

title: "Rapport de Test – cicd-todo-app"

author: "Evan Sottile & Diego Teixeira – GRP3D"

date: "2025"

---

## 1. Introduction

---

### 1.1 Titre

P\_Test – cicd-todo-app

### 1.2 Description

Dans le cadre du module ICT-450 (Test), nous avons travaillé sur une application Todo App qui nous a été fournie, avec un frontend en Vue.js, un backend en Node.js/Express et une base MySQL. Le but du projet n'était pas de développer l'application, mais plutôt d'apprendre à la tester correctement et à adopter une vraie démarche qualité.

## 2. Tests manuels et classification des bugs

---

### 2.1 Compte / Profil / Authentification

Le lien vers l'oubli de mot de passe ne fonctionne pas.

"Profile" doit être écrit "Profil".

L'erreur affichée lors de la création d'un profil avec une adresse mail déjà existante n'est pas claire.

Une erreur apparaît alors qu'aucune adresse email n'a encore été saisie.

Changement de nom : si on met un nom trop long, l'application crash.

---

### 2.2 NPA / Validation des données

Si on met des lettres dans le NPA, l'application crash au lieu d'afficher une erreur. v Le NPA devrait :

- accepter uniquement des chiffres
  - afficher un message d'erreur clair
  - ne pas faire cracher l'application
- 

### 2.3 Tâches (TODO)

On peut créer des tâches (todo) dans le passé, ce qui ne devrait pas être autorisé.

La description ne se réinitialise pas après la création d'une tâche.

---

## 2.4 Description des tâches

La description trop longue déborde visuellement.

Une description extrêmement longue provoque un crash (limite dépassée).

Une description contenant un mélange de gras, italique, souligné, barré ne peut pas être recherchée.

---

## 2.5 Barre de recherche

Une tâche composée de seulement deux lettres ne peut pas être recherchée.

Une tâche dont le texte est trop long ne s'affiche pas dans les résultats.

Lorsqu'on crée une tâche pendant qu'une recherche est active, elle apparaît quand même dans la liste filtrée.

---

## 2.6 Interface / Texte général

Footer mal écrit :

- "tout" devrait être "tous"
- "réserver" devrait être "réservés"

## 2.7 Classification des bugs

Catégorie	Bug	Sévérité	Gravité	Priorité	Justification
Compte / Profil	Crash si nom trop long	Critique	Bloquante	Haute	Crash -> fonctionnalité brisée et bloque l'usage
Compte / Profil	Lien "oubli mot de passe" KO	Majeure	Bloquante	Haute	Empêche récupérer l'accès au compte
Compte / Profil	Erreur email déjà existant pas claire	Mineure	Mineure	Moyenne	Perturbant mais contournable
Compte / Profil	Erreur affichée sans avoir mis d'email	Majeure	Sérieuse	Haute	Message erroné qui empêche la création correcte
Compte / Profil	"Profile" → "Profil"	Triviale	Cosmétique	Basse	Purement esthétique
NPA	Crash si lettres dans NPA	Critique	Bloquante	Haute	Crash immédiat
NPA	Mauvaise validation du NPA	Majeure	Sérieuse	Moyenne	Données invalides acceptées
Tâches	Créer une tâche dans le passé possible	Mineure	Mineure	Basse	Gênant mais pas bloquant

Catégorie	Bug	Sévérité	Gravité	Priorité	Justification
Tâches	Description non réinitialisée	Mineure	Mineure	Basse	Mauvaise UX sans impact critique
Description	Débordement visuel description longue	Mineure	Mineure	Moyenne	UI dégradée
Description	Crash avec description très longue	<b>Critique</b>	<b>Bloquante</b>	<b>Haute</b>	Crash direct
Description	Texte riche non recherchable	Majeure	Sérieuse	Moyenne	Fonction recherche partiellement brisée
Recherche	Tâche 2 lettres non trouvable	Mineure	Mineure	Basse	Limite mais non bloquante
Recherche	Tâche très longue non affichée	Majeure	Sérieuse	Haute	Perte de données visibles
Recherche	Tâche créée apparaît malgré filtre actif	Majeure	Mineure	Moyenne	Comportement incohérent mais non bloquant
Interface	Fautes dans le footer	Triviale	Cosmétique	Basse	Aucun impact fonctionnel

### 3. Stratégie et plans de tests

#### Objectifs de test

- Vérifier la conformité fonctionnelle de l'application par rapport aux spécifications.
- Identifier les comportements inattendus via des tests manuels.
- Renforcer la qualité du backend grâce à des tests unitaires Jest.
- Établir une base de tests automatisés pour une intégration CI/CD future.

#### 3.1 Tests manuels

Réalisés sur l'ensemble de l'application :

- Authentification (login/signup)
- Profil utilisateur
- Création/édition/suppression de tâches
- Recherche
- Interface & ergonomie
- Gestion du thème sombre/clair

Ces tests permettent d'observer le comportement réel depuis la perspective de l'utilisateur.

