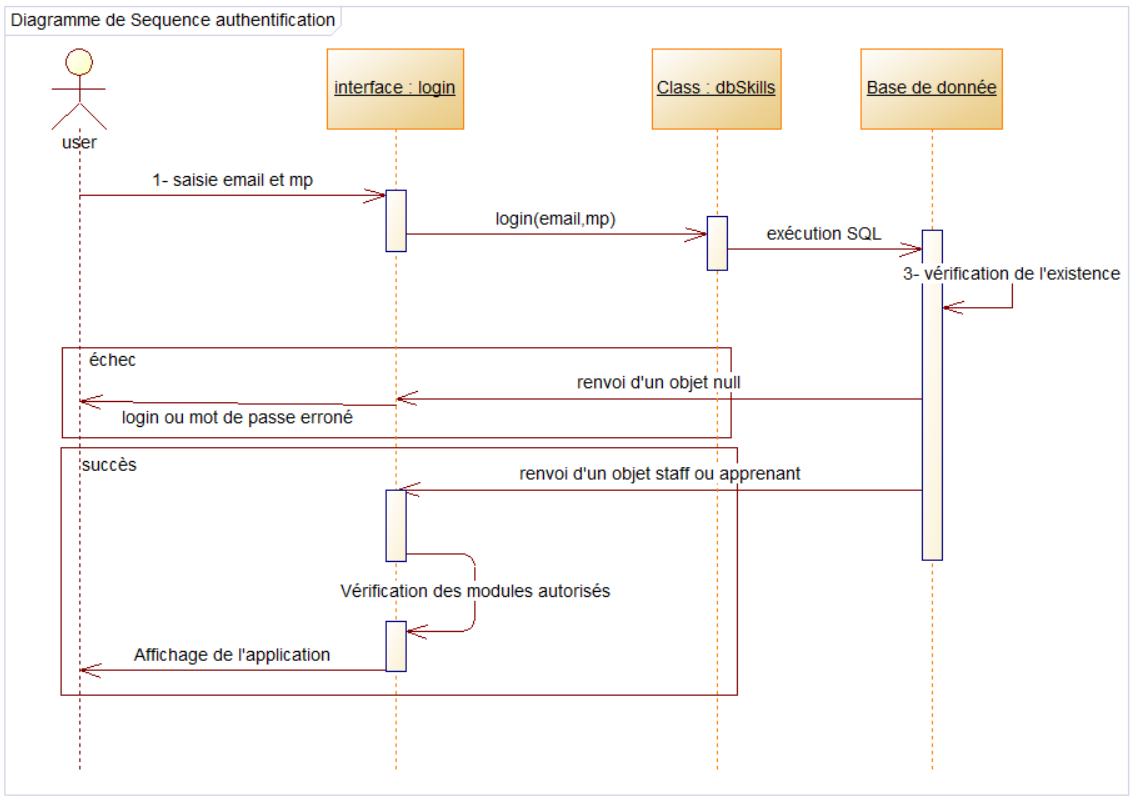
Des diagrammes de classe:

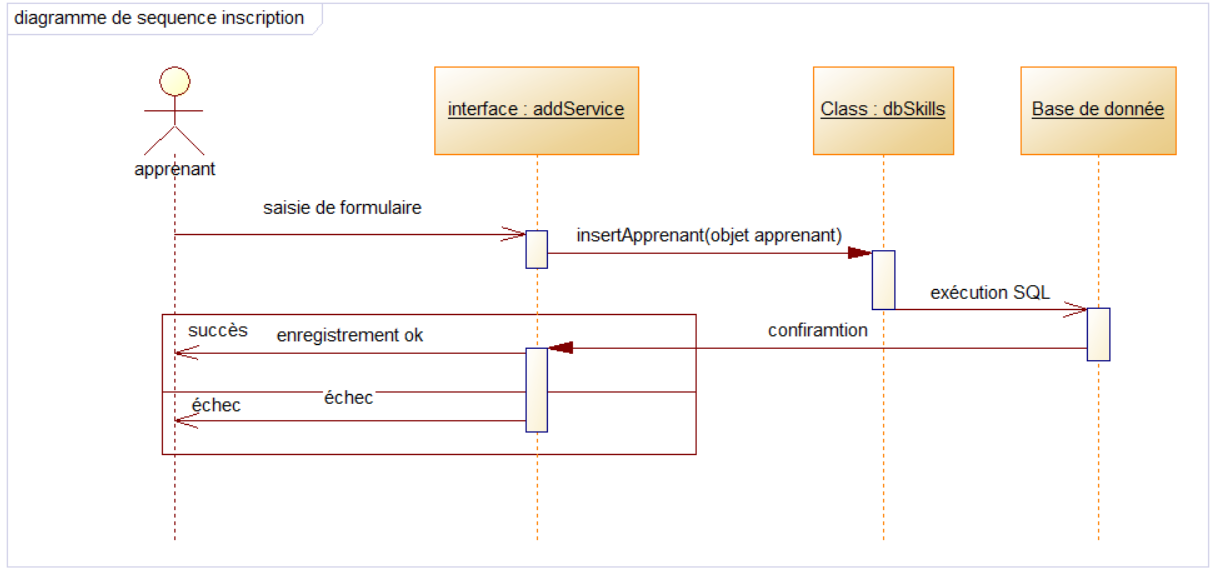


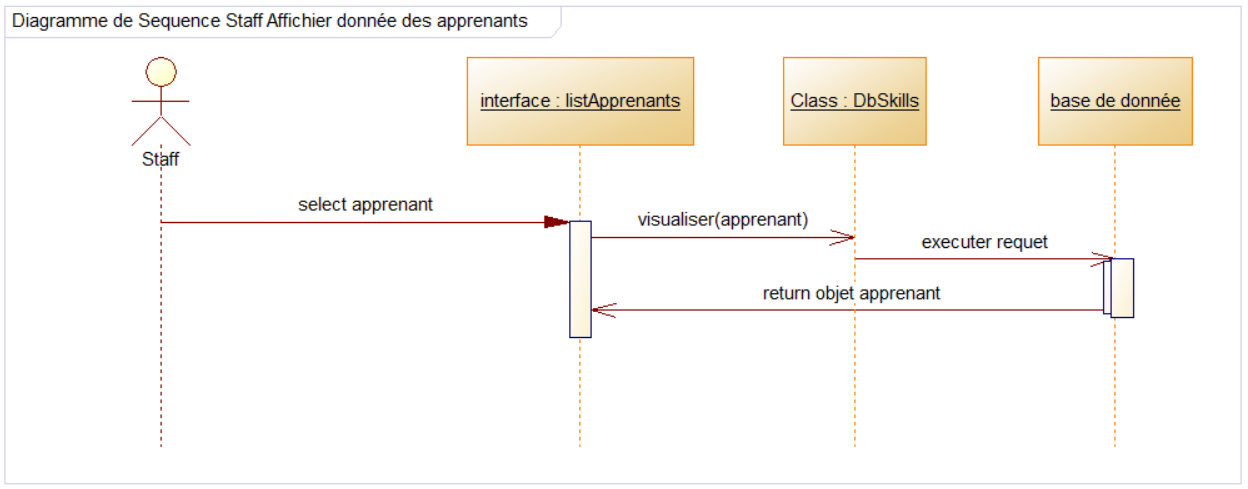
Des diagrammes des Cas d’utilisation:



Des diagrammes de séquences:

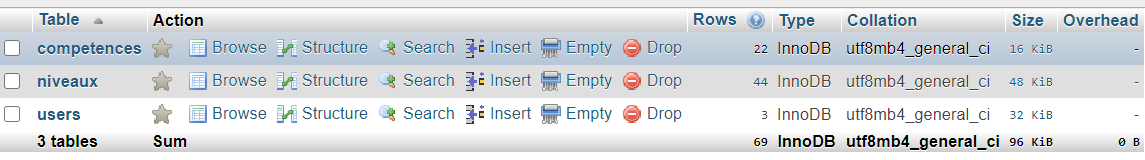






# Realisation:

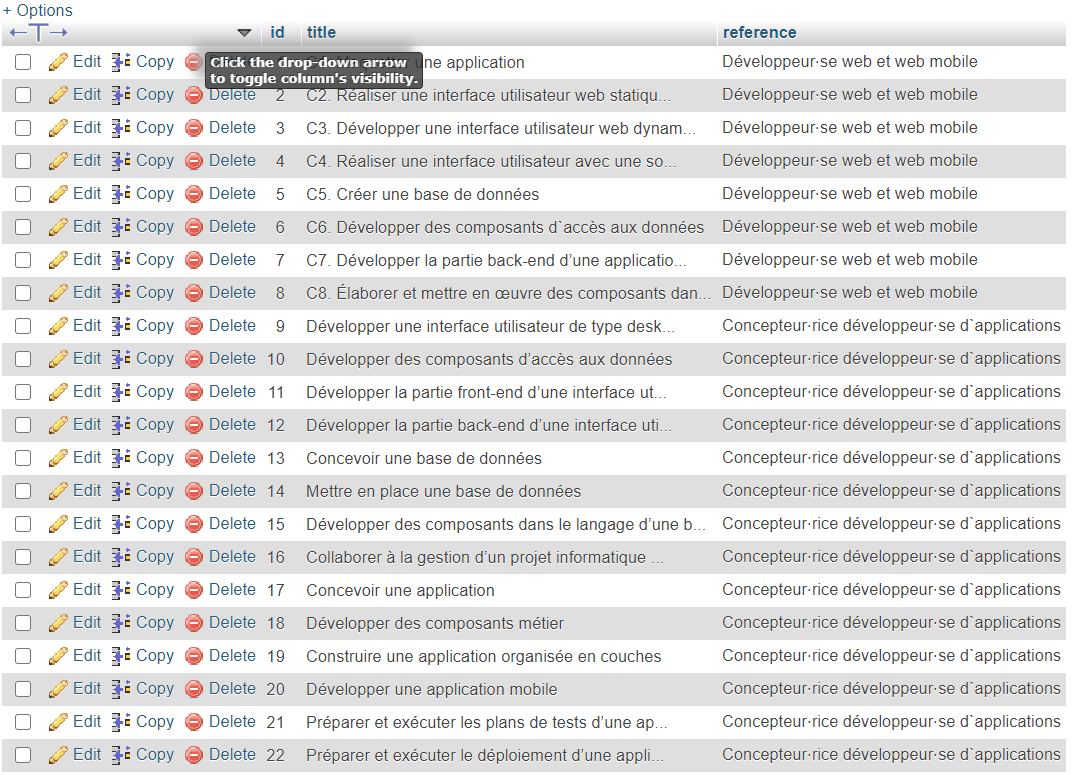
## Base de données

On a 3 tableau un pour les utilisateur un pour les compétence et un pour relier les deux en ajoutant et manipulant le niveau de chaqu’une:

### La table user :

### La table competence:

Une table qui contient les 22 compétences des 2 ans:



# Les triguers et les procedure:

## Trigger ajouter\_competance user

Après chaque création d’un user on ajoute lie son ID avec des compétences du tableau compétences avec des niveaux initialisés dépendant de son référence.

Create Trigger ajouter\_competence\_user

after insert

on users

For each ROW

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE ids INT;

DECLARE cur CURSOR FOR SELECT id FROM competences WHERE reference = 'Développeur⋅se web et web mobile';

DECLARE cur1 CURSOR FOR SELECT id FROM competences WHERE reference = new.reference;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

IF new.role = 'apprenant' THEN

IF new.reference='Concepteur⋅rice développeur⋅se d`applications' THEN

OPEN cur;

ins\_loop: LOOP

FETCH cur INTO ids;

IF done THEN

LEAVE ins\_loop;

END IF;

INSERT INTO niveaux(id\_user, id\_competence,niveau) VALUES (new.id,ids,3);

END LOOP;

CLOSE cur;

END IF;

SET done = FALSE;

OPEN cur1;

ins\_loop: LOOP

FETCH cur1 INTO ids;

IF done THEN

LEAVE ins\_loop;

END IF;

INSERT INTO niveaux(id\_user, id\_competence) VALUES (new.id,ids);

END LOOP;

CLOSE cur1;

END IF;

END; //

## PROCEDURE change\_user\_reference

C’est une procedure qui change la référence d’un user donnés:

CREATE PROCEDURE change\_user\_reference (id1 INT)

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE ids INT;

DECLARE cur CURSOR FOR SELECT id FROM competences WHERE reference = 'Concepteur⋅rice développeur⋅se d`applications';

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

update users set reference='Concepteur⋅rice développeur⋅se d`applications' where id=id1;

OPEN cur;

ins\_loop: LOOP

FETCH cur INTO ids;

IF done THEN

LEAVE ins\_loop;

END IF;

INSERT INTO niveaux(id\_user, id\_competence) VALUES (id1,ids);

END LOOP;

CLOSE cur;

END $$

# Les classes ultils:

## **La class User:**

Une classe abstract parent des type d’utilisateur d'où les classes apprenant et staff héritent les méthodes communes et les attribut communs.

package application;

public abstract class Users {

private int idUser;

private String nom;

private String prenom;

private String email;

private String password;

public Users(String nom, String prenom, String email, String password) {

super();

this.nom = nom;

this.prenom = prenom;

this.email = email;

this.password = password;

}

public Users(int idUser, String nom, String prenom) {

super();

this.idUser = idUser;

this.nom = nom;

this.prenom = prenom;

}

public int getIdUser() {

return idUser;

}

public void setIdUser(int idUser) {

this.idUser = idUser;

}

public String getNom() {

return nom;

