1) Escribe una función **esMayorEdad** que reciba una edad como argumento y devuelva una promesa que compruebe si la edad es mayor de edad y en ese caso tome el valor true.

```
function esMayorEdad(edad) {
    return new Promise((resolve, reject) => {
        if (edad >= 18)
            resolve(true);
        else
            reject(false);
    })
}


// PRUEBA DE LLAMADA-usando fórmula 1 de escribir el then y catch
let edad1=19;
esMayorEdad(edad1)
    .then(resultado => console.log(edad1 + " SÍ ES MAYOR EDAD"))
    .catch(resultado => console.log(edad1+" NO ES MAYOR EDAD"));

let edad2=12;
esMayorEdad(edad2)
    .then(resultado=> console.log(edad2 + " SÍ ES MAYOR EDAD"))
    .catch(resultado => console.log(edad2+" NO ES MAYOR EDAD"));


// PRUEBA DE LLAMADA-usando fórmula 2 de escribir el then y catch
// Cuando no voy a usar el resultado que me llega al then o al catch,
// como en este caso (no uso el true, ni el false), puedo cambiarlo por
// () indicando que no hay resultado que me interese.
edad1=19;
esMayorEdad(edad1)
    .then(() => console.log(edad1 + " SÍ ES MAYOR EDAD"))
    .catch(() => console.log(edad1+" NO ES MAYOR EDAD"));

edad2=12;
esMayorEdad(edad2)
    .then(()=> console.log(edad2 + " SÍ ES MAYOR EDAD"))
    .catch(() => console.log(edad2+" NO ES MAYOR EDAD"));
```

2) Escribe una función **arrayStringsToMayusculas** que recibe un array de palabras como argumento y pasa todas las palabras del array a mayúsculas. Usa promesas para hacerlo:

SOLUCIÓN 1: programando sin usar métodos de arrays propios de javascript:

```
function arrayStringsToMayusculas(array) {
    return new Promise((resolve, reject) => {
        if (!array) reject(new Error("No hay array"));

        for (var i = 0; i < array.length; i++) {
            if (typeof array[i] !== 'string')
                reject(new Error("No es un array de strings"));

            array[i] = array[i].toUpperCase();
        }
        resolve(array);
    });
}


// PRUEBA DE LLAMADA
const miArray = ['casa', 'coche', 'agua'];
const miArrayMalo = ['casa', 23, 'coche'];
//prueba 1
arrayStringsToMayusculas(miArray)
    .then(array => {
        console.log(array);
    })
    .catch(error => console.log(error.message));

//prueba 2
arrayStringsToMayusculas(miArrayMalo)
    .then(array => {
        console.log(array);
    })
    .catch(error => console.log(error.message));

//prueba 3
arrayStringsToMayusculas()
    .then(array => {
        console.log(array);
    })
    .catch(error => console.log(error.message));
```

SOLUCIÓN 2: programando usando el método “map” que tiene javascript para los arrays y que permite aplicar una función de transformación a cada elemento del array:

La función “arrayStringsToMayusculas(array) es la que cambia, el resto sería igual:

```
function arrayStringsToMayusculas(array) {
    return new Promise((resolve, reject) => {
        if (!array) reject(new Error("No hay array"));
```

```
        let arrayTransformado =
            array.map(palabra => {
                if (typeof palabra !== 'string')
                    reject(new Error("No es un array de strings"));
                return palabra.toUpperCase();
            });

        resolve(arrayTransformado);

    });
}
```

3) Escribe una función **ordenarPalabrasFromArray** que recibe un array de palabras como argumento y ordena las palabras alfabéticamente. Controlar el caso de que no se pase ningún array como error y si el array no es de "string" también sería error. Usa promesas para hacerlo:

SOLUCIÓN 1: programando sin usar métodos de arrays propios de javascript:

```
function ordenarPalabrasFromArray(array) {
    return new Promise((resolve, reject) => {
        if (!array) reject(new Error("No hay array"));

        for (var i = 0; i < array.length; i++) {
            if (typeof array[i] !== 'string')
                reject(new Error("No es un array de strings"));

        }
        resolve(array.sort());
    });
}


// PRUEBA DE LLAMADA
const miArray = ['premio', 'coche', 'agua'];
const miArrayMalo = ['premio', 23, 'coche'];
//prueba 1
ordenarPalabrasFromArray(miArray)
    .then(array => {
        console.log(array);
    })
    .catch(error => console.log(error.message));

//prueba 2
ordenarPalabrasFromArray(miArrayMalo)
    .then(array => {
```

```
        console.log(array);
    })
    .catch(error => console.log(error.message));

//prueba 3
ordenarPalabrasFromArray()
    .then(array => {
        console.log(array);
    })
    .catch(error => console.log(error.message));
```

