Analyse et conception orientée objet des SI

Plate-forme générique pour la gestion des activités des ingénieurs apprentis

Adrien AMESLANT

Corentin AVRIL

Maxime GONNORD

Tom HÉRAULT

Contents

Partie 1 : L'analyse et la conception du SI avec UML

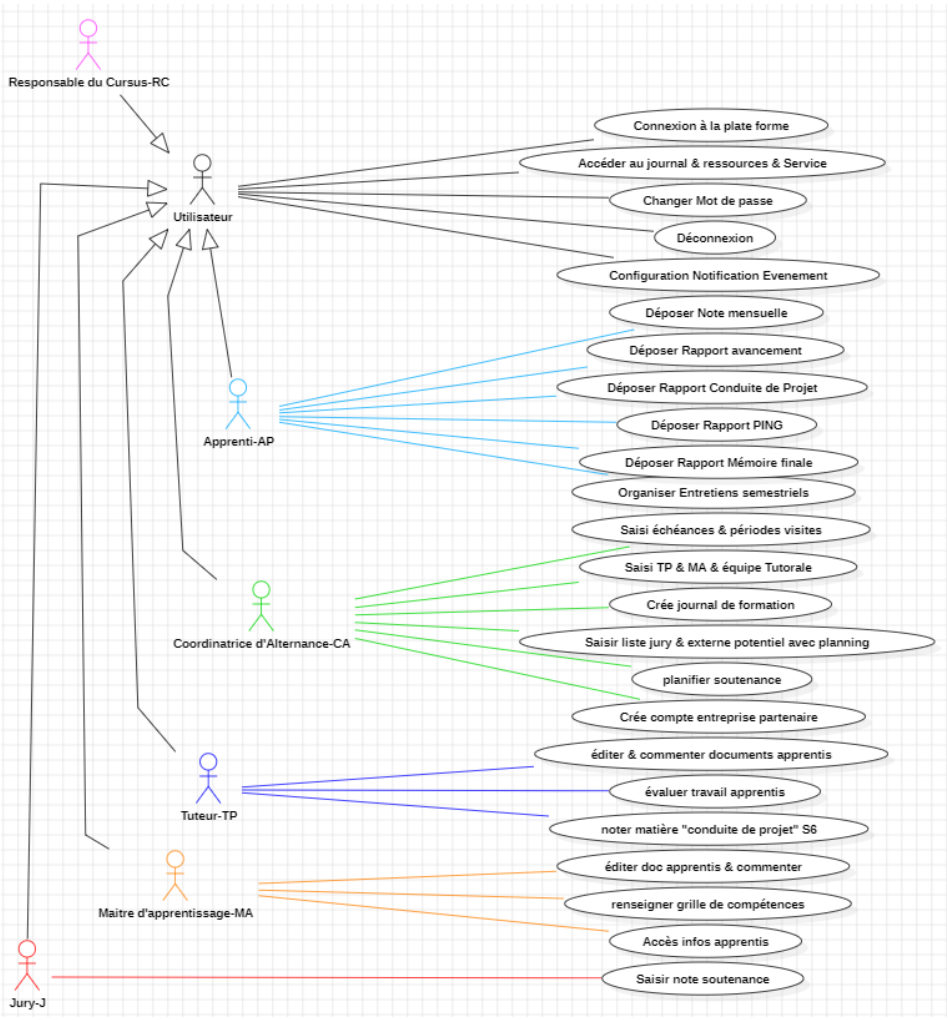
1. Acteurs et les principaux cas d'utilisation du système

Les acteurs du système sont tous des utilisateurs de la plateforme :

