



Projet Clean code

Rattrapage Epitech – février 2026






Le projet pédagogique

- Application simple « Todo List » : <https://github.com/docker/getting-started-todo-app>
- Application monolithique
- Nécessite d'être refondue
- Mise en place de l'authentification et du profil utilisateur conforme RGPD




Livrables attendus

- Application refondue :
 - Séparation frontend / backend
 - Migration en TypeScript
 - Métier découplé des briques techniques
 - Couverture de tests des cas métiers (E2E, intégration et unitaire)
 - Projet tournant sous Docker Compose
 - Code dans un repository Git (front + back)
 - Documentation technique et suivi des décisions
- 




Livrables attendus : Backend

- 
- Dernière version LTS de Node
 - Dépendances : Express, Mysql2, Sqlite3
 - Dépendances de dev : TypeScript, ESLint, Prettier
 - Framework pour tests unitaire et intégration : Jest
 - DockerFile



Livrables attendus : Frontend

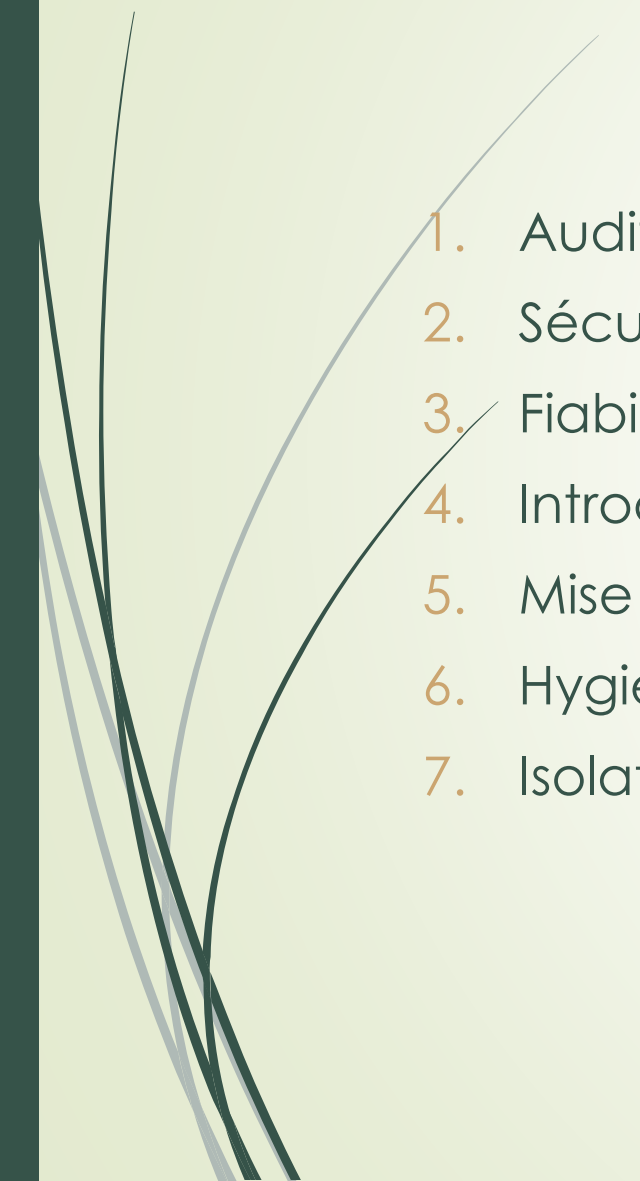
- 
- Dernière version LTS de Node
 - Dépendances : React, Bootstrap, Sass, FontAwesome
 - Dépendances de dev : TypeScript, Vite, ESLint, Prettier
 - Framework pour tests E2E : Playwright
 - DockerFile

Organisation pratique & calendrier

- Projet individuel
- Mono-repository GitHub :
 - Frontend (source + tests)
 - Backend (source + tests)
 - Procédure de lancement
 - Documentation des décisions
- Échéance : le **23 février 2026 à 12h**
- Soutenance du projet :
 - 15mn de présentation et démonstration
 - 5mn de questions



Processus de refonte

- 
1. Audit et état initial
 2. Sécurisation par les tests
 3. Fiabilisation de l'environnement
 4. Introduction de TypeScript
 5. Mise à jour de Node
 6. Hygiène du projet
 7. Isolation de l'infrastructure



Processus de refonte : Audit et état initial

- Identifier :
 - les responsabilités mélangées
 - les dépendances fortes (framework, DB, filesystem)
 - les zones à risque (persistance SQLite, couplage API ↔ DB)
- Mise en place d'un repository Git :
 - Commit de chaque étape indépendante
 - Permet de pouvoir revenir à un état stable en cas de souci
 - Inutile de push sur un repository dédié, les commit locaux suffisent



