# 🧪 Test technique – Développeur Fullstack (React + TypeScript)

### 🎯 Objectif :

Développer une application de gestion de tâches simple, destinée à une équipe interne. L'application doit permettre d'ajouter, consulter, supprimer et (optionnellement) mettre à jour le statut de tâches.

---

### 🧱 Contexte :

Tu es chargé de développer une **application fullstack en TypeScript** avec :

- Un backend en **Fastify** ou **Express**
- Un frontend en **React**

L'application gère une liste de tâches. Chaque tâche contient les informations suivantes :

- id (string ou number, généré automatiquement)
- title (string)
- description (string)
- status (enum : "pending" ou "done")

Pas besoin de base de données : les données peuvent être stockées **en mémoire** (ex. : tableau local).

---

### 🔧 Partie Backend (Fastify ou Express en TypeScript) :

Créer une API REST avec les endpoints suivants :

| Méthode | URL | Description |
|---|---|---|
| GET | /tasks | Retourne la liste des tâches |
| POST | /tasks | Crée une nouvelle tâche |
| DELETE | /tasks/:id | Supprime une tâche par son ID |
| PATCH | /tasks/:id | *(Optionnel)* Met à jour le statut de la tâche |

**Contraintes :**

- L'API doit être entièrement typée avec TypeScript.
- Le code doit être bien structuré (routes, services, types...).
- Une gestion d'erreurs simple est attendue (ex. : tâche non trouvée).

- L'utilisation de bibliothèques comme **Zod** (validation des entrées) est un **plus apprécié**.

---

### 💻 Partie Frontend (React + TypeScript) :

Créer une interface utilisateur avec React (sans obligation de design complexe, mais une UI claire et lisible).

**Fonctionnalités attendues :**

- Affichage de la liste des tâches (GET /tasks)
- Formulaire pour ajouter une tâche (POST /tasks)
- Suppression d'une tâche (DELETE /tasks/:id)
- *(Bonus)* Mise à jour du statut via un bouton ou un toggle (PATCH /tasks/:id)

**Contraintes :**

- L'application doit être codée en **TypeScript**.
- Les composants doivent être réutilisables autant que possible.
- Les appels API doivent être isolés (via un service ou hook).

**Bonus appréciés :**

- Utilisation de **TanStack Query (React Query)** pour la gestion de cache et de requêtes.
- Utilisation de **Zustand** pour la gestion d'état globale si besoin.
- Utilisation de **React Hook Form** pour la gestion des formulaires.
- Utilisation de **Zod** pour le typage et la validation côté client.

---

### 📂 Livrables attendus :

- Le code source complet (fichiers zip ou lien GitHub public/privé).
- Un fichier README.md clair contenant :
    - Instructions pour lancer le **backend**
    - Instructions pour lancer le **frontend**
    - Stack technique utilisée
    - (Optionnel) Tes choix techniques ou commentaires

---

### 🧠 Éléments d'évaluation :

Nous évaluerons :

- La qualité du code (lisibilité, structure, typage)

- L’architecture globale (clarté des responsabilités)
- La rigueur dans la gestion des types et des erreurs
- La logique de développement (organisation, simplicité, efficacité)
- La capacité à livrer un projet fonctionnel et lisible

---

**⏱ Temps estimé : 1h30 à 2h**