TP - Microservices

Ce TP sert de démarrage commun pour le projet qui viendra après, le code, la documentation et le schéma serviront de bases pour vos projets.

⚠️ Quand je parle de projet dans ce document, je parle de votre projet final, à terminer pour la fin du cours (le Jeudi 12/02/2026)

Attentes

Code: Mettre en place une architecture avec une Gateway et au moins 2 Services (avec chacun une BDD) et 1 Front
Documentation:

- Expliquez le contexte de votre projet et sa fonctionnalité principale.
- Faites un schéma de l'architecture avec 1 gateway et au moins 2 services: Ce schéma représente ce que vous prévoyez pour votre projet, pensez à la faisabilité.
- Expliquez pour votre projet, l'architecture (rôle de chaque "bloc"), la répartition des services (qui fait quoi), les choix technologiques (choix de DB, ORM, Framework) - *en excluant les* ***Contraintes***

Contraintes

NodeJS avec Express pour les APIs (Gateway et Services)
La Gateway doit être faite avec: https://www.npmjs.com/package/http-proxy-middleware
La Gateway et l'ensemble des Services doivent être Dockerisés.
Mettre des logs sur tous les services (avec https://www.npmjs.com/package/morgan par exemple)

Format du rendu

Code: Lien Github (à référencer sur le sheet de suivi: Suivi Projets 2025/2026)
Documentation: README.md à la racine de vos repos

Objectif pédagogique

Préparer la classe aux choses imposées pour le projet final: Architecture en microservices avec gateway(s), NodeJS/Express/Docker.
Présenter pas à pas la mise en place d'une stack.
Apporter des notions d'architecture pour se rendre compte du périmètre complet du Fullstack
Sensibiliser sur les outils comme ESLint (vérification du code) ou Husky (pre-commit hook, se déclenche avant un commit, permet de bloquer le commit si par exemple un test ou le linter retournent une erreur)

Barème

Documentation	**Critère**	**Barème**
	Mise en forme du README	/1
	Explication projet	/2
	Schéma de l'architecture des communications	/2
	Explications des choix technologiques	/2
Code	**Critère**	**Barème**
	Gateway dockerisée fonctionelle	/1
	2 Services dockerisés fonctionnels	/1
	Le front dockerisé fonctionnel	/1

Bonus

Les éléments ci après ne comptent pas dans le barème mais permettent d'aller plus loin, certain pourraient servir pour la réalisation de votre projet

Mettre en place nodemeon => pour ne pas avoir à faire npm start tout le temps

Mettre en place ESLint, puis Husky => pour avoir une syntaxe identique entre collaborateurs et avec Husky le code non valide ne pourra pas être commit par exemple

Et si vraiment vous avez le temps…

Faire un nouveau service Express mais avec TypeScript => TypeScript pourrait devenir très pertinent en fonction de votre projet

Mettre en place quelques tests => Pour s'assurer de ne jamais casser les logiques importantes dans votre code

Vraiment le temps…

Trouver une manière de mutualiser du code entre 2 services Express => pour moins de code à maintenir, et des services facilement duplicables