SOLUCIÓN 2: programando usando el método "foreach" que tiene javascript para los arrays y que permite aplicar una función **(no de transformación**) que se aplica a cada elemento del array:

La función "ordenarPalabrasFromArray(array) es la que cambia, el resto sería igual:

```
function ordenarPalabrasFromArray(array) {
    return new Promise((resolve, reject) => {
        if (!array) reject(new Error("No hay array"));

        array.forEach(palabra => {
            if (typeof palabra !== 'string')
                reject(new Error("No es un array de strings"));

        });

        resolve(array.sort());
    });
}
```

4) Reutilizando las funciones hechas en los ejercicios 2 y 3 y suponiendo que devuelven promesas (sino, no podrás hacer este ejercicio), usa promesas encadenadas para conseguir que a partir de un array cualquiera de strings, obtengas al final el array en mayúsculas y ordenado alfabéticamente. Si se le pasa un array incorrecto, informará de ello y si no se le pasa también informará de ello.

Ejemplo de posible llamada:

const arrayEntrada=['piso', 'estrategia', 'libro', 'papel'];

Resultado en pantalla, el array: ['ESTRATEGIA', 'LIBRO', 'PAPEL', 'PISO']

SOLUCIÓN:

```
function ordenarPalabrasFromArray(array) {
    return new Promise((resolve, reject) => {
        if (!array) reject(new Error("No hay array"));

        array.forEach(palabra => {
            if (typeof palabra !== 'string')
                reject(new Error("No es un array de strings"));

        });

        resolve(array.sort());
    });
}


function arrayStringsToMayusculas(array) {
    return new Promise((resolve, reject) => {
        if (!array) reject(new Error("No hay array"));

        let arrayTransformado =
            array.map(palabra => {
                if (typeof palabra !== 'string')
                    reject(new Error("No es un array de strings"));
                return palabra.toUpperCase();
            });

        resolve(arrayTransformado);

    });
}




// PRUEBA DE LLAMADA
const miArray = ['premio', 'coche', 'agua'];
const miArrayMalo = ['premio', 23, 'coche'];

//prueba 1
arrayStringsToMayusculas(miArray)
    .then(ordenarPalabrasFromArray)
    .then(resultado=>console.log("Resultado de prueba 1: "+ resultado))
    .catch(error => console.log("Resultado de prueba 1: "+ error.message));

//prueba 2
arrayStringsToMayusculas(miArrayMalo)
    .then(ordenarPalabrasFromArray)
```

```
    .then(resultado=>console.log("Resultado de prueba 2: "+resultado))
    .catch(error => console.log("Resultado de prueba 2: "+ error.message));

//prueba 3
arrayStringsToMayusculas()
    .then(ordenarPalabrasFromArray)
    .then(resultado=>console.log("Resultado de prueba 3: "+resultado))
    .catch(error => console.log("Resultado de prueba 3: "+ error.message));

    /* IMPORTANTE: fijaros que en el primer then, sólo le pasa el nombre de la
función "ordenarPalabrasFromArray",
       no se hace:

                .then(array => ordenarPalabrasFromArray(array))

       Eso funciona porque al pasar el nombre de una función a un "then", lo
que hace es
       aplicar esa función al resultado que reciba el then, es decir, en este
caso,
       aplicaría "ordenarPalabrasFromArray" al array que nos devuelva
"arraStringsToMayusculas".
       Y eso era lo que queríamos. Por tanto, es una forma abreviada de
escribir el then.
       La forma larga, mostrada justo después de IMPORTANTE, también funciona
siempre, pero en estos
       casos en que sólo queremos aplicarle otra función, no se suele usar.
       */
```

5) ¿Qué salida muestra el siguiente código?:

```
let promise = new Promise(function(resolve, reject) {
  resolve(1);

  setTimeout(() => resolve(2), 1000);
});

promise.then(alert);
```

SOLUCIÓN.
Muestra 1 porque el segundo resolve() no tiene efecto, lo ignora, ya que las promesas una vez resueltas se terminan.

