Détails du Commit

Commit par : latifnjimoluh

Suggestions d'Amélioration

Analyse du code :

Fichier modifié : analyzeCommit.mjs

Modifications effectuées :

- Suppression des commentaires inutiles et de certaines variables redondantes.
- Simplification du code et amélioration de la lisibilité.
- Ajout de gestion d'erreurs pour les appels à `execSync`.
- Suppression de la génération du fichier de code, car la fonctionnalité était redondante avec l'analyse de code Gemini.
- Suppression du code d'envoi d'un fichier de code en pièce jointe dans l'email.
- Correction de la regex pour extraire l'adresse email.
- Amélioration du formatage du message email.

Suggestions d'amélioration :

- **Déplacer les informations de configuration dans un fichier séparé :** Cela permettra de séparer le code et la configuration et de faciliter la maintenance.
- **Utiliser une librairie de gestion de fichiers plus robuste :** Au lieu de `fs.writeFileSync`, il serait préférable d'utiliser une librairie comme `fs-extra` qui offre des fonctionnalités plus avancées.
- **Créer des fonctions plus spécifiques pour chaque tâche :** Cela permettra de mieux organiser le code et de le rendre plus facilement testable.
- **Ajouter des tests unitaires :** Cela permettra de garantir que le code fonctionne correctement et de détecter les erreurs potentielles avant la mise en production.
- **Implémenter une meilleure gestion des erreurs :** Il est important d'afficher des messages d'erreur clairs et d'utiliser des mécanismes de récupération d'erreurs pour garantir la robustesse du code.
- **Ajouter une documentation plus complète :** Cela permettra de rendre le code plus facile à comprendre et à utiliser.
- **Définir un format de retour pour la fonction `analyzeCodeWithGemini` :** Cela permettra de garantir que les données retournées sont cohérentes et faciles à traiter.
- **Ajouter une vérification pour s'assurer que l'adresse email est valide avant de l'utiliser pour envoyer l'email :** Cela permettra d'éviter les erreurs d'envoi d'email.

Exemple de code amélioré :

```javascript
```

```
import { GoogleGenerativeAI } from "@google/generative-ai";
import PDFDocument from 'pdfkit';
import fs from 'fs';
import nodemailer from 'nodemailer';

// Charger les informations de configuration du fichier config.json
const config = require('./config.json');

// Créer une instance de Gemini
const genAI = new GoogleGenerativeAI(config.GEMINI_API_KEY);

// Fonction pour récupérer les détails du dernier commit
function getCommitDetails() {
  try {
    const commitInfo = execSync('git log -1 --pretty=format:"%H - %an <%ae> -
%s"').toString();
    const commitDetails = execSync('git log -1 --pretty=format:"Commit par :
%an\nEmail : %ae\nMessage : %s\nDate : %ad"').toString();
    return { commitInfo, commitDetails };
  } catch (error) {
    console.error('Erreur lors de la récupération des détails du commit:', error);
    return {};
  }
}

// Fonction pour récupérer les modifications du dernier commit
function getDiff() {
  try {
    return execSync('git diff HEAD~1 HEAD').toString();
  } catch (error) {
    console.error('Erreur lors de la récupération des différences de code:', error);
    return '';
  }
}

// Fonction pour récupérer les fichiers modifiés du dernier commit
function getModifiedFiles() {
  try {
    return execSync('git diff --name-only HEAD~1 HEAD').toString().trim().split('\n');
  } catch (error) {
    console.error('Erreur lors de la récupération des fichiers modifiés:', error);
    return [];
  }
}

// Fonction pour analyser le code via Gemini
```

```
async function analyzeCodeWithGemini(code) {
  const model = genAI.getGenerativeModel({ model: "gemini-1.5-flash" });
  const prompt = `Veuillez examiner le code suivant puis fournir le nom des fichiers
modifiés, la modification effectuée et suggérer des améliorations. Répond uniquement
en français:\n\n${code}`;

  try {
    const result = await model.generateContent(prompt);
    const text = result?.response?.text() || 'Aucune suggestion d\'amélioration disponible.';
    return text;
  } catch (error) {
    console.error("Erreur d'analyse avec Gemini:", error);
    return 'Erreur d\'analyse avec Gemini.';
  }
}

// Fonction pour créer un PDF avec les suggestions
function createPDF(commitDetails, suggestions, outputPath) {
  const doc = new PDFDocument();
  doc.pipe(fs.createWriteStream(outputPath));
  doc.fontSize(16).text('Détails du Commit', { underline: true });
  doc.moveDown().fontSize(12).text(commitDetails);
  doc.moveDown().fontSize(16).text('Suggestions d\'Amélioration', { underline: true });
  doc.moveDown().fontSize(12).text(suggestions);
  doc.end();
}

// Fonction pour envoyer un email avec les suggestions
async function sendEmail(userEmail, commitDetails, modifiedFiles, pdfPath) {
  const transporter = nodemailer.createTransport({
    host: config.SMTP_HOST,
    port: config.SMTP_PORT,
    secure: config.SMTP_SECURE === 'true',
    auth: { user: config.SMTP_USERNAME, pass: config.SMTP_PASSWORD },
  });

  const mailOptions = {
    from: config.SMTP_FROM,
    to: userEmail,
    subject: "Nouveau commit effectué",
    html: `
      <div style="font-family: Arial, sans-serif; background-color: #f4f4f4; padding: 20px;">
        <div style="background-color: #ffffff; padding: 20px; border-radius: 5px;">
          <h1 style="color: #333;">Nouveau Commit Effectué !</h1>
          <p>Voici les détails du dernier commit :</p>
          <pre>${commitDetails}</pre>
```

```
          <p><strong>Fichiers modifiés :</strong></p>
          <ul>${modifiedFiles.map(file => `<li>${file}</li>`).join('')}</ul>
          <p>Veuillez trouver en pièce jointe les suggestions d'amélioration.</p>
        </div>
      </div>
    `,
    attachments: [{ filename: 'suggestions_gemini.pdf', path: pdfPath }],
  };

  try {
    await transporter.sendMail(mailOptions);
    console.log(`Notification de commit envoyée à ${userEmail} !`);
  } catch (error) {
    console.error(`Erreur lors de l'envoi de l'email : ${error.message}`);
  }
}

// Fonction principale pour exécuter le script
async function main() {
  const { commitInfo, commitDetails } = getCommitDetails();
  const diff = getDiff();
  const modifiedFiles = getModifiedFiles();

  if (!commitInfo || !diff.trim()) {
    console.log('Aucune modification de code à analyser.');
    return;
  }

  const geminiSuggestions = await analyzeCodeWithGemini(diff);
  const pdfPath = 'suggestions_gemini.pdf';
  createPDF(commitDetails, geminiSuggestions, pdfPath);

  let userEmail = (commitInfo.match(/<(.*?)>/) || [])[1] || '';
  if (userEmail) {
    // Vérifier si l'adresse email est valide avant de l'envoyer
    if (userEmail.includes('@gmail.com')) {
      await sendEmail(userEmail, commitDetails, modifiedFiles, pdfPath);
    } else {
      console.error("L'adresse email n'est pas valide.");
    }
    fs.unlinkSync(pdfPath);
  }
}

main();
```