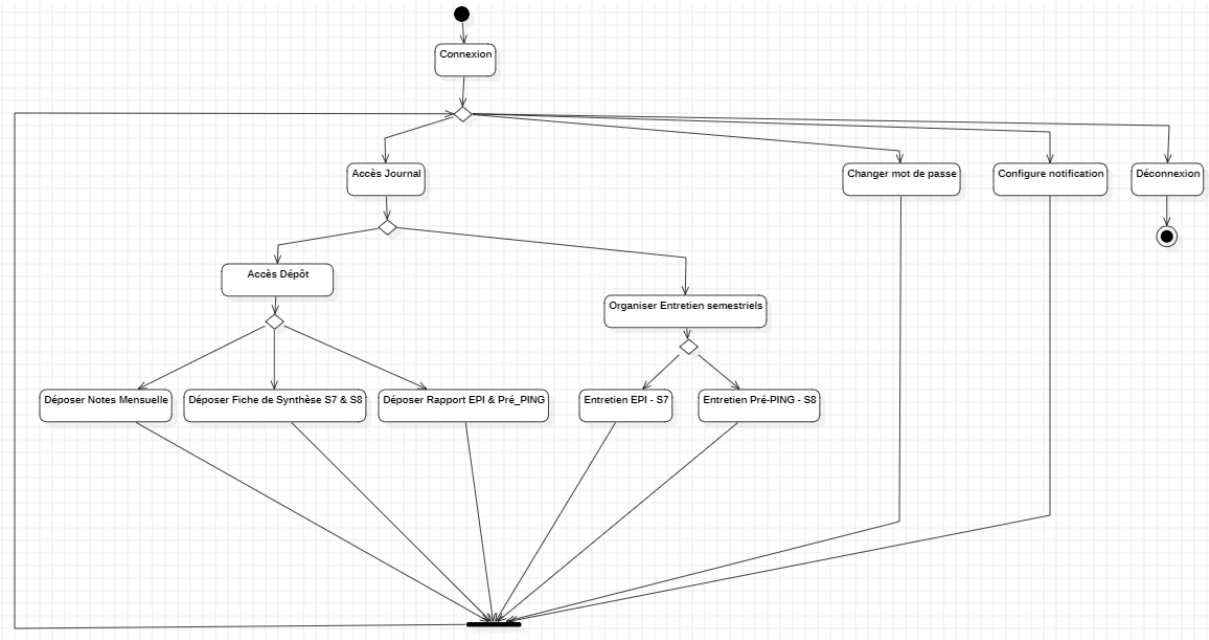
- Responsable du Cursus (RC)
- Apprenti (AP)
- Coordinatrice d'Alternance (CA)
- Tuteur pédagogique (TP)
- Maître d'apprentissage (MA)
- Jury (J)

Les principaux cas d'utilisation sont ceux d'utilisateur dans le diagramme de la question 2