Processus de refonte : Sécurisation par les tests

- Tests E2E (frontend) :
 - Installer et configurer Playwright
 - Tester les parcours utilisateurs clés
- Tests backend existants (Jest) :
 - Configurer Jest (déjà présent)
 - Exécuter les tests existants
- Gestion de la base de données en tests :
 - Séparer les tests de persistance des tests des routes

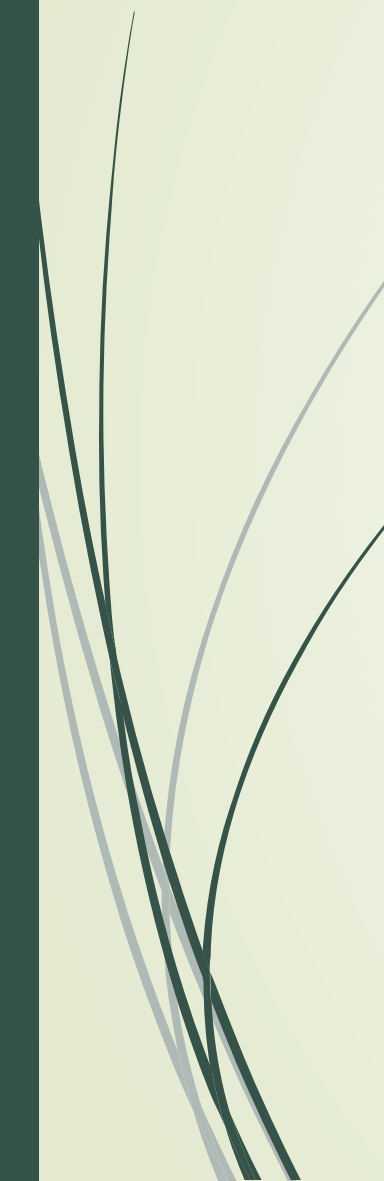


Processus de refonte : Fiabilisation de l'environnement

- Génération du lockfile
 - Vérifier :
 - installation
 - exécution des tests
 - versions critiques
- 

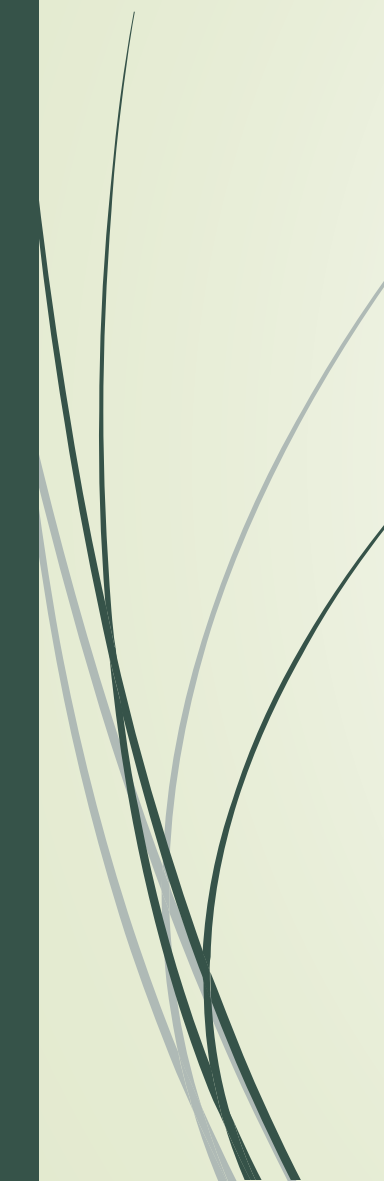


Processus de refonte : Introduction de TypeScript

- Ajouter TypeScript
 - Aucune modification de structure
 - Renommer les fichiers progressivement
 - Corriger uniquement les erreurs bloquantes !
- 



Processus de refonte : Mise à jour de Node

- Mettre à jour Node après stabilisation
 - Vérifier :
 - tests
 - compatibilité des dépendances
 - Ne rien refactorer en même temps !
- 



Processus de refonte : Hygiène du projet

- Gestion des dépendances :
 - Séparer :
 - « dependencies » → runtime
 - « devDependencies » → outils
 - Mettre à jour progressivement (une dépendance à la fois)
- Linting & règles d'architecture :
 - Configurer ESLint
 - Ajouter « dependency-cruiser »
 - Interdire certaines dépendances (ex: DB dans le domaine)



Processus de refonte : Isolation de l'infrastructure

- Créer des interfaces (ports) :
 - SqliteRepository
 - InMemoryRepository
 - Injecter l'implémentation selon l'environnement
 - Test de non-régression structurelle :
 - Ajouter un test interdisant sqlite3 en test
 - Bloquer toute régression future
- 