Ejercicio 06 - API REST Completa

Los eldars (una raza evolucionada de los elfos) viven en una nave planetaria muy grande llamada Craftworlds. Los jerarquía de los eldars tiene 3 tipos diferentes de rango,
y cada uno tiene acceso a diferentes tipos de recursos.

Los tres tipos de rangos son:

- Comandante
- Brujo
- Soldado

También hay un rango especial llamado *admin*. Los detalles los explicaremos más adelante.

Los tres tipos de recursos que hay disponibles son:

- Libros
- Armas
- Puertas (con código de acceso)

Las relaciones de acceso entre rangos y recursos son las siguientes:

- Los brujos pueden acceder a Libros.
- Los soldados pueden acceder a Armas.
- Las puertas tienen una lista de rangos que tienen acceso, p.ej ['brujo', 'comandante']

Admin

El admin es el único rango que puede crear un nuevo Craftworld. También **tiene la autoridad para promocionar un brujo a soldado, o un soldado a comandante**. Es responsable
de crear/borrar/modificar **nuevos recursos como libros, armas o puertas**.

Registro

En el proceso de registro, un eldar debe especificar el Craftworld al que quiere pertenecer. Este se identifica por el nombre del Craftworld o su _id (utiliza sólo uno por simplicidad si quieres).
Si el Craftworld no existe, se devuelve un error.

También, el proceso de registro debe especificar si el eldar es un warlock o un soldado.
Un eldar no puede registrarse como comandante.

Recurso

Un recurso (libro, arma, puerta) sólo puede pertenecer a uno de los Craftworlds.
Esto significa que sólo los eldars del Craftworld al que pertenece el recurso pueden acceder a él.

Aquí tenéis un ejemplo de los Schemas para los recursos:

Libro:

```
{
    titulo: String,
    npaginas: Number,
    craftworld: ObjectID (Ref: Craftworld)
}
```

Arma:

```
{
    modelo: String,
    calibre: Number,
    craftworld: ObjectID (Ref: Craftworld)
}
```

Puerta:

```
{
    piso: Number,
    acceso_a: String,
    codigo_acceso: String,
    rangos: [String],
    craftworld: ObjectID (Ref: Craftworld)
}
```

Usuario

El usuario tiene un esquema tal que así:

```
{
    nombre: String,
    codigo_acceso: String,
    craftworld: ObjectID(Ref: Craftworld),
    rango: String
}
```

Después del registro, un Usuario puede autenticarse en el sistema para obtener acceso a los recursos a los que puede acceder dadas las condiciones expuestas anteriormente.

Craftworld

Un craftworld tiene un Schema similar a este:

```
{
    nombre: String,
    poblacion: Number
}
```

En todos los casos, no os limitéis a especificar el esquema tal cual está puesto aquí, añadid la validación que creáis oportuna a los atributos. Por ejemplo, en este último caso, la población debería ser un natural, o en el caso del rango, este sólo puede tener los valores que hemos indicado.

URLs de la API

Conceptualmente, cada recurso pertenece sólo a un craftworld. Esto quiere decir que la url para acceder a un libro, por ejemplo, debe ser del tipo:

```
/craftworld/:id_craftworld/libro/:id_libro
```

Se pide

Programad una API Rest que maneje los accesos a los recursos disponibles en cada craftworld, dependiendo del usuario que esté logueado en el sistema.

Extra 1

Escribir un programa en node.js que pruebe todo el sistema. Podéis usar `http.request` o cualquier otro módulo (https://www.npmjs.com/package/request) para hacer peticiones a vuestra API y loguear/comprobar las respuestas para ver si son las correctas.

Extra 2

Añadir `coordenada_x` y `coordenada_y` a los Craftworlds. Permitid a los eldars de un Craftworld acceder a los recursos de otro craftworld que esté cerca del suyo. Recordad Pitágoras para calcular la distancia! Considerad *cerca* la distancia que queráis.

Extra 3

Añadir stock a los libros y armas. Un atributo `stock: Number`. Cuando alguien obtiene un recurso, el stock baja en una unidad. Los usuarios tendrían un atributo `recursos: [ObjectID] con todos los recursos que tienen.

Para devolver un recurso, hacer una petición GET al mismo objeto, y si ya lo tienes, lo devuelves, incrementando en uno el stock.

Si no hay stock de un recurso, no se permite acceso a él.

Info extra

Los extras son totalmente opcionales y permitirán subir nota.

Este ejercicio es **muy importante**. Si lo conseguís hacer completamente significa que ya tenéis conocimientos más que suficientes para hacer una API Rest medianamente compleja y de calidad.

Es por esto que con este ejercicio podréis compensar malas notas de ejercicios anteriores.