Rapport du Mini-Projet – Module 110

Gestion de Notes avec Monitoring (Prometheus + Grafana)

1. But et fonctionnalités de l'application

L'objectif principal de l'application est de permettre à des enseignants ou administrateurs scolaires de :

- gérer des **élèves**, des **cours**, et des **notes pondérées** (de x1 à x5),
- **calculer automatiquement les moyennes** par élève et par cours,
- **surveiller en temps réel** le bon fonctionnement de l'application à l'aide de **Prometheus**,
- et **visualiser les métriques** sur un **tableau de bord Grafana**.

L'application est développée en **C# (.NET 8)** avec une interface **Windows Forms sombre** ergonomique.

2. Modélisation de la base de données (MCD + MLD)

La base de données relationnelle MariaDB contient 3 entités :

- **Élève** (id, nom)
- **Cours** (id, nom)
- **Note** (id, eleve_id, cours_id, valeur, pondération)

Le MLD comprend les relations suivantes :

- Un élève peut avoir plusieurs notes.
- Un cours peut être associé à plusieurs notes.
- Chaque note est liée à un élève et un cours.

Les scripts SQL fournis (`create_db.sql` et `insert_data.sql`) permettent de créer les tables et d'y insérer des données d'exemple.

3. Fiabilité et points critiques de l’application

Les points critiques incluent :

- Les erreurs de connexion à la base de données
- Les champs non remplis lors de l’ajout
- Les saisies invalides (ex. : pondération hors bornes)
- Le risque de moyenne incorrecte si pondérations mal gérées
- Une surcharge potentielle des métriques ou erreurs silencieuses

Pour cela :

- Nous **comptons les erreurs** via `error_count`
- Nous **mesurons les volumes critiques** avec :
    - `eleve_count`, `cours_count`, `note_count`
- Nous **calculons la moyenne globale pondérée** en temps réel
- Toutes ces données sont exposées automatiquement à Prometheus via `KestrelMetricServer`.

4. Paramétrage de Prometheus et Grafana

Prometheus :

- Port d'écoute : `1234`
- Scrape toutes les 5 secondes sur `http://localhost:1234/metrics`
- Fichier de configuration utilisé : `prometheus.yml`

Grafana :

- Port d’accès : `http://localhost:3000`
- Datasource : Prometheus (via `http://localhost:9090`)

- Dashboard importé via JSON fourni (`dashboard_gestion_notes.json`)
- Panneaux : compteurs, jauges, courbes de moyennes

5. Données surveillées (métriques)

- `eleve_count` → Nombre total d'élèves
- `cours_count` → Nombre total de cours
- `note_count` → Nombre total de notes
- `moyenne_globale_ponderee` → Moyenne pondérée globale
- `error_count` → Nombre d'erreurs capturées dans l'application

6. Captures d'écran

📸 Exemples à inclure :

- Page `/metrics` affichant les valeurs exposées
- Tableau de bord Grafana (barres, jauge de moyenne)
- Graphique en live d'un compteur ou d'une alerte

7. Alerte

Une **alerte est créée dans Grafana** :

- Condition : `moyenne_globale_ponderee < 10`
- Notification visuelle et panneau en rouge
- Capture d'écran à ajouter pour prouver l'alerte déclenchée

8. Conclusion (1) – Améliorations

Certaines améliorations n'ont pas pu être intégrées par manque de temps :

- Système d'authentification avec rôles

- Logs complets dans un fichier texte
- Export PDF ou Excel
- Moyenne par élève avec historique des notes

9. Conclusion (2) – Apprentissage

Ce projet m’a permis de :

- Mieux comprendre la **structure d’un projet WinForms**
- Utiliser **Prometheus pour exposer des métriques personnalisées**
- Configurer un **dashboard Grafana** avec alertes
- Automatiser la **surveillance d’une application logicielle** avec visualisation temps réel

10. Répartition du travail

Projet réalisé par Timoleaon, Maxime et Zackary