Cahier des charges pour l’application web de conversion d’unités, taux de change et météo

1. Introduction

L'objectif de ce cahier des charges est de définir les fonctionnalités, les exigences et les spécifications de l'application web qui permet de convertir différentes unités, afficher les taux de change et la météo pour u ne ville donnée. L'application est développée en utilisant Angular pour le frontend et Spring Boot pour le b ackend.

2. Description des fonctionnalités

2.1. Convertisseur d'unités

L'application doit inclure un convertisseur d'unités prenant en charge les conversions de l'astronomie aux hertz et mégahertz. Les conversions doivent être réalisées côté backend à l'aide de Spring Boot et les rés ultats doivent être renvoyés au frontend Angular pour affichage sous forme de tableau.

2.2. Gestion des conversions

Les conversions effectuées par les utilisateurs doivent être enregistrées dans un fichier JSON avec un ide ntifiant unique (ID) pour chaque conversion.
Le frontend Angular doit afficher toutes les conversions dans un tableau avec les différents paramètres uti lisés pour la conversion et l'ID correspondant.
L'application doit permettre la suppression des conversions via des boutons "Supprimer" (CRUD), qui env erront l'ID associé au backend Spring Boot pour effectuer la suppression et mettre à jour le fichier JSON.

2.3. Taux de change

L'application doit permettre aux utilisateurs d'accéder à un onglet "Taux de change" qui leur permettra de saisir des valeurs dans des champs d'entrée Angular. Ces valeurs seront transmises au backend Spring Boot, qui se connectera à une API externe pour obtenir les taux de change associés. Le backend renverr a ensuite la réponse à Angular pour affichage.

2.4. Météo

L'application doit disposer d'un onglet "Météo" qui permettra aux utilisateurs de saisir le nom de la ville do nt ils souhaitent obtenir les informations météorologiques. Les données de la ville seront envoyées au bac kend Spring Boot, qui utilisera une API externe (Weather API) pour récupérer les informations météorolog iques de la ville. Le backend renverra ensuite les données à Angular pour les afficher, y compris le nom d e la ville, l'état du temps avec une image appropriée et la température.

3. Environnement de développement

Le développement du frontend (Angular) a été réalisé en utilisant Visual Studio Code (VSCode).
Le développement du backend (Spring Boot) a été réalisé en utilisant IntelliJ IDEA.

4. Exigences non fonctionnelles

L'application doit être conviviale et réactive, assurant une expérience utilisateur fluide.
Le frontend et le backend doivent être bien structurés, modulaires et facilement maintenables.
L'application doit être testée pour assurer la fiabilité des conversions, l'exactitude des taux de change et l' obtention précise des données météorologiques.
La sécurité de l'application doit être prise en compte pour protéger les données utilisateur et empêcher to ute vulnérabilité. (utilisation du cross-origin avec uniquement l'adresse du serveur web angular autorisé)