API REST: Manejador de movimientos y emociónes del robot Eva.
Autor: José Armando Menéndez Clavijo.
Curso: Sistemas Distribuidos.

Los escenarios de uso de la API REST serían principalmente en:
1. Que el robot atienda al lugar desde donde se esta emitiendo un sonido.
2. Realizar un emoción como decir que si o que no.
3. Definición de acciones o emociones en el futuro tales como avisar de un evento determinado.

Introducción:
Se propone la implementación de una API REST, que posibilite el control del robot Eva. Este se encuentra montado en una plataforma robótica con grados de movilidad 90 hacia adelante, 90 hacia atrás y 90 a la izquierda y 90 a la derecha. Va a permitir el manejo remoto del robot así como mantener un historial de todas las interacciones y respuestas del mismo para su uso en investigaciones médicas. El API va a ser desplegado en un Raspberry el cual va a ser el encargado de administrar las acciones de la plataforma robótica.

URI y Estructura del JSON que va a regresar mi API REST

Recursos	POST	GET	PUT	DELETE
/robot/actions	Error.	Devuelve el historial completo de las acciones del robot dado un rango de fecha.	Error.	Ejecuta un borrado lógico de todo el historial de acciones dado un rango de fecha.
/robot/actions/types	Adiciona una acción nueva al robot.	Devuelve las acciones que puede hacer el robot.	Error.	Error.
/robot/actions/types/:id	Realiza una acción atómica con el robot.	Devuelve un tipo de accion del robot.	Actualiza los datos de una acción del robot.	Ejecuta un borrado lógico de la acción del robot.
/robot/emotions	Error.	Devuelve el historial completo de las emociones del robot dado un rango de fecha.	Error.	Ejecuta un borrado lógico de todo el historial de emociones dado un rango de fecha.
/robot/emotions/:id	Error.	Devuelve la emoción realizada por el robot.	Actualiza los datos de la emoción relizada por el robot.	Ejecuta un borrado lógico de la emoción realizada por el robot.
/robot/emotions/types	Adiciona una emoción nueva al	Devuelve el historial completo de las emociones del robot dado un	Error.	Ejecuta un borrado lógico de todo el historial de emociones dado un

	robot.	rango de fecha.		rango de fecha.
/robot/emotions/types/:id	Ejecuta una emoción con el robot.	Devuelve el tipo de emoción del robot.	Actualiza los datos de una emoción del robot.	Ejecuta un borrado lógico de la emoción del robot.
/robot/dialogs	Ejecuta la reproducción de audio de un texto de entrada.	Devuelve los frases que dijo el robot.	Error.	Ejecuta un borrado lógico de las frases dichas por el robot en un rango de fecha.
/robot/dialogs/:id	Error.	Devuelve una frase que haya dicho el robot.	Actualiza los datos de una frase dicha por el robot.	Ejecuta un borrado lógico de una frase dicha por el robot.
/robot/listenings	Ejecuta la recopilación de un audio externo.	Devuelve los audios recopilados.	Error.	Ejecuta un borrado lógico de los audios recopilados en un rango de fecha.
/robot/listenings/:id	Error.	Devuelve un audio recopilado por el robot.	Actualiza los datos interpretados del audio recopilado por el robot.	Ejecuta un borrado lógico de los datos de un audio recopilado por el robot.

Estructuras JSON

```
var emotionLogSchema = new Schema({
	begin: { type: Date, default: Date.now },
	emotion: {type: Schema.Types.ObjectId, ref: 'Emotion'},
	end : { type: Date, default: Date.now },
	active: {type: Boolean, default: true}
});

var actionLogSchema = new Schema({
	begin: { type: Date, default: Date.now },
	action: {type: Schema.Types.ObjectId, ref: 'Action'},
	end : { type: Date, default: Date.now },
	active: {type: Boolean, default: true}
});

var speechSchema = new Schema({
	begin: { type: Date, default: Date.now },
	text: {type: String},
	end : { type: Date, default: Date.now },
	active: {type: Boolean, default: true}
});

var listeningSchema = new Schema({
	begin: { type: Date, default: Date.now },
	frase : {type: String},
	audio: {type: String},
	end : { type: Date, default: Date.now },
	active: {type: Boolean, default: true}
});

var emotionSchema = new Schema({
	name: {type: String},
	actions: [{type: Schema.Types.ObjectId, ref: 'Action'}],
	command : {type: String},
	active: {type: Boolean},
	active: {type: Boolean, default: true}
});

var actionSchema = new Schema({
	name: {type: String},
	emotions: [{type: Schema.Types.ObjectId, ref: 'Emotion'}],
	command : {type: String},
	active: {type: Boolean, default: true}
});
```