# 1 Visualización de entradas

Crea una nueva aplicación web en la que se muestren 20 entradas disponible (Array tickets). Cada entrada va a tener una **id** y un **owner**.

const tickets = [

  { id: 1, owner: null },

  { id: 2, owner: null },

  { id: 3, owner: null },

  { id: 4, owner: null },

  { id: 5, owner: null },

  { id: 6, owner: null },

  { id: 7, owner: null },

  { id: 8, owner: null },

  { id: 9, owner: null },

  { id: 10, owner: null },

  { id: 11, owner: null },

  { id: 12, owner: null },

  { id: 13, owner: null },

  { id: 14, owner: null },

  { id: 15, owner: null },

  { id: 16, owner: null },

  { id: 17, owner: null },

  { id: 18, owner: null },

  { id: 19, owner: null },

  { id: 20, owner: null }

];

La aplicación debe contener los siguientes endpoints.

* **GET /tickets** retorna una lista con todos los tickets de la aplicación.
* **GET /tickets/:id** retorna un objeto con la información del ticket con la id correspondiente.Sí la **id** es de un ticket que no existe en la aplicación se debe retornar un error.

# 2 Añadir nuevas entradas

Crea un nuevo endpoint para agregar nuevos tickets en la aplicación anterior.

* **POST /tickets** añade un nuevo ticket a la lista anterior, este endpoint no recibe ningún dato por el body, simplemente añade un nuevo objeto a la lista de tickets, la propiedad **id** debe ser la inmediatamente siguiente al último ticket de la lista y el **owner** debe ser null.

# 3 Reservar una entrada

Crea un nuevo endpoint para reservar tickets en la aplicación anterior. Reservar un ticket consiste en modificar la propiedad **owner** del objeto.

* **PUT /tickets/:id** este endpoint recibe en el body un dato **owner,** almacena ese **owner** en el tiket (de la listaTikets) con la **id** correspondiente.

Se debe retornar un error y no permitir reservar tickets cuando:

* La **id** sea de un ticket que no existe en la aplicación
* Se intente reservar un ticket ya reservado, es decir que ya tenga un **owner**

# 4 Eliminar una entrada

Crea un nuevo endpoint para eliminar un ticket de en la aplicación anterior.

* **DELETE /tickets/:id** este endpoint elimina el ticket con la **id** indicada.

Se debe retornar un error y no permitir reservar tickets cuando:

* La **id** sea de un ticket que no existe en la aplicación
* Se intente eliminar un ticket ya reservado, es decir que ya tenga un **owner**

# 5 Gestión de precio de las entradas

Añade una nueva propiedad **price** a todos los tickets, inicialmente con el valor **10** en todos.

Permite que el endpoint de modificación **PUT /tickets/:id** pueda recibir un dato **price** , este precio modificará el **price** anterior del ticket indicado en la **id**

Se debe retornar un error y no permitir reservar tickets cuando:

* La **id** sea de un ticket que no existe en la aplicación
* Se intente eliminar un ticket ya reservado, es decir que ya tenga un **owner**

Los tickets agregados mediante **POST /tickets** también deben contener una propiedad **price**

# 6 Visualización de entradas con query

Modifica el endpoint **GET /tickets** para que de forma opcional pueda recibir un dato por **query** con clave **owner**, retornará todas las entradas de ese **owner**.

<http://localhost:3000/tikets?owner=jordan>

Modifica el endpoint **GET /tickets** para que de forma opcional pueda recibir un dato por **query** con clave **lessPriceThan**, retornará todas las entradas que tengan menos precio del enviado.

<http://localhost:3000/tikets?lessPriceThan=300>