6) Crear una api para el manejo de un array de objetos de tipo persona, entendidos como objetos persona json con los atributos: nombre, edad, salario y edad. Un ejemplo de array personas podría ser:

```
var personas = [
    {
        nombre: 'Ana',
        edad: 30,
        salario:2000,
        id: 1
    },
   {
        nombre: 'Bea',
        edad: 18,
        salario:1500,
        id: 2
    },
    {
        nombre: 'Carlos',
        edad: 15,
        salario:1000,
        id: 3
    },
    {
        nombre: 'Daniel',
        edad: 40,
        salario:1800,
        id: 4
    }
];
```

La api tiene que tener métodos para:

- Obtener una persona a partir de su id
- Obtener la persona con mayor edad
- Obtener la media del salario de todas las persona
- Añadir una persona
- Modificar una persona a partir de su id
- Borrar una persona a partir de su id

La api será un objeto como:

var api={
getPersona://definir una función que obtenga una persona a partir de su id,
getPersonaConMayorEdad://definir función
…}

Las llamadas a las funciones serán del tipo:

var miPersona=getPersona(id,…..);

REALIZAR EL EJERCICIO CUMPLIMENTO LOS SIGUIENTES REQUISITOS:

a) Implementar todos los métodos de la api usando **CALLBACKS**
b) Implementar todos los métodos de la api usando **PROMESAS**

