

CAHIER DES CHARGES

Backlog Produit & Spécifications

Projet de Validation Logicielle (ISTQB)

Todo App (Spring Boot + React)

Équipe QA

Mohamed Dhia Eddine Thabet - Test Manager

Mohamed Aziz Dridi - Tester

Table des matières

1	Introduction et Portée	2
1.1	Objectifs du Projet	2
1.2	Acteurs (Personas)	2
2	Product Backlog (Synthèse)	2
3	Spécifications Détaillées	3
3.1	Module 1 : Gestion des Tâches (Frontend)	3
3.2	Module 2 : API Backend (Intégration)	3
3.3	Module 3 : Performance (Non-Fonctionnel)	3

1 Introduction et Portée

Ce document définit les spécifications fonctionnelles et techniques pour l'application **Todo App**. Il sert de référence pour le développement (Dev) et la validation (QA).

1.1 Objectifs du Projet

L'objectif est de fournir une application de gestion de tâches minimaliste mais robuste, permettant aux utilisateurs de créer, lire, mettre à jour et supprimer des tâches (CRUD) en temps réel.

1.2 Acteurs (Personas)

- **Utilisateur Final** : Personne souhaitant organiser ses tâches quotidiennes via une interface web.
- **Système (API)** : Le service backend qui traite les données et assure la persistance.

2 Product Backlog (Synthèse)

Le tableau ci-dessous recense l'ensemble des User Stories (US) et Exigences Non-Fonctionnelles (NFR).

ID	Titre / Description	Module	Priorité
US-01	Affichage de la liste des tâches En tant qu'utilisateur, je veux voir l'ensemble de mes tâches pour planifier ma journée.	Frontend	Haute
US-02	Création d'une tâche En tant qu'utilisateur, je veux ajouter une nouvelle tâche à ma liste.	Frontend	Haute
US-03	Modification d'une tâche En tant qu'utilisateur, je veux changer le texte ou marquer une tâche comme "Faite".	Frontend	Moyenne
US-04	Suppression d'une tâche En tant qu'utilisateur, je veux retirer une tâche de la liste définitivement.	Frontend	Moyenne
US-05	API CRUD Complète Le système doit exposer des endpoints REST pour gérer la persistance des données.	Backend	Haute
US-06	Robustesse sous charge Le système doit supporter 50 utilisateurs simultanés sans perte de données.	Performance	Haute
NFR-01	Qualité du Code Le code doit respecter les standards SonarQube (Pas de code smells critiques).	Qualité	Basse

3 Spécifications Détaillées

3.1 Module 1 : Gestion des Tâches (Frontend)

US-01 : Affichage de la liste

Description : Au chargement de l'application, la liste des tâches existantes doit être récupérée depuis le serveur et affichée.

- **Règle de Gestion :** Si la liste est vide, afficher un message ou un état vide (cependant, une règle backend actuelle recrée une tâche par défaut, voir Bug FRONT-002).
- **Critères d'Acceptation :**
 1. La page charge sans erreur console.
 2. Les tâches s'affichent avec leur statut correct (barré si fait).

US-02 : Création d'une tâche

Description : L'utilisateur saisit du texte dans un champ input et valide.

- **Règle de Gestion :** Le champ ne doit pas être vide.
- **Flux :** Saisie → Touche Entrée → Appel POST API → Ajout au DOM.

US-04 : Suppression d'une tâche

Description : Chaque tâche possède un bouton de suppression (Corbeille/Croix).

- **Règle de Gestion :** La suppression doit être définitive.
- **Critères d'Acceptation :** La tâche disparaît de l'écran immédiatement.

3.2 Module 2 : API Backend (Intégration)

US-05 : Endpoints REST

Le backend Spring Boot doit exposer les ressources suivantes sur `/api/todos` :

- **GET / :** Retourne la liste JSON des tâches.
- **POST / :** Crée une tâche (Body : { "task": "..." }).
- **PUT /{id} :** Met à jour une tâche.
- **DELETE /{id} :** Supprime une tâche.

3.3 Module 3 : Performance (Non-Fonctionnel)

US-06 : Performance & Concurrency

Contexte : L'application sera utilisée par plusieurs utilisateurs simultanément.

- **Exigence :** Temps de réponse < 200ms pour 99% des requêtes.
- **Contrainte Critique :** Garantir l'unicité des IDs générés (Thread-Safety).
- **Validation :** Test de charge JMeter avec 50 Threads.