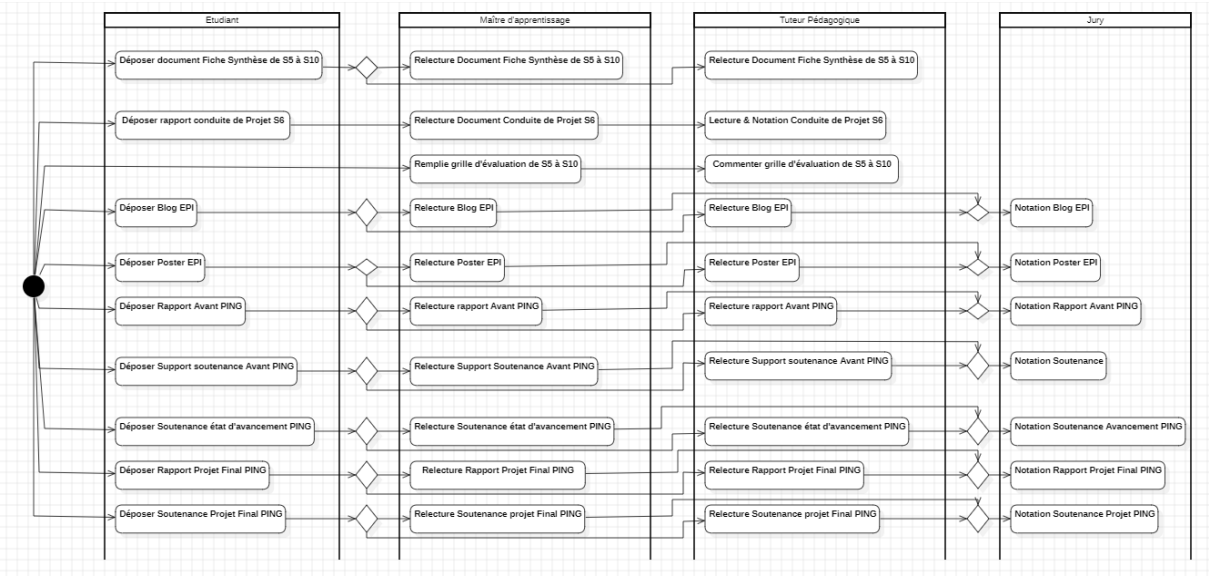
2. Diagramme des cas d'utilisations



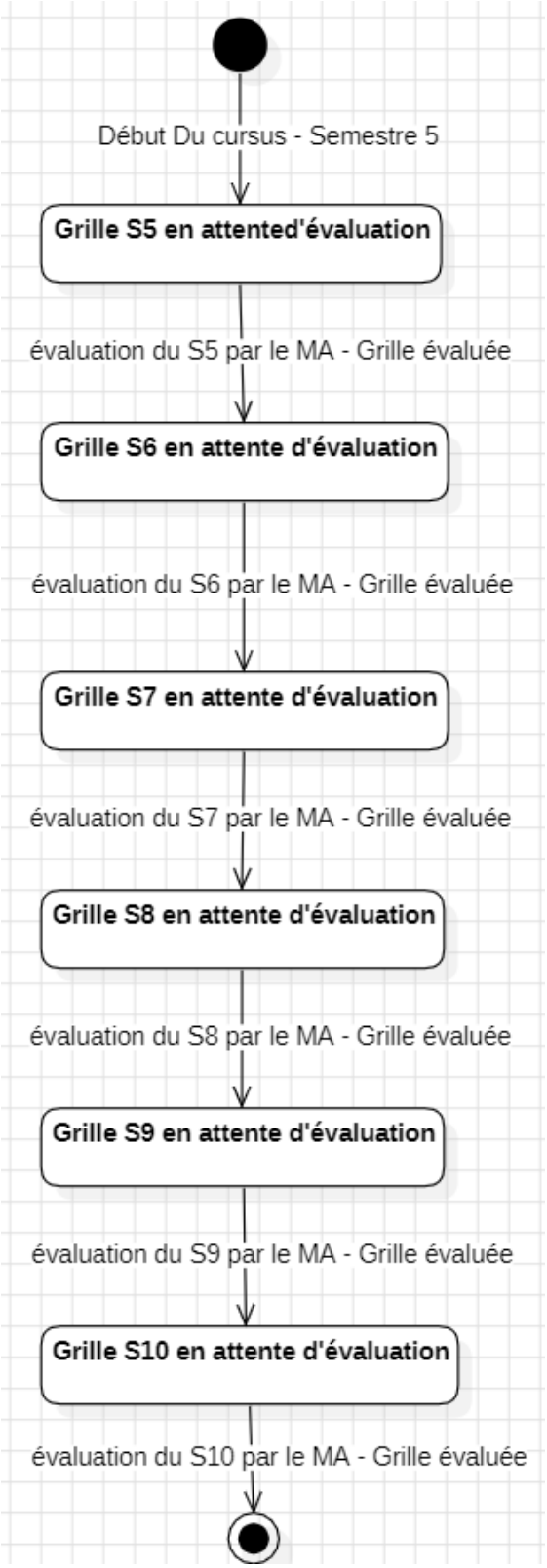