SOLUCIÓN BÁSICA: sin usar la potencia de los arrays de Javascript ni sus métodos:

```
var personas = [
    {
        nombre: 'Ana',
        edad: 30,
        salario: 2000,
        id: 1
    }, {
        nombre: 'Bea',
        edad: 18,
        salario: 1500,
        id: 2
    },
    {
        nombre: 'Carlos',
        edad: 15,
        salario: 1000,
        id: 3
    },
    {
        nombre: 'Daniel',
        edad: 40,
        salario: 1800,
        id: 4
    }
];


var api = {
    getPersona: function (id, callback) {
        for (var i = 0; i < personas.length; i++) {
            if (personas[i].id == id) {
                return callback(null, personas[i]);
            }
        }
        callback(new Error("no existe"), null);
    },

    getPersonaPromesa: function (id) {
        return new Promise((resolve, reject) => {
            for (var i = 0; i < personas.length; i++) {
                if (personas[i].id == id) {
```

```
                resolve(personas[i]);
            }
        }
        reject(new Error("no existe"));
    });
},

getPersonaConMayorEdad: function (callback) {
    var max = 0;
    var index = -1;
    for (var i = 0; i < personas.length; i++) {
        if (personas[i].edad > max) {
            max = personas[i].edad;
            index = i;

        }
    }
    if (index != -1)
        callback(null, personas[index]);
    else
        callback(new Error("no existe ninguna persona"), null);
},

getPersonaConMayorEdadPromesa: function () {
    return new Promise((resolve, reject) => {
        var max = 0;
        var index = -1;
        for (var i = 0; i < personas.length; i++) {
            if (personas[i].edad > max) {
                max = personas[i].edad;
                index = i;

            }
        }
        if (index != -1)
            resolve(personas[index]);
        else
            reject(new Error("no existe ninguna persona"), null);
    })
},


getMediaSalario: function (callback) {
    var suma = 0;

    if (personas.length === 0)
        callback(new Error("No hay personas en el array"), null);
    for (var i = 0; i < personas.length; i++) {
        suma += personas[i].salario;
```

```
        }
        callback(null, suma / personas.length);
    },

    getMediaSalarioPromesa: function () {
        return new Promise((resolve, reject) => {
            if (personas.length === 0)
                reject(new Error("No hay personas en el array"));
            var suma = 0;
            for (var i = 0; i < personas.length; i++) {
                suma += personas[i].salario;
            }
            resolve(suma / personas.length);
        });
    },

    //addPersona recibe un objeto persona y devuelve su id o -1 si hubo error
    addPersona: function ({ nombre, edad, salario }, callback) {
        let persona = { nombre, edad, salario };
        persona.id = personas[personas.length - 1].id + 1;
        personas.push(persona);
        callback(null, persona.id);
    },

    addPersonaPromesa: function ({ nombre, edad, salario }) {
        return new Promise((resolve, reject) => {
            let persona = { nombre, edad, salario };
            persona.id = personas[personas.length - 1].id + 1;
            personas.push(persona);
            resolve(persona.id);
        });
    },

    //devuelve la persona con los nuevos datos
    updatePersona: function (id, { nombre, edad, salario }, callback) {
        for (var i = 0; i < personas.length; i++) {
            if (personas[i].id === id) {
                if (nombre)
                    personas[i].nombre = nombre;
                if (edad)
                    personas[i].edad = edad;
                if (salario)
                    personas[i].salario = salario;
                callback(null, personas[i]);
                return;
            }
        }
        callback(new Error("No se encuentra ese id"), null);
    },
```

```
    updatePersonaPromesa: function (id, { nombre, edad, salario }) {
        return new Promise((resolve, reject) => {
            for (var i = 0; i < personas.length; i++) {
                if (personas[i].id === id) {
                    if (nombre)
                        personas[i].nombre = nombre;
                    if (edad)
                        personas[i].edad = edad;
                    if (salario)
                        personas[i].salario = salario;
                    resolve(personas[i]);
                }
            }
            reject(new Error("No se encuentra ese id"));
        });
    },

    //devuelve un 1 si todo fue ok
    borrarPersona: function (id, callback) {
        var index = -1;
        for (var i = 0; i < personas.length; i++) {
            if (personas[i].id === id) {
                index = i;
                break; //salgo del for
            }
        }
        if (index === -1)
            callback(new Error("La persona con id " + id + " no existe"),
null);
        personas.splice(index, 1);
        callback(null, 1);
    },

    borrarPersonaPromesa: function (id) {
        return new Promise((resolve, reject) => {
            var index = -1;
            for (var i = 0; i < personas.length; i++) {
                if (personas[i].id === id) {
                    index = i;
                    break; //salgo del for
                }
            }
            if (index === -1)
                reject(new Error("La persona con id " + id + " no existe"));
            personas.splice(index, 1);
            resolve(1);
        })
    }
```

```
};



//PROBANDO NUESTRA API

api.getPersona(3, function (error, persona) {
    if (error)
        console.log("ERROR en 'Persona con id=3, usando callaback': " +
error.message);
    else
        console.log("Persona con id=3, usando callaback: " +
JSON.stringify(persona));
});

api.getPersonaPromesa(1)
    .then(persona => console.log("Persona con id=1, usando promesas: " +
JSON.stringify(persona)))
    .catch(error => console.log("Persona con id=1, usando promesas: " +
error.message));


api.getPersonaConMayorEdad((error, persona) => {
    if (persona)
        console.log("Persona con mayor edad es: " + JSON.stringify(persona));
    else
        console.log("ERROR en persona con mayor edad: " + error.message);

});

api.getPersonaConMayorEdadPromesa()
    .then(persona => console.log("Persona con mayor edad, usando promesas es:
" + JSON.stringify(persona)))
    .catch(error => console.log("ERROR en persona con mayor edad, usando
promesas: " + error.message));


api.getMediaSalario((error, media) => {
    if (error)
        console.log("Error en la media de salario: " + error.message);
    else
        console.log("La media del salario de todos es: " + media);
});

api.getMediaSalarioPromesa()
    .then(media => console.log("La media del salario de todos, usando promesas
es: " + media))
```

```
    .catch(error => console.log("Error en la media de salario, usando
promesas: " + error.message));

api.addPersona({ nombre: 'Estela', edad: 24, salario: 1300 }, (error, idNuevo)
=> {
    if (error)
        console.log("Error insertando la persona: " + error.message);
    else {
        console.log("Nueva persona insertada con id: " + idNuevo);
        console.log(personas);
    }
});

api.addPersonaPromesa({ nombre: 'Fernando', edad: 50, salario: 900 })
    .then(idNuevo => {
        console.log("Nueva persona insertada, usando promesas, con id: " +
idNuevo);
        console.log(personas);
    })
    .catch(error => console.log("Error insertando la persona, usando promesas:
" + error.message));

api.updatePersona(3, { nombre: '', edad: 33, salario: 2000 }, (error, persona)
=> {
    if (error)
        console.log("Error actualizando esa persona: " + error.message);
    else {
        console.log("Actualizando persona.Los datos de la nueva persona son: "
+ JSON.stringify(persona));
        console.log(personas);
    }
});

api.updatePersonaPromesa(4, { nombre: '', edad: undefined, salario: 2800 })
    .then(persona => console.log("Actualizando persona con promesas. Los datos
de la nueva persona son: " + JSON.stringify(persona)))
    .catch(error => console.log("Error actualizando esa persona con promesas:
" + error.message));


api.borrarPersona(3, (error, resultado) => {
    if (error)
        console.log("Error borrando la persona " + error.message);
    else
        console.log("Persona borrada!");
});
api.borrarPersonaPromesa(2)
    .then(() => console.log("Persona borrada usando promesas"))
```

```
    .catch(error => console.log("Error borrando persona con promesas " +
error.message));
```

