Grille de correction détaillée – EC03

Informations générales

- **Épreuve :** EC03 – Développement back-end avec BDD relationnelle + NoSQL
- **Durée :** 4h – Individuelle
- **Projet fil rouge :** SkillHub – Gestion des utilisateurs
- **Technologies imposées :** PostgreSQL (relationnel) + MongoDB (NoSQL), environnement Docker fourni

Critères d'évaluation et attentes

Critère	Intitulé officiel	Attentes côté correcteur	Points de vigilance
C7.1	Environnement installé	Vérifier que Docker et le projet se lancent correctement (`docker-compose up`). Fichiers `.env` cohérents, services PostgreSQL et MongoDB accessibles.	Vérifier captures ou logs fournis. Scripts de lancement fonctionnels.
C7.2	Utilisation du terminal	Scripts et commandes correctement documentés dans le README (`setup.sh` , migrations, commandes Mongo).	Commandes absentes ou trop sommaires = points en moins.
C8.1	POO & MVC	Code structuré en entités, contrôleurs, repositories. Respect des principes MVC.	Vérifier conventions de nommage, arborescence claire (`src/controllers` , `src/models` , etc.).

Critère	Intitulé officiel	Attentes côté correcteur	Points de vigilance
C8.2	Clarté / maintenance	Code lisible, commenté. README expliquant architecture et conventions.	Attention aux projets « spaghetti » sans séparation claire.
C9	Optimisation & sécurité serveur	Vérifier présence de configs sécurisées (`.env` , headers HTTP, gestion erreurs).	Mot de passe en clair ou hardcodé = pénalité forte.
C10	BDD relationnelle	Migrations ou schéma SQL présent. Entité `User` persistée dans PostgreSQL avec cohérence des données.	Vérifier dump fourni. Tables et colonnes correctement définies.
C11	BDD NoSQL	Collections cohérentes (sessions, logs, etc.). Dump JSON fourni.	Vérifier que les données sont exploitables. Pas d'artefacts vides.
C12	Justification des choix	README doit expliquer pourquoi certaines données vont dans SQL vs NoSQL.	Argumentation faible ou absente = gros manque.
C13	Sauvegarde / restauration	Scripts `backup.sh` et `restore.sh` présents et fonctionnels.	Vérifier syntaxe des commandes et lisibilité.

Livrables à corriger

- **Code source :** projet structuré en MVC
- **README.md :** explications techniques, commandes terminal, arbitrages SQL/NoSQL
- **Dumps :** SQL (PostgreSQL) + JSON (MongoDB)
- **Scripts :** `backup.sh` , `restore.sh`
- **Captures éventuelles :** exécution des commandes terminal

Barème indicatif (sur 20 points)

Axe évalué	Points
Environnement & terminal (C7.1, C7.2)	/3
Architecture POO/MVC (C8.1, C8.2)	/4
Sécurisation & optimisation (C9)	/2
Base SQL – cohérence et usage (C10)	/4
Base NoSQL – cohérence et usage (C11)	/3
Justification technique (C12)	/2
Sauvegarde / restauration (C13)	/2

Points positifs attendus

- Code clair, structuré, respectant le modèle MVC
- Utilisation cohérente des deux bases : SQL pour les données structurées, NoSQL pour les logs ou historiques
- Dumps propres et réutilisables
- Scripts de backup/restauration simples et exécutables
- README détaillé, montrant une bonne compréhension des arbitrages techniques

Erreurs fréquentes à surveiller

- Absence de séparation MVC (tout le code dans un seul fichier)
- Base NoSQL sous-exploitée ou laissée vide
- Mots de passe hardcodés dans le code source
- Dumps corrompus ou non exploitables
- Scripts de backup/restore incomplets ou inutilisables
- README superficiel (exemple : « J’ai utilisé Mongo car c’est demandé »)

Rappel pédagogique

Cette épreuve doit valider que l'apprenant :

- Sait manipuler **deux bases complémentaires** dans un contexte professionnel
- Maîtrise la **programmation objet et l'architecture MVC** côté back-end
- Est capable de **sécuriser et documenter son environnement**
- Peut justifier ses choix techniques et produire des scripts de maintenance

Elle constitue une étape charnière vers **EC04 (API sécurisée et documentée)** où l'accent sera mis sur l'exposition de ces données.