3. Diagramme d'activité simple



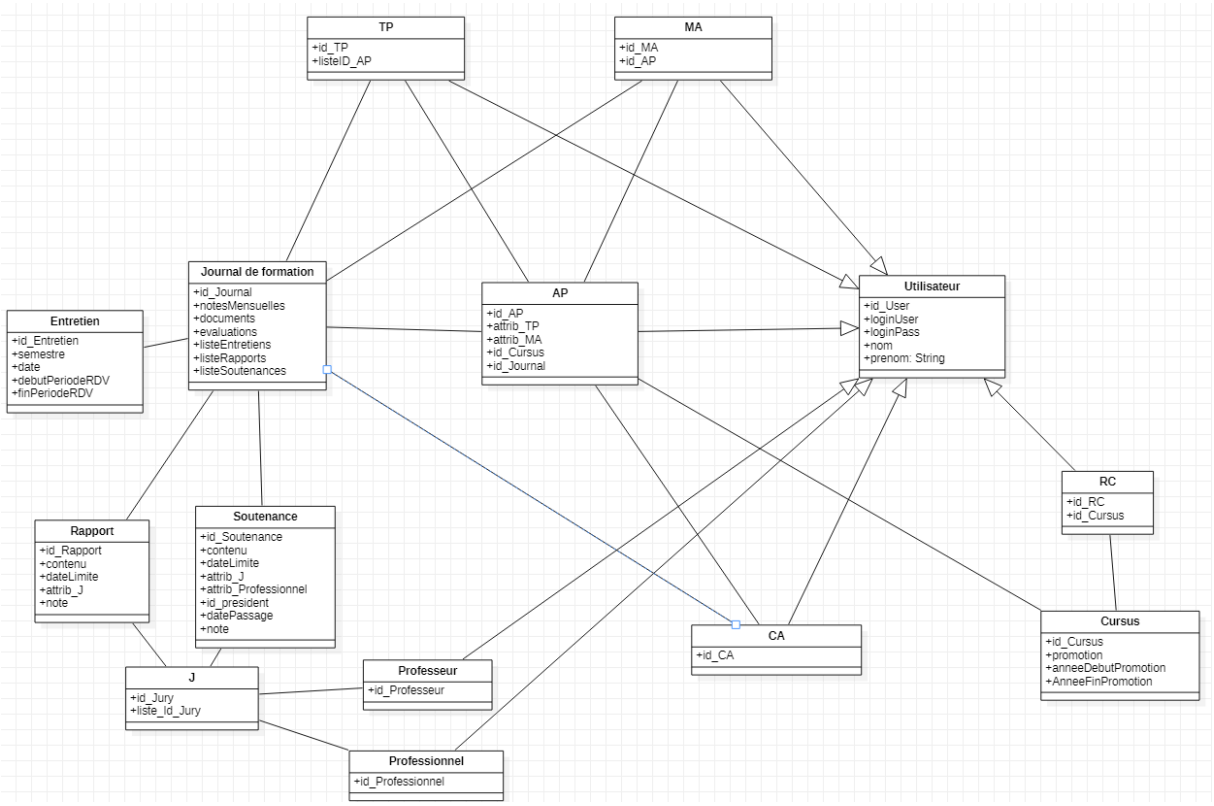
4. Diagramme d'activité avec couloirs « suivi et évaluation » (S5 à S10)



