Documentation technique

Fichier : 01 - app.js

Tutoriel Node.js avec Express, CORS et Dotenv

Ce tutoriel explique comment mettre en place un serveur Node.js simple en utilisant les bibliotheques Express, CORS et Dotenv. Nous allons egalement definir des routes et des services specifiques pour une API d'intelligence artificielle (AI).

Importation des Modules

```
import express from 'express';
import cors from 'cors';
import dotenv from 'dotenv';

import aiRoutes from './routes/ai.js';
import aiServices from './config/ai-services.js';
```

Dans ce bloc de code, nous importons les modules necessaires pour notre application :

- `express` : un framework pour Node.js qui simplifie le developpement d'applications web.
- `cors` : un package Node.js pour fournir un middleware Connect/Express qui peut etre utilise pour activer CORS (Cross-Origin Resource Sharing) avec diverses options.
- `dotenv` : un module qui charge les variables d'environnement a partir d'un fichier `.env` dans `process.env`.
- `aiRoutes` : un module definissant les routes pour notre API d'AI.
- `aiServices` : un module definissant les services disponibles pour notre API d'AI.

Configuration de l'Application

```
dotenv.config();

const app = express();
const port = 3000;

app.use(cors());
app.use(express.json());
```

Ici, nous configurons notre application :

- Nous appelons `dotenv.config()` pour charger les variables d'environnement a partir d'un fichier `.env`.
- Nous initialisons une nouvelle application Express et definissons le port sur lequel notre serveur ecoutera.
- Nous utilisons le middleware `cors()` pour activer CORS sur notre serveur.
- Nous utilisons `express.json()` pour analyser les corps des requetes entrantes au format JSON.

Definition des Routes

```
app.use('/api/ai', aiRoutes);

app.get('/api/ai/services', (req, res) => {
  res.json({ services: aiServices });
});
```

Dans ce bloc, nous definissons les routes pour notre application :

- Nous utilisons `app.use()` pour monter le routeur `aiRoutes` sur le chemin `/api/ai`.
- Nous definissons une route GET a `/api/ai/services` qui renvoie la liste des services d'AI disponibles.

Demarrage du Serveur

```
app.listen(port, () => {
  console.log(`Server listening on http://localhost:${port}`);
});
```

Enfin, nous demarrons notre serveur en appelant `app.listen()`. Une fois que le serveur est pret, il affiche un message indiquant qu'il ecoute sur le port specifie.

Fichier : 02 - config_ai-services.js

Tutoriel : Comprendre le code Node.js pour les services d'IA

Dans ce tutoriel, nous allons examiner un bloc de code Node.js qui definit une liste de services d'IA. Nous ne nous concentrerons pas sur chaque ligne, mais plutot sur les concepts et les roles cles qui sont en jeu.

Le code

```
const aiServices = {
  llm: [
    { type: 'chatgpt', label: 'OpenAI', purpose: 'Text generation, summarization, Q&A, code completion' },
    { type: 'claude', label: 'Claude', purpose: 'Structured reasoning, content writing, safe dialogue' },
    //...
  ],

  tts: [
    { type: 'elevenlabs', label: 'ElevenLabs', purpose: 'High-quality voice synthesis from text, multilingual' },
  ],

  //...

};

export default aiServices;
```

Explication du code

Definition des services d'IA

Le code commence par definir une constante `aiServices`, qui est un objet JavaScript. Cet objet contient plusieurs cles, chacune representant une categorie de services d'IA. Les categories incluent `llm` (langage et modeles d'apprentissage), `tts` (text-to-speech), `avatar`, `image`, `agent` et `music`.

Chaque categorie est un tableau d'objets. Chaque objet represente un service d'IA specifique et contient trois proprietes : `type`, `label` et `purpose`.

- `type` est un identifiant unique pour le service.
- `label` est le nom du service.
- `purpose` decrit ce que fait le service.

Par exemple, dans la categorie `llm`, nous avons un service de type `chatgpt` avec le label `OpenAI` qui est utilise pour la generation de texte, la resume, les questions-reponses et la completion de code.

Exportation du module

A la fin du code, nous avons `export default aiServices;`. Cela signifie que nous exportons l'objet `aiServices` comme module par defaut. Cela permet a d'autres fichiers de code d'importer et d'utiliser les donnees definies dans `aiServices`.

Conclusion

Ce code est un exemple de comment on peut structurer et organiser des informations sur differents services d'IA dans une application Node.js. En utilisant des objets et des tableaux, nous pouvons creer une structure de donnees claire et facile a utiliser.