**ELBARCHAOUI Soufiane : IAWM TA**

**Rapport Technique Application to-do list ( tp2)**

**Présentation Générale du Projet**

***Présentation générale du projet :***

L’objectif du projet est de réaliser une application de gestion de tâches (« to-do list ») permettant à un utilisateur de :

Consulter la liste des tâches existantes

Ajouter une nouvelle tâche

Modifier une tâche (titre, statut complété)

Supprimer une tâche

L’application est structurée en deux parties :

### ***Architecture de l’application***

L’application est composée de deux parties :

**Frontend (React.js)** : interface graphique permettant d’ajouter, afficher et supprimer des tâches.

**Backend (Express.js)** : API REST qui gère les requêtes HTTP, les données des tâches et la communication avec la base de données.

La communication entre le frontend et le backend se fait via des appels API HTTP (GET, POST, DELETE...).

### ***Prérequis***

Avant le déploiement, les éléments suivants sont nécessaires :

Un **compte Railway**.

Node.js et npm installés sur la machine.

Git pour la gestion de version.

Une application fonctionnelle en local (frontend et backend).

### ***Étapes de déploiement***

#### **A. Déploiement du backend (Express.js)**

1. Créer un nouveau projet sur https://railway.com/
2. Connecter son dépôt GitHub contenant le backend ou importer les fichiers.
3. Définir les **variables d’environnement**
4. Railway installe automatiquement les dépendances et lance le serveur.
5. Une **URL publique** est générée pour accéder à l’API <https://full-stack-proj-production.up.railway.app/>

#### **Déploiement du frontend (React.js)**

Exécuter la commande de build :

npm run build

Importer le dossier build dans un nouveau projet Railway

Configurer le frontend pour qu’il pointe vers l’URL du backend (modification des appels API dans le code).

Déploiement terminé : une URL est générée pour le frontend.

<https://full-stack-proj-production.up.railway.app/>

Connexion Frontend ↔ Backend

Tester l’ajout, la suppression et l’affichage des tâches via l’interface React.

***Problèmes rencontrés & solutions***

### **Absence de structure dans le** package.json **du backend**

**Problème** :  
Le projet backend n’avait pas de fichier package.json correctement configuré, ce qui empêchait l’installation des dépendances et le démarrage du serveur.

**Solution** :  
Un fichier package.json a été créé via npm init, puis enrichi avec les scripts utiles (start, dev) et les dépendances nécessaires (express, mongoose, dotenv, nodemon). Cela a permis d’automatiser le lancement de l’application et de centraliser les dépendances.

### **Le dossier node\_modules était suivi par Git**

**Problème** :  
Au début du projet, le dossier node\_modules a été accidentellement ajouté et versionné dans le dépôt Git, ce qui a considérablement augmenté la taille du dépôt et causé des lenteurs lors des pulls et pushes.

**Solution** :  
Nous avons utilisé la commande git rm -r --cached node\_modules pour supprimer le dossier du suivi Git sans supprimer les fichiers localement. Ensuite, l’entrée node\_modules/ a été ajoutée dans le fichier .gitignore pour éviter qu’il soit suivi à l’avenir.

### ***Configuration du fichier package.json pour le backend***

Le fichier package.json gère toutes les dépendances et scripts nécessaires au projet Node.js backend.

#### Exemple de structure utilisée :

json

CopierModifier

{

"name": "my-node-backend",

"version": "1.0.0",

"description": "A backend API for my app",

"main": "index.js",

"scripts": {

"start": "node index.js",

"dev": "nodemon index.js",

"test": "echo \"Error: no test specified\" && exit 1"

},

"dependencies": {

"express": "^4.17.1",

"mongoose": "^5.10.9",

"dotenv": "^8.2.0"

},

"devDependencies": {

"nodemon": "^2.0.7"