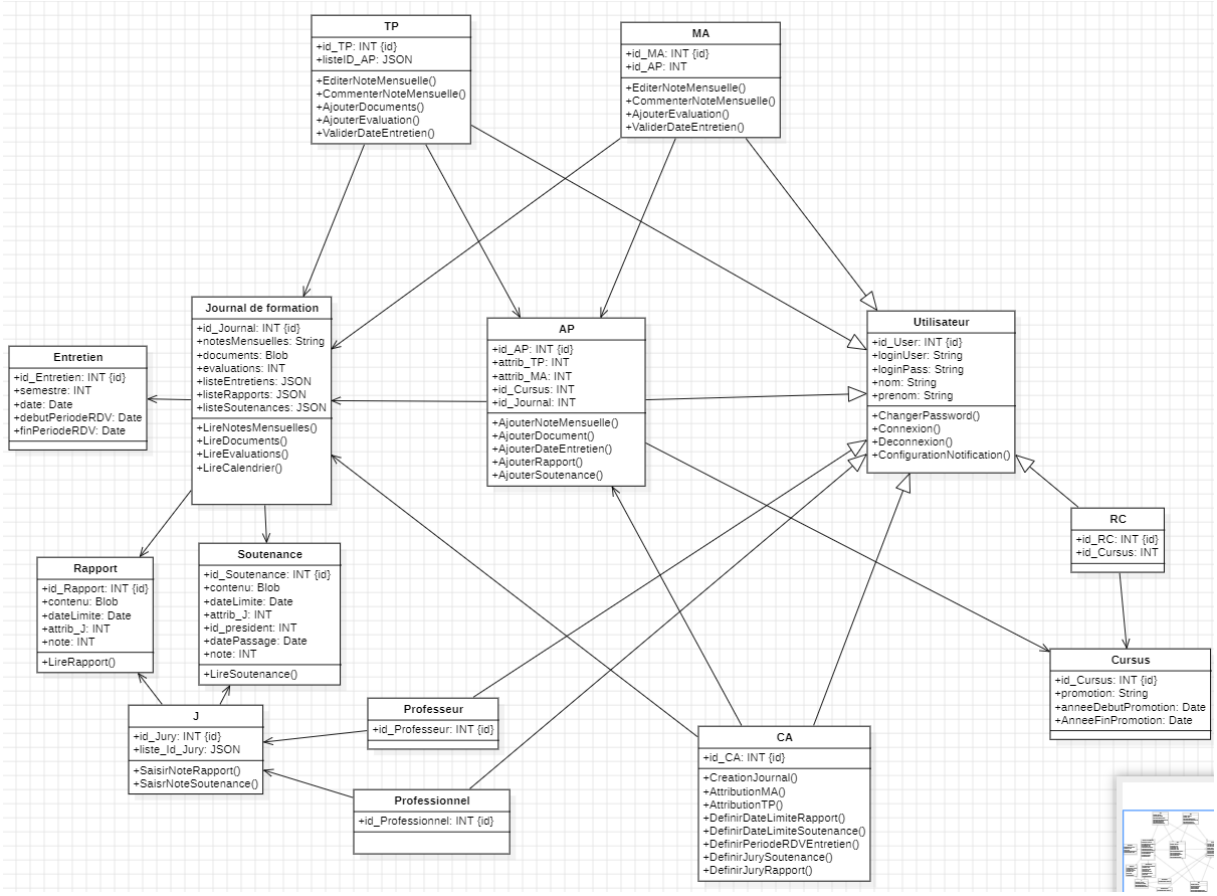
5. Diagramme d'états transitions



6. Diagramme des classes en phase d'analyse



7. Diagramme des classes en conception



Partie 2 : Une mise en œuvre orienté SI sous ORACLE :

1) Les schémas logique et physique de la BD

A retrouver dans l'archive zip : « Création_Insertion_BDD.sql ».

2) Les données insérées dans la DB

A retrouver dans l'archive zip : « Création_Insertion_BDD.sql ».

3) Les techniques proposées (Triggers, Fonction, Procédure,…)

Trigger :

- On ne peut pas décaler une date de soutenance inférieur à celle d'aujourd'hui.

```
CREATE TRIGGER check_soutenance_date
BEFORE UPDATE ON soutenance
FOR EACH ROW
BEGIN
    IF NEW.datePassage < CURDATE() THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cannot insert new passage date in the past.';
  END IF;
END;
```

- Ne peut pas déposer un rapport après la date limite.

```
CREATE TRIGGER check_limite_date
BEFORE UPDATE ON rapport
FOR EACH ROW
BEGIN
    IF rapport.dateLimite < CURDATE() THEN
        SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'Cannot insert new limite date in the past.';
    END IF;
END;
```

