Auto-évaluation – Laboratoire 3

Nom :Maksym Pravdin Code permanent : PRAM86290201
Cours : LOG430 – Architecture Logicielle

Session Été 2025

Informations générales

- URL du dépôt GitHub/GitLab : https://github.com/PraMaks/LOG430_Labo_0
- Langage utilisé : Python et JavaScript
- Framework utilisé : Django et Express
- Outils de tests : Jest

1. Autoévaluation par section

1.1 Structuration et REST API

Elements	Oui	Non	Commentaire / Justification
Les routes REST sont bien définies et respectent les conventions REST	✓		Oui consulter la documentation Swagger UI
L'architecture suit un modèle MVC ou hexagonal (ou une autre architecture qui sera justifier)	✓		Oui, côté frontend et backend les fichiers sont séparés
Les URIs sont bien structurées (ex : /api/v1/resource)	✓		Oui consulter la documentation Swagger UI
La couche API est clairement séparée de la logique métier	✓		Oui, consulter la documenation Swagger

1.2 Documentation des API

Elements	Oui	Non	Commentaire / Justification
La documentation Swagger (OpenAPI) est présente	✓		Oui consulter la documentation Swagger UI ou voir le fichier de preuve
Les méthodes, statuts, entrées/sorties sont décrites	✓		Oui consulter la documentation Swagger UI
Swagger UI ou Redoc est intégré	✓		Oui consulter la documentation Swagger UI
Des exemples de requêtes/réponses sont fournis	✓		Oui consulter la documentation Swagger UI

1.3 Sécurité et accessibilité

Elements	Oui	Non	Commentaire / Justification
CORS est configuré correctement	✓		Oui voir dans app.js
Une authentification est implémentée (token statique, Basic Auth, JWT)	✓		Oui, mot de passe chiffrée et génération d'un token (génération custom pas statique)
Les endpoints sensibles sont protégés	✓		Oui, côté frontend et backend les URLS sont protégés (backend = Header authorization et frontend = decorateurs)

1.4 Tests et validation

Elements	Oui	Non	Commentaire / Justification
Une collection Postman ou equivalente est fournie	✓	✓	Pour la documentation Swagger un exemple est fourni techniquement pour chaque route

Des tests automatisés (JUnit, MockMVC, etc.) sont présents	✓		Jest est utilisée pour lancer des tests unitaires 'supertest'
Les tests sont intégrés à la CI/CD	✓		Le pipeline reste fonctionnel depuis le labo0

1.5 Déploiement et exécution

Elements	Oui	Non	Commentaire / Justification
L'API est conteneurisée avec Docker	✓		Une image du backend est créé avec Dockerfile
Les instructions d'exécution sont claires dans le README.md	✓		Oui, fichier mis à jour
L'API est fonctionnelle en local ou via conteneur	✓		Oui, les 2 fonctionnent car le conteneur expose le même port qu'en local

1.6 Bonnes pratiques REST

Elements	Oui	Non	Commentaire / Justification
Respect des verbes HTTP (GET, POST, PUT, DELETE, PATCH)	✓		Oui, cela est corrigé du dernier labo
Utilisation de codes HTTP standard (200, 201, 400, etc.)	✓		Oui, mis à jour aussi
Pagination, tri et filtrage implémentés si pertinent		✓	Pas fait encore, c'est marqué dans le rapport
Messages d'erreur structurés et utiles	✓		Oui tous les messages d'erreurs sont envoyés avec le même format JSON que dans l'énoncé

2. Réflexion personnelle

1. Quelles sont les principales difficultés rencontrées lors de l'exposition de l'API RESTful?

S'adapter aux nouvelle règles pour les organiser, car ce n'est plus de la même manière que j'ai vu en technique.

2. Qu'avez-vous appris en matière de bonnes pratiques REST?

La différence pour quand utiliser POST ou PUT, avant je pensais que POST était uniquement pour les ajouts et PUT uniquement pour les mises à jour.

3. Quels choix techniques avez-vous faits pour la documentation, les tests et la sécurité?

Documentation est faite avec Swagger, ça m'a permis de se pratiquer à l'utiliser, car je dois la faire en LOG680 aussi.

Pour les tests, j'ai utilisé Jest et supertest, car je connaissais déjà la syntaxe et j'étais familier avec.

Pour la sécurité, j'ai appris à utiliser bcrypt pour chiffrer les mots de passes des utilisateurs et j'ai appris comment implémenter les tokens sans utiliser de librairies JWT de manière custom et non statique, car chaque token est différent

4. Que souhaiteriez-vous améliorer dans un futur projet API?

Pour ce laboratoire je souhaite améliorer mes tests backend, ajouter le support de pagination et mettre un load balancer.