}

public void setNom(String nom) {

this.nom = nom;

}

public String getPrenom() {

return prenom;

}

public void setPrenom(String prenom) {

this.prenom = prenom;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

}

## La class staff:

Contenant juste le constructeur

package application;

public class Staff extends Users {

public Staff(String nom, String prenom, String email, String password) {

super(nom, prenom, email, password);

}

}

## La class apprenant:

La class apprenant a un attribut de plus "référentiel" par rapport à la class staff et le role diffère, affectant bien évidemment les constructeurs.

package application;

public class Apprenant extends Users {

private String reference;

public String getReference() {

return reference;

}

public void setReference(String reference) {

this.reference = reference;

}

public Apprenant(String nom, String prenom, String email, String password, String reference) {

super(nom, prenom, email, password);

this.reference = reference;

}

public Apprenant(int idUser, String nom, String prenom, String reference) {

super(idUser,nom, prenom);

this.reference = reference;

}

}

## La class competence

Continent les attribut et les méthodes pour manipuler les compétences

package application;

public class Competence {

private int id;

private String title;

private String reference;

private int niveau;

public int getNiveau() {

return niveau;

}

public void setNiveau(int niveau) {

this.niveau = niveau;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getReference() {

return reference;

}

public void setReference(String reference) {

this.reference = reference;

}

public Competence(int id, String title, String reference, int niveau) {

super();

this.id = id;

this.title = title;

this.reference = reference;

this.niveau = niveau;

}

}

## La class db skills

Contient les attribut et les méthodes qui permette l’interaction avec la base de donnés, comme la connexion , l’inscription, prendre les listes des apprenant, modifier les compétences, prendre les données concernant les compétences d’un apprenant avec son id, et changement de référentiel.

package application;

import java.sql.CallableStatement;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

public class DbSkills implements Interface {

public static Connection getConnection(){

Connection con = null;

try {

Class.forName("com.mysql.jdbc.Driver");

con =DriverManager.getConnection("jdbc:mysql://localhost:3306/skills", "root","");

} catch (SQLException | ClassNotFoundException e) {

Main.getAlert("erreur de connection", "Connection");

}

return con;

}

public static Users login(String email, String password) {

Users user = null;

try {

String query ="SELECT \* FROM users WHERE email=? and password=?";

Connection con = DbSkills.getConnection();

PreparedStatement ps=con.prepareStatement(query);

ps.setString(1, email);

ps.setString(2, password);

ResultSet rs= ps.executeQuery();

if(rs.next()) {

if(rs.getString(6).equals("apprenant")) {

user = new Apprenant(rs.getString(2), rs.getString(3), email, password, rs.getString(7));

} else {

user = new Staff(rs.getString(2), rs.getString(3), email, password);

}

user.setIdUser(rs.getInt(1));

}

con.close();

} catch (SQLException e) {

e.printStackTrace();

}

return user;

}

public static int inscription(Apprenant user) {

int ret=0;

try {

String query = "INSERT into users (`nom`,`prenom`,`email`,`password`,`reference`) values(?,?,?,?,?)";

Connection con = DbSkills.getConnection();

PreparedStatement ps = con.prepareStatement(query);

ps.setString(1, user.getNom());

ps.setString(2, user.getPrenom());

ps.setString(3, user.getEmail());

ps.setString(4, user.getPassword());

ps.setString(5, user.getReference());

ret = ps.executeUpdate();

con.close();

} catch (SQLException e) {

e.printStackTrace();

}

return ret;

}

public static void getListApprenants() {

try {

listeApprenant.clear();

String query ="SELECT \* FROM `users` WHERE role='apprenant'";

Connection con = DbSkills.getConnection();

PreparedStatement ps=con.prepareStatement(query);

ResultSet rs= ps.executeQuery();

while(rs.next()) {

listeApprenant.add(new Apprenant(rs.getInt(1), rs.getString(2), rs.getString(3), rs.getString(7)));

}

con.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

public static void getCompetenceApprenant(int id) {

try {

listeCompetence.clear();

String query ="SELECT id\_competence,title,users.reference,niveau from niveaux,competences,users WHERE users.reference=competences.reference and users.id=niveaux.id\_user and niveaux.id\_competence=competences.id and users.id="+id;

Connection con = DbSkills.getConnection();

PreparedStatement ps=con.prepareStatement(query);

ResultSet rs= ps.executeQuery();

while(rs.next()) {

listeCompetence.add(new Competence(rs.getInt(1), rs.getString(2), rs.getString(3), rs.getInt(4)));

}

con.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

public static void updateCompetences(int id){

for (Competence competence : listeCompetence) {

try {

String query = "UPDATE `niveaux` SET `niveau`=? WHERE id\_competence=? and id\_user=?";