Fonction :

- En fonction de l'ID d'un jury, on renvoie toutes les soutenance et rapport qu'un jury a assisté.

```sql
DELIMITER //

CREATE FUNCTION calculate_average_grade(jury_id INT)
RETURNS FLOAT
BEGIN
    DECLARE average FLOAT;
    DECLARE total FLOAT;
    DECLARE count INT;

    -- Calculer la somme des notes et le nombre de notes pour un jury donné
    SELECT SUM(R.note), COUNT(R.note)
    INTO total, count
    FROM Rapport R
    JOIN Soutenance S ON R.attrib_J = S.id_Soutenance
    WHERE S.attrib_J = jury_id;

    -- Calculer la moyenne
    IF count = 0 THEN
        SET average = NULL;
    ELSE
        SET average = total / count;
    END IF;

    RETURN average;
END //

DELIMITER ;
```

- En fonction de l'ID d'un élève, on renvoie le Maître d'Apprentissage, le Tuteur Pédagogique et le Responsable de Cursus liés à l'étudiant.

```sql
CREATE OR REPLACE FUNCTION get_student_details(p_student_id IN NUMBER)
RETURN VARCHAR2
IS
    v_result VARCHAR2(4000);
BEGIN
    SELECT
        'EtudiantID: ' || etudiant.id_User ||
        ', Maître Apprentissage: ' || maitre_apprentissage.nom ||
        ', Tuteur Pédagogique: ' || tuteur_pedagogique.nom ||
        ', Responsable de Cursus: ' || responsable_cursus.nom
    INTO v_result
    FROM
        utilisateur AS etudiant
    INNER JOIN
        ap ON etudiant.id_User = ap.id_AP
    INNER JOIN
        utilisateur AS maitre_apprentissage ON ap.attrib_MA = maitre_apprentissage.id_User
    INNER JOIN
        utilisateur AS tuteur_pedagogique ON ap.attrib_TP = tuteur_pedagogique.id_User
    INNER JOIN
        rc ON ap.id_Cursus = rc.id_Cursus
    INNER JOIN
        utilisateur AS responsable_cursus ON rc.id_RC = responsable_cursus.id_User
    WHERE
        etudiant.id_User = p_student_id;

    RETURN v_result;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 'Pas d etudiant avec cet ID';
    WHEN OTHERS THEN
        RETURN 'Erreur : ' || SQLERRM;
END;
```

Procédure :

- Renvoie l'ID des soutenances ayant eu une note inférieure à 10.

```
DELIMITER //
CREATE PROCEDURE ObtenirSoutenancesFaibleNote()
BEGIN
  DECLARE id_soutenance INT;

  DECLARE cur CURSOR FOR
    SELECT id_soutenance
    FROM soutenance
    WHERE note < 10;

  OPEN cur;

  FETCH cur INTO id_soutenance;
  WHILE id_soutenance IS NOT NULL DO
    SELECT id_soutenance;
    FETCH cur INTO id_soutenance;
  END WHILE;

  CLOSE cur;
END //
DELIMITER ;
```

- Retourne l'ID de toutes les personnes du jury qui sont des professeurs

```
DELIMITER //
BEGIN
  -- Créer une table temporaire pour stocker les IDs des professeurs du jury
  CREATE TEMPORARY TABLE IF NOT EXISTS TempProfesseursJury (
    id_professeur INT
  );

  INSERT INTO TempProfesseursJury (id_professeur)
  SELECT JSON_UNQUOTE(JSON_EXTRACT(liste_ID_Jury, '$.id')) FROM jury;

  -- Sélectionner les IDs des professeurs qui sont dans le jury
  SELECT p.id_professeur
  FROM professeur p
  JOIN TempProfesseursJury tpj ON p.id_professeur = tpj.id_professeur;

  -- Supprimer la table temporaire à la fin de la procédure
  DROP TEMPORARY TABLE IF EXISTS TempProfesseursJury;
END //
DELIMITER ;
```