},

"author": "Your Name",

"license": "ISC",

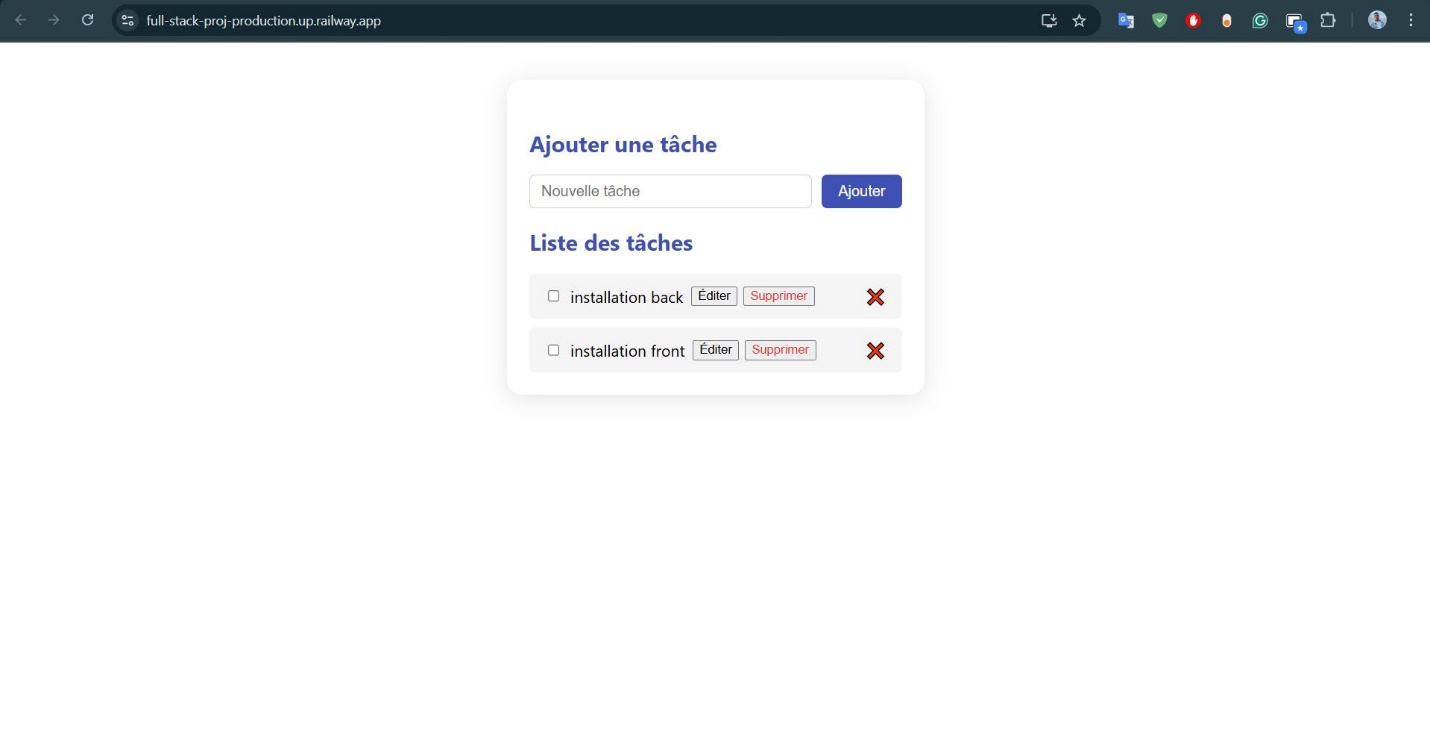
"engines": {

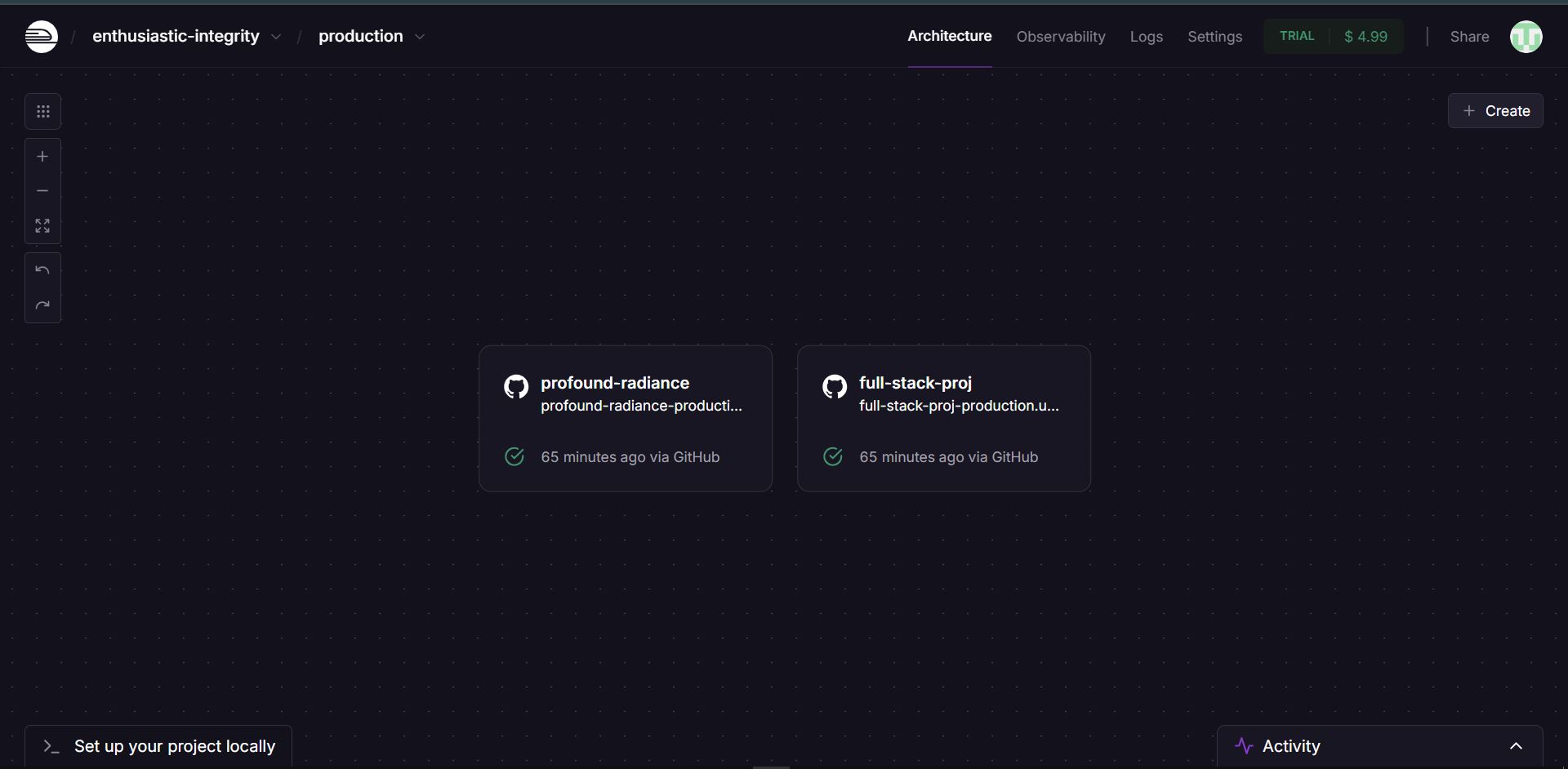
"node": ">=12.0.0"

}

}

### ***Application To-Do List en ligne***

  
Cette capture montre l’application fonctionnelle sur le lien généré par Railway :  
https://full-stack-proj-production.up.railway.app.  
On peut y ajouter, éditer ou supprimer des tâches.