Connection con = DbSkills.getConnection();

PreparedStatement ps = con.prepareStatement(query);

ps.setInt(1, competence.getNiveau());

ps.setInt(2, competence.getId());

ps.setInt(3, id);

ps.executeUpdate();

con.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

public static void updateReference(int id){

try {

String query = "CALL change\_user\_reference(?)";

Connection con = DbSkills.getConnection();

CallableStatement stmt = con.prepareCall(query);

stmt.setInt(1, id);

stmt.execute();

stmt.close();

con.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

# Les interfaces:

On a deux arraylist un de type apprenant et l’autre de type compétence pour les appeler dans le package ou on a besoin,

package application;

import java.util.ArrayList;

public interface Interface {

public ArrayList<Apprenant> listeApprenant = new ArrayList<Apprenant>();

public ArrayList<Competence> listeCompetence = new ArrayList<Competence>();

}

# Methodes utiles

|  |  |
| --- | --- |
| Signature de la fonction | Description |
| public void start(Stage primaryStage) | Le rôle de cette procédure est la configuration de stage et scène et ajout des premier élément principaux. |
| public static void connexionInscription() | Invocation des deux méthode connexion () et inscription ainsi que les initialisations des positions de chacun des deux panel reliés à ces deux fonctions, en montrant un et cachant l’autre. |
| public static void connexion() | La population des élément du panel de la connexion ainsi ajoutant la condition sur label de l’inscription pour aller à l’inscription si voulus.elle utilise la méthode “login” de dbSkills pour vérifier les informations entrées. Si échec elle affiche une alerte d'échec sinon elle affiche une lettre de succès et exécute la fonction apprenant ou staff dépendant de type de retour de user. |
| isValidPassword(String password) | Prend une string et vérifie si elle respect la paterne donné, commence avec une string, il y a au moin un nombre, une string minuscule, une string majuscule, un caractère spécial, il y au moin 8 caractère et au plus 20 caractère et qu’il n'y a pas d’espace vide. |
| public static void inscription() | Cette méthode et presque la même que la connexion avec la différence qu' au lieu de vérifier les données à la base de données elle insère après qu'elle vérifie que les donné entrés sont valides.Après elle switch au panel de connexion. |
| public static void pageApprenant(Apprenant apprenant) | Après la creation de l’objet apprenant par la le méthode connexion et appel cette méthode en donnant le user créer comme argument, cette méthode crée le panel de l’apprenant et fait la population en utilisant sont ces donné pour retrouver ces compétences ainsi que leur niveaux,pour remplir l’interface listecompetances avec des compétences et des niveaux tout en posant la condition sur les label de niveau créé par le clic pour changer,  le niveau dépendant de l'état actuel en utilisant la méthode” changerCompetence”  Après création ou modification on clique sur modifier en fait une mise à jour à la base de donné des état actuel des Arraylistes des interfaces sinon si .on clic sur connexion on vide les interfaces et les array listes et on retourne à la panel exécuté par connexioninscription() |
| public static void changerCompetence(int i, int j) | En utilisant l’interface déjà remplie par la fonction précédente , en change le css et les niveau dans arraylist liste compétence |
| public static void pageStaff(Staff staff) | Cette procédure commence par afficher un combobox pour choisir les apprenants disponible dans la base de données, une fois choisi elle exécute le même processus de la méthode précédente avec la différence du manque des méthodes de modification . |
| public static void hideAndShow(Pane hide, Pane show, int time) | Cette Procédure fait l’animation de positionner deux panel en rotation de panel sur le plan x jusqu'à la total visibilité et l’autre jusqu'à la total invisibilité. |
| public static void getAlert(String text, String title) | Cette fonction prend des string et affiche des alertes en les utilisants |
| public static Connection getConnection() | Cette fonction prend en charge la connexion a la base de données |
| public static Users login | Cette Procédure prend l’email et le password et vérifie l'existence d’un user dans la base de donné si il existe elle crée et retourne un objet de type user |
| public static int inscription(Apprenant user) | Cette Procédure a pour rôle de faire l’insertion  A la base de donné les information d’un nouveau user |
| public static void getListApprenants() | Cette Procédure popularise l'interface listeApprenant |
| public static void getCompetenceApprenant(int id) | Cette Procédure popularise l'interface listeCompetence |
| public static void updateCompetences(int id) | Cette Procédure fait la mise a jours des niveaux des compétences |
| public static void updateReference(int id) | Cette méthode fait appel à la procédure dans la base de donné “change\_user\_réference”  Pour changer le référentiel. |