Vue :

- Afficher toutes les notes et moyenne d'un élève

```
SELECT
    u.id_User,
    u.nom,
    u.prenom,
    j.evaluations AS Note_Journal,
    r.note AS Note_Rapport,
    s.note AS Note_Soutenance,
    (COALESCE(j.evaluations, 0) + COALESCE(r.note, 0) + COALESCE(s.note, 0)) /
    (CASE
        WHEN j.evaluations IS NOT NULL AND r.note IS NOT NULL AND s.note IS NOT NULL THEN 3
        WHEN (j.evaluations IS NOT NULL AND r.note IS NOT NULL) OR (j.evaluations IS NOT NULL AND s.note IS NOT NULL)
        OR (r.note IS NOT NULL AND s.note IS NOT NULL) THEN 2
        ELSE 1
    END) AS Moyenne
FROM
    utilisateur u
LEFT JOIN
    ap a ON u.id_User = a.id_AP
LEFT JOIN
    journal j ON a.id_Journal = j.id_Journal
LEFT JOIN
    rapport r ON r.attrib_J = j.id_Journal
LEFT JOIN
    soutenance s ON s.attrib_J = j.id_Journal
WHERE
    u.id_User = 1;
```

On modifie le WHERE en bas en fonction de l'user qu'on veut afficher

- Pour chaque soutenance d'un élève, l'on renvoie le nom de l'élève avec meilleur note.

```
SELECT
    u.id_User,
    u.nom,
    u.prenom,
    s.note AS meilleure_note_soutenance
FROM
    utilisateur u
JOIN
    ap a ON u.id_User = a.id_AP
JOIN
    journal j ON a.id_Journal = j.id_Journal
JOIN
    soutenance s ON s.attrib_J = j.id_Journal
ORDER BY
    s.note DESC
LIMIT 1;
```

Curseur :

- On calcule la moyenne d'une promotion pour une évaluation.

```
CREATE OR REPLACE PROCEDURE calculate_average_evaluation(p_promotion_id IN NUMBER) IS
    CURSOR c_evaluations IS
        SELECT j.evaluations
        FROM ap a
        INNER JOIN cursus c ON a.id_Cursus = c.id_Cursus
        INNER JOIN journal j ON a.id_Journal = j.id_Journal
        WHERE c.id_Cursus = p_promotion_id;

    v_total_evaluations NUMBER := 0;
    v_count_evaluations NUMBER := 0;
    v_average_evaluation NUMBER;
BEGIN
    FOR r_evaluation IN c_evaluations LOOP
        v_total_evaluations := v_total_evaluations + r_evaluation.evaluations;
        v_count_evaluations := v_count_evaluations + 1;
    END LOOP;

    IF v_count_evaluations > 0 THEN
        v_average_evaluation := v_total_evaluations / v_count_evaluations;
        DBMS_OUTPUT.PUT_LINE('La moyenne des évaluations pour la promotion ' || p_promotion_id || ' est : ' || v_average_evaluation);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Aucune évaluation trouvée pour la promotion ' || p_promotion_id);
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Erreur : ' || SQLERRM);
END;
```

- On calcule le nombre d'apprentis que possède un Tuteur Pédagogique.

```
CREATE OR REPLACE PROCEDURE count_apprentices_of_a_tutor(p_tutor_id IN NUMBER) IS
    CURSOR c_apprentices IS
        SELECT id_AP
        FROM ap
        WHERE attrib_TP = p_tutor_id;

    v_count_apprentices NUMBER := 0;
BEGIN
    FOR r_apprentice IN c_apprentices LOOP
        v_count_apprentices := v_count_apprentices + 1;
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('Le Tuteur Pédagogique ' || p_tutor_id || ' a : ' || v_count_apprentices || ' apprenti(s).');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Erreur : ' || SQLERRM);
END;
```