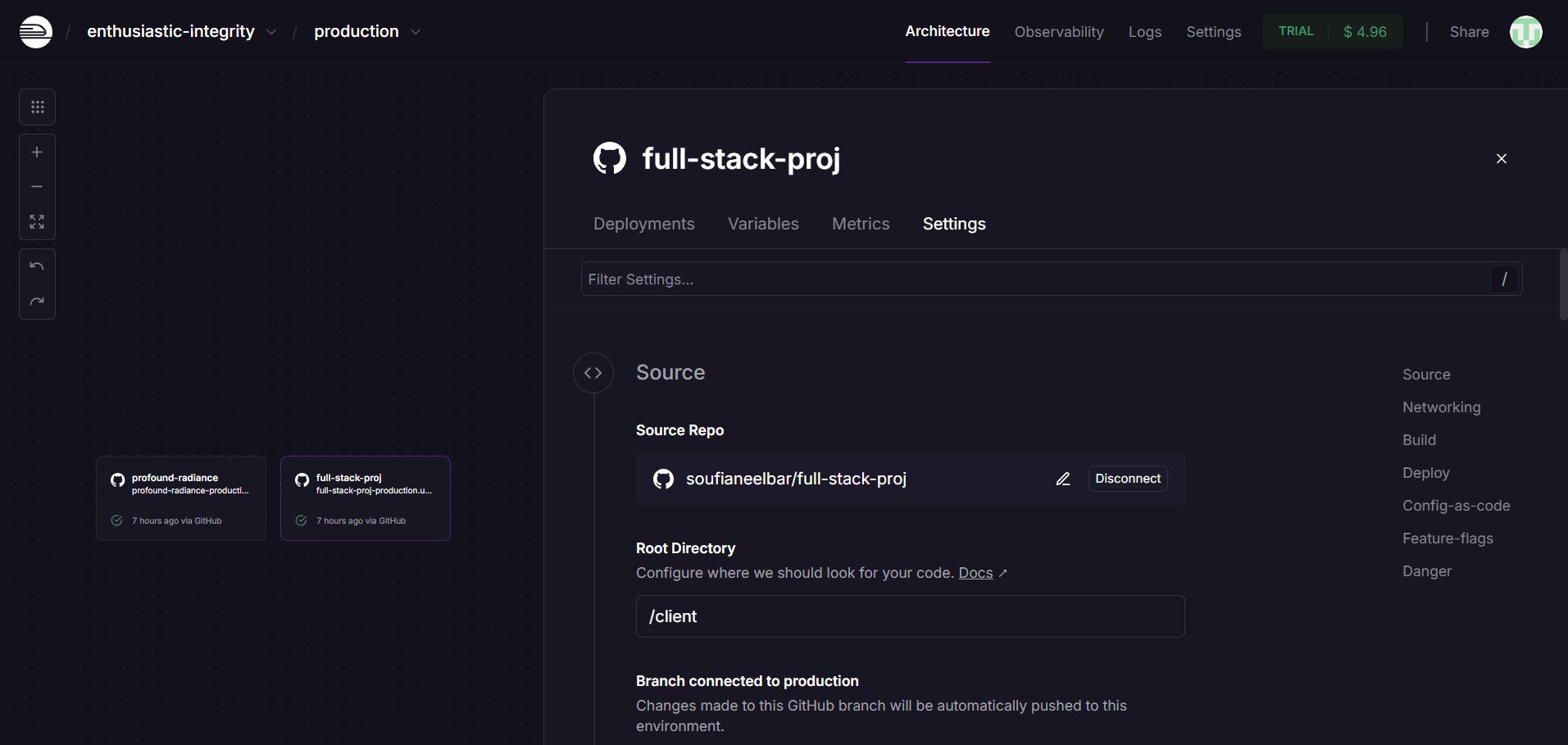


Cette capture illustre la configuration du projet déployé sur Railway :

Répertoire racine : /client (cela indique que c’est le dossier contenant le frontend React).

Repo GitHub connecté : soufianeelbar/full-stack-proj

Déploiement continu : chaque push sur GitHub déclenche automatiquement un nouveau déploiement.



deux projets : profound-radiance et full-stack-proj.  
Le projet full-stack-proj correspond à notre application To-Do List, déployée via GitHub.

### ***Conclusion***

Ce projet a permis de créer et déployer une application to-do list complète, avec un frontend en React.js et un backend en Express.js. Le déploiement sur Railway a été l'occasion de comprendre les étapes techniques essentielles d’un projet full-stack, de la configuration au débogage. Ce travail a renforcé nos compétences en développement web moderne et en gestion de projet.