// Importez le modèle "Model" depuis le fichier "../models/Player.js"

import Model from "../models/Player.js";

// Exportez une fonction asynchrone "signup" qui gère une demande d'inscription

export const signup = async (req, res, next) => {

try {

// Attendez la création d'un nouvel enregistrement en utilisant les données de la requête (req.body)

await Model.create({ ...req.body });

// Répondez avec un statut HTTP 201 (Created) et un message JSON "Hello"

res.status(201).json("Hello");

} catch (error) {

// En cas d'erreur, passez l'erreur au middleware suivant pour la gestion des erreurs

next(error);

}

};

// Exportez une fonction asynchrone "allPlayers" qui récupère tous les joueurs

export const allPlayers = async (req, res, next) => {

try {

// Attendez la recherche de tous les joueurs dans la base de données

const players = await Model.find();

// Répondez avec un statut HTTP 200 (OK) et renvoyez la liste des joueurs au format JSON

res.status(200).json(players);

} catch (error) {

// En cas d'erreur, passez l'erreur au middleware suivant pour la gestion des erreurs

next(error);

}

};

// Exportez une fonction asynchrone "deletePlayer" qui supprime un joueur en fonction de son ID

export const deletePlayer = async (req, res, next) => {

try {

// Attendez de trouver un joueur en fonction de l'ID passé dans les paramètres de la requête (req.params.id)

const player = await Model.findById(req.params.id);

// Si le joueur n'est pas trouvé, répondez avec un statut HTTP 404 (Non trouvé) et un message JSON "Player not found."

if (!player) return res.status(404).json("Player not found.");

// Supprimez le joueur en fonction de son ID

await Model.findByIdAndRemove(id);

// Répondez avec un statut HTTP 200 (OK) et un message JSON "The hero has been deleted."

res.status(200).json("The hero has been deleted.");

} catch (error) {

// En cas d'erreur, passez l'erreur au middleware suivant pour la gestion des erreurs

next(error);

}

}

// Exportez une fonction asynchrone "putPlayer" qui met à jour un joueur en fonction de son ID

export const putPlayer = async (req, res, next) => {

try {

// Attendez de trouver un joueur en fonction de l'ID passé dans les paramètres de la requête (req.params.id)

const player = await Model.findById(req.params.id);

// Si le joueur n'est pas trouvé, répondez avec un statut HTTP 404 (Non trouvé) et un message JSON "player not found !"

if (!player) return res.status(404).json("player not found !");

// Mettez à jour le joueur en fonction de son ID avec les données de la requête (req.body)

const updatePlayer = await Model.findByIdAndUpdate(

req.params.id,

{ $set: req.body },

{ new: true } // Renvoie la version mise à jour du joueur

);

// Répondez avec un statut HTTP 200 (OK) et un message JSON contenant un message et le joueur mis à jour

res.status(200).json({

message: "player updated",

updatePlayer

});

} catch (error) {

// En cas d'erreur, passez l'erreur au middleware suivant pour la gestion des erreurs

next(error);

}

}

// Importe le module "express" et l'assigne à la variable "express"

import express from "express";

// Importe l'objet "env" depuis le fichier "./config/index.js" et l'assigne à la variable "env"

import { env } from "./config/index.js";

// Importe le module "mongoose" et l'assigne à la variable "mongoose"

import mongoose from "mongoose";

// Crée une instance de l'application Express et l'assigne à la variable "app"

const app = express();

// Définit la variable "PORT" qui prend la valeur de la variable d'environnement "PORT" si elle est définie, sinon elle prend la valeur 8080 par défaut

const PORT = process.env.PORT || 8080;

// Établit une connexion à la base de données MongoDB en utilisant l'URL spécifiée dans "env.mongoURL"

mongoose

.connect(env.mongoURL)

.then(() => console.log("Connexion à MongoDB réussie !")) // Affiche un message si la connexion à MongoDB réussit

.catch((error) => console.log(error)); // Affiche une erreur si la connexion à MongoDB échoue

// Utilise le middleware d'Express pour traiter les requêtes JSON

app.use(express.json());

// Démarre le serveur Express et écoute les connexions entrantes sur le port spécifié par "PORT"

app.listen(PORT, () => {

console.log(`Listening at http://localhost:${PORT}`); // Affiche un message indiquant que le serveur écoute sur un port donné

});