Documentation Projet-NPM

1. Introduction

Ce projet consiste à développer une application backend permettant de suivre en temps réel la position des véhicules de transport de fonds. L'application utilise Node.js pour le serveur, Express pour la création de l'API, et Socket.IO pour la communication en temps réel.

3. *Fonctionnalités Implémentées*

- ***Authentification des utilisateurs :***
  - *Création de comptes utilisateurs : Permettre aux nouveaux utilisateurs de créer un compte.*
  - *Gestion des sessions : Authentification des utilisateurs et gestion des sessions pour sécuriser l'accès aux ressources.*
  - *Protection des ressources : Assurer que seules les requêtes authentifiées peuvent accéder à certaines parties de l'API.*
- ***Suivi en temps réel des véhicules :***
  - *Simulation des positions : Générer des données de position GPS pour les véhicules.*
  - *Transmission en temps réel : Utiliser Socket.IO pour envoyer les positions des véhicules aux clients connectés en temps réel.*
  - *Stockage des données : Enregistrer les positions des véhicules dans une base de données MongoDB pour une récupération ultérieure.*
  - *Alertes d'immobilité : Envoyer des alertes si un véhicule reste immobile pendant une période définie.*
- ***Sécurité :***
  - *Validation des données : Utiliser des outils comme express-validator pour s'assurer que les données entrantes sont valides.*
  - *Gestion de l'authentification : Implémenter une authentification sécurisée, y compris le hachage et le salage des mots de passe avant de les stocker.*

4. Installation et Déploiement

Prérequis

- *Node.js : Assurez-vous d'avoir Node.js installé (version 14 ou supérieure).*
- *MongoDB : Une instance de MongoDB pour stocker les données des véhicules et des utilisateurs.*

Explication :

1. **Importation des Modules :** Les modules nécessaires sont importés, y compris Express, Socket.IO, et d'autres utilitaires comme dotenv et les fichiers de configuration propres à l'application.
2. **Configuration de l'Environnement :** Les variables d'environnement sont chargées à partir du fichier .env pour configurer les paramètres sensibles comme les URI de base de données et les secrets.
3. **Initialisation d'Express et du Serveur HTTP :** Une instance d'Express est créée pour gérer les requêtes HTTP, et un serveur HTTP est créé à partir de cette instance. Socket.IO est ensuite intégré au serveur HTTP pour ajouter la capacité de gérer les WebSockets.
4. **Connexion à la Base de Données :** La fonction connectDB est appelée pour établir une connexion à la base de données MongoDB.
5. **Middleware :** Le middleware Express est configuré pour gérer les requêtes JSON et servir des fichiers statiques.
6. **Routes :** Les routes d'authentification sont ajoutées sous le chemin /api/auth.
7. **Gestion des Connexions Socket.IO :** Les événements de connexion et de déconnexion des clients sont gérés, et des messages sont affichés dans la console pour ces événements.
8. **Simulateur de Véhicule :** Un simulateur de véhicule est initialisé et configuré pour émettre des événements de position et d'alerte à tous les clients connectés via Socket.IO.
9. **Démarrage du Serveur :** Le serveur est démarré et écoute les requêtes sur le port défini, affichant un message dans la console pour indiquer qu'il fonctionne correctement.

Voici le code :

```
import express from 'express';
import http from 'http';
import { Server } from 'socket.io';
import dotenv from 'dotenv';
import connectDB from './utils/db.js';
import authRoutes from './routes/auth.js';
import VehicleSimulator from './simulators/VehicleSimulator.js';

dotenv.config();

const app = express();
const server = http.createServer(app);
const io = new Server(server);

connectDB();

app.use(express.json());

app.use(express.static('public'));
```

```
app.use('/api/auth', authRoutes);

io.on('connection', (socket) => {
    console.log('nouveau client connectée');
    socket.on('connectée', () => {
        console.log('Client deconnectée');
    });
});

const vehicleSimulator = new VehicleSimulator();
vehicleSimulator.on('position', (data) => {
    io.emit('position', data);
});
vehicleSimulator.on('alert', (data) => {
    io.emit('alert', data);
});
vehicleSimulator.start();

const PORT = process.env.PORT || 5000;
server.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

2.Routage

1. **Importations** : Charge les modules nécessaires (express, express-validator) et les contrôleurs (register, login, logout).
2. **Routeur** : Crée une instance de routeur avec express.Router().
3. **Routes** :
    a. POST /register : Valide les champs username et password, puis appelle register.
    b. POST /login : Valide les champs username et password, puis appelle login.
    c. POST /logout : Appelle logout pour déconnecter l'utilisateur.
    d. GET /test : Renvoie une réponse textuelle simple.
4. **Exportation** : Exporte le routeur pour l'utiliser dans l'application principale

Voici le code :

```
import express from 'express';
import { check, validationResult } from 'express-validator';
import { register, login, logout } from '../controllers/authController.js';

const router = express.Router();

router.post(    '/register',
    [
        check('username', 'Username is required').not().isEmpty(),
```

```
        check('password', 'Password is required').isLength({ min: 6 })
    ],
    (req, res) => {
        const errors = validationResult(req);
        if (!errors.isEmpty()) {
            return res.status(400).json({ errors: errors.array() });
        }
        register(req, res);
    }
);

router.post(
    '/login',
    [
        check('username', 'Username is required').not().isEmpty(),
        check('password', 'Password is required').isLength({ min: 6 })
    ],
    (req, res) => {
        const errors = validationResult(req);
        if (!errors.isEmpty()) {
            return res.status(400).json({ errors: errors.array() });
        }
        login(req, res);
    }
);

router.post('/logout', logout);

router.get("/test", function(req, res) {
    res.send("je renvoie quelques chose")
})

export default router;
```