#### 3.2 Tests unitaires backend (Jest)

Les tests unitaires couvrent :

- les contrôleurs (`auth`, `todo`, `user`)
- le middleware d'authentification
- les fonctions de validation

## 4. Description des tests backend effectués

---

Les tests unitaires ont été développés avec Jest, en respectant les règles suivantes :

- mock des modèles Sequelize
- mock de bcrypt et jsonwebtoken
- tests isolés (aucun appel à la base de données)
- couverture des cas d'erreurs + cas valides

### Fonctionnalité testés

#### 4.1 Auth Controller

Tests effectués :

- Login avec email vide
- Login avec email inexistant
- Login avec mauvais mot de passe
- Signup avec email invalide
- Signup avec email déjà existant
- Signup valide renvoyant un token

#### 4.2 User Controller

- Lecture du profil utilisateur
- Mise à jour du profil
- Gestion du cas où l'utilisateur n'existe pas

#### 4.3 Todo Controller

- création de todo avec titre vide → rejet
- création valide
- mise à jour inexiste → erreur 404

#### 4.4 Middleware d'authentification

- absence de token → 401
- token valide → next()

---

## 5. Résultats de la campagne de test unitaire et e2e

---

Manque capture d'écran des tests unitaires et e2e.

## Synthèse générale

Module	Résultat	Commentaire
Auth Controller	Réussi	Couverture complète des cas critiques
User Controller	Réussi	Gestion correcte des erreurs
Todo Controller	Réussi	Validation robuste
Auth Middleware	Réussi	Vérification correcte du token
Validations	Réussi	Permet d'éviter des crashes

## Erreurs détectées grâce aux tests unitaires

- mauvaise gestion des valeurs nulles dans certains contrôleurs
- messages d'erreur incohérents
- besoin d'un module de validation dédié

## Taux de réussite

100% des tests unitaires passant en vert après corrections.

---

## 6. Rapport de tests (statut, conclusion)

---

### Statut actuel

- Les bugs backend majeurs sont corrigés.
- Les tests unitaires garantissent un comportement stable et cohérent.
- Le backend est désormais robuste et prêt pour une pipeline CI/CD.

### Conclusion

La campagne de test a permis :

- d'identifier des faiblesses majeures
  - d'améliorer significativement la validation des données
  - d'éviter plusieurs crashes critiques
  - d'obtenir un backend stable et maintenable
- 

## 7. Corrections apportées au code

---

Suite aux tests manuels et aux tests unitaires réalisés, plusieurs corrections ont été apportées au code afin d'améliorer la stabilité, la robustesse et la qualité globale de l'application.

## 7.1 Corrections backend

Les corrections backend concernent principalement la validation des données et la gestion des erreurs.

- Ajout de validations sur les champs critiques (email, mot de passe, titre de tâche)
- Gestion correcte des valeurs nulles ou vides
- Prévention des crashes causés par des chaînes trop longues
- Amélioration des messages d'erreur retournés par l'API
- Sécurisation du middleware d'authentification (vérification stricte du token)

Ces corrections ont été validées à l'aide des tests unitaires Jest afin d'éviter toute régression.

## 7.2 Corrections frontend (limitées)

Certaines corrections mineures ont été identifiées côté frontend, notamment :

- amélioration de certains messages d'erreur
- corrections orthographiques dans l'interface
- ajustements mineurs de comportement (UX)

Les corrections frontend restent cependant hors du périmètre principal de ce projet, qui était orienté tests.

---

# 8. Usage de l'intelligence artificielle dans le projet

L'intelligence artificielle (ChatGPT/Gemini) a été utilisée comme **outil de support et de référence** tout au long du projet, afin de disposer d'une **base de travail** pour la rédaction, les tests et l'analyse. Elle a permis de gagner du temps tout en conservant une validation humaine à chaque étape.

## 8.1 Aide à la rédaction et à la correction du rapport

L'IA a été utilisée pour :

- reformuler certaines sections afin d'améliorer la lisibilité
- corriger les fautes d'orthographe et de grammaire
- uniformiser le style d'écriture sur l'ensemble du document
- aider à la mise en forme du rapport en Markdown

## 8.3 Aide à la création des tests unitaires

L'intelligence artificielle a été utilisée comme **base de réflexion** pour :

- l'écriture des tests unitaires avec Jest
  - la génération de mocks pour les dépendances (bcrypt, jsonwebtoken, modèles Sequelize)
- 

# 9. Bilan du projet

## 9.1 Bilan des fonctionnalités testées

L'ensemble des fonctionnalités principales de l'application a été testé :

- Authentification et inscription
- Gestion du profil utilisateur
- Gestion des tâches (création, modification, suppression)
- Recherche
- Validation des données

Toutes ces fonctionnalités ont pu être validées, et les bugs critiques ont été identifiés puis corrigés côté backend.

## 9.2 Bilan de la campagne de test

La campagne de test a permis :

- d'identifier de nombreux bugs, dont plusieurs critiques
- d'améliorer significativement la stabilité de l'application
- de mettre en place une base solide de tests unitaires
- de renforcer la qualité du code backend

## 9.3 Bilan personnel

Ce projet a permis :

- d'acquérir une méthodologie de test structurée
- de comprendre l'importance de la validation des données
- de découvrir l'écriture de tests unitaires et e2e avec Jest et Cypress
- de travailler efficacement en collaboration

---

# 10. Conclusion générale

---