const validateUser = (

req: Request<{}, {}, ICreateUserDto>,

res: Response,

next: NextFunction

) => {

const { name, email, birthdate, nDni, username, password } = req.body;

try {

// Validación de name:

if (!name) throw new Error("El campo name es requerido");

if (name.length < 4) throw new Error("El campo name debe tener al menos 4 caracteres");

if (name.length > 50) throw new Error("El campo name debe tener como máximo 50 caracteres");

// Validación de email:

if (!email) throw new Error("El campo email es requerido");

const emailRegex = /^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$/;

if (!emailRegex.test(email)) throw new Error("El campo email debe ser un email válido");

// Validación de birthdate:

if (!birthdate) throw new Error("El campo birthdate es requerido");

const dateRegex = /^\d{4}-\d{2}-\d{2}$/;

if (!dateRegex.test(birthdate)) throw new Error("El campo birthdate debe estar en formato yyyy-mm-dd");

const today = new Date();

today.setHours(0, 0, 0, 0);

const birthdateDate = new Date(birthdate);

const ageDiff = new Date(today.getTime() - birthdateDate.getTime());

const age = ageDiff.getUTCFullYear() - 1970;

if (age < 16) throw new Error("La persona debe tener al menos 16 años");

// Validación del número de DNI:

if (!nDni) throw new Error("El campo nDni es requerido");

if (typeof nDni !== "number") throw new Error("El campo nDni debe ser un número");

if (nDni < 0) throw new Error("El campo nDni debe ser un número");

// Validación de username:

if (!username) throw new Error("El campo username es requerido");

if (username.length < 4) throw new Error("El campo username debe tener al menos 4 caracteres");

if (username.length > 28) throw new Error("El campo username debe tener como máximo 28 caracteres");

// Validación de password:

if (!password) throw new Error("El campo password es requerido");

if (password.length < 4) throw new Error("El campo password debe tener al menos 4 caracteres");

if (password.length > 18) throw new Error("El campo password debe tener no mas de 18 caracteres");

const passwordRegex = /^(?=.\*[a-z])(?=.\*\d)(?=.\*[@$!%\*?&#])[A-Za-z\d@$!%\*?&#]+$/;

if (!passwordRegex.test(password))

throw new Error(

"El password debe tener al menos una letra, un número y un caracter especial: @$!%\*?&#"

);

} catch (error: any) {

// if (error instanceof Error) {

return res.status(400).json({ error: error.message });

// }

// return res.status(400).json({ error: "El campo name es requerido" });

}

};

import { Request, Response, NextFunction } from "express";

import ICreateCredentialDto from "../dtos/ICreateCredentialDto";

const validateCredential = (

req: Request<{}, {}, ICreateCredentialDto>,

res: Response,

next: NextFunction

) => {

const { username, password } = req.body;

try {

if (!username) throw new Error("El campo username es requerido");

if (!password) throw new Error("El campo password es requerido");

} catch (error: any) {

if (error instanceof Error) {

return res.status(400).json({ error: error.message });

}

}

next();

};

import { Request, Response, NextFunction } from "express";

import IScheduleAppointmentDto from "../dtos/IScheduleAppointmentDto";

const validateAppointment = (

req: Request<{}, {}, IScheduleAppointmentDto>,

res: Response,

next: NextFunction

) => {

const { date, time, description } = req.body;

try {

// Validación de la fecha

if (!date) throw new Error("El campo date es requerido");

const appointmentDate = new Date(date);

const today = new Date();

today.setHours(0, 0, 0, 0);

const tomorrow = new Date(today);

tomorrow.setDate(today.getDate() + 1);

const in14Days = new Date(today);

in14Days.setDate(today.getDate() + 14);

if (appointmentDate < tomorrow || appointmentDate > in14Days)

throw new Error("La fecha debe estar entre mañana y dentro de 14 días");

// Validación del tiempo

if (!time) throw new Error("El campo time es requerido");

const validTimes = [

"09:00",

"09:30",

"10:00",

"10:30",

"11:00",

"11:30",

"12:00",

"12:30",

"13:00",

"13:30",

"14:00",

"14:30",

"15:00",

"15:30",

"16:00",

"16:30",

"17:00",

"17:30",

];

if (!validTimes.includes(time)) {

throw new Error("El campo time debe estar entre las 9:00 y 17:30 en intervalos de 30 minutos");

}

// Validación de la descripción

if (!description) throw new Error("El campo description es requerido");

if (typeof description !== "string") throw new Error("El campo description debe ser un string");

if (description.length < 4 || description.length > 50)

throw new Error("El campo description debe tener entre 4 y 50 caracteres");

} catch (error: any) {

// if (error instanceof Error) {

return res.status(400).json({ error: error.message });

// }

}

next();

};