SOLUCIÓN ÓPTIMA: usando la potencia de los arrays de Javascript y sus métodos:
La API es lo que cambia y sería:

```
var api = {
    getPersona: function (id, callback) {
        var persona = personas.find(p => id === p.id);

        if (persona)
            callback(null, persona);
        else
            callback(new Error("no existe"), null);
    },

    getPersonaPromesa: function (id) {
        return new Promise((resolve, reject) => {
            var persona = personas.find(p => id === p.id);
            if (persona)
                resolve(persona);

            reject(new Error("no existe"));
        });
    },

    getPersonaConMayorEdad: function (callback) {
        var maxEdad = Math.max(...personas.map(persona => persona.edad));
        var persona = personas.find(persona => persona.edad === maxEdad);
        if (persona)
            callback(null, persona);
        else
            callback(new Error("no existe ninguna persona"), null);
    },

    getPersonaConMayorEdadPromesa: function () {
        return new Promise((resolve, reject) => {
            var maxEdad = Math.max(...personas.map(persona => persona.edad));
            var persona = personas.find(persona => persona.edad === maxEdad);
            if (persona)
                resolve(persona);
            reject(new Error("no existe ninguna persona"), null);
        })
    },

    getMediaSalario: function (callback) {
        if (personas.length === 0)
```

```
            callback(new Error("No hay personas en el array"), null);
        else {
            var media = personas.map(persona =>
persona.salario).reduce((previo, actual) => previo + actual) /
personas.length;
            callback(null, media);
        }
    },

    getMediaSalarioPromesa: function () {
        return new Promise((resolve, reject) => {
            if (personas.length === 0)
                reject(new Error("No hay personas en el array"));

            var media = personas.map(persona =>
persona.salario).reduce((previo, actual) => previo + actual) /
personas.length;
            resolve(media);
        });
    },

    //addPersona recibe un objeto persona y devuelve su id o -1 si hubo error
    addPersona: function ({ nombre, edad, salario }, callback) {
        let persona = { nombre, edad, salario };
        persona.id = personas[personas.length - 1].id + 1;
        personas.push(persona);
        callback(null, persona.id);
    },

    addPersonaPromesa: function ({ nombre, edad, salario }) {
        return new Promise((resolve, reject) => {
            let persona = { nombre, edad, salario };
            persona.id = personas[personas.length - 1].id + 1;
            personas.push(persona);
            resolve(persona.id);
        });
    },

    //devuelve la persona con los nuevos datos
    updatePersona: function (id, { nombre, edad, salario }, callback) {
        var index = personas.findIndex(persona => persona.id === id);
        if (index === -1)
            callback(new Error("No se encuentra ese id"), null);
        else {
            if (nombre)
                personas[index].nombre = nombre;
            if (edad)
                personas[index].edad = edad;
            if (salario)
```

```
                personas[index].salario = salario;
            callback(null, personas[index]);
        }
    },

    updatePersonaPromesa: function (id, { nombre, edad, salario }) {
        return new Promise((resolve, reject) => {
            var index = personas.findIndex(persona => persona.id === id);
            if (index === -1)
                reject(new Error("No se encuentra ese id"));

            if (nombre)
                personas[index].nombre = nombre;
            if (edad)
                personas[index].edad = edad;
            if (salario)
                personas[index].salario = salario;
            resolve(personas[index]);
        });

    },

    //devuelve un 1 si todo fue ok
    borrarPersona: function (id, callback) {
        var index = personas.findIndex(persona => persona.id === id);
        if (index === -1)
            callback(new Error("La persona con id " + id + " no existe"),
null);
        personas.splice(index, 1);
        callback(null, 1);
    },

    borrarPersonaPromesa: function (id) {
        return new Promise((resolve, reject) => {
            var index = personas.findIndex(persona => persona.id === id);
            if (index === -1)
                reject(new Error("La persona con id " + id + " no existe"));
            personas.splice(index, 1);
            resolve(1);
        })
    }
};
```