

Backend

Challenge

To create a REST API with Node.js.

The challenge consists in a REST API that offers the **management of a resource: cards**

A card is defined by its name, image and rarity.

There are two types of cards: regular and limited. So, the owner should be able to define how many cards will be available for those that are limited.

A card could be published, so other users can see it. Or not published, so, no one except the owner could have access to it.

The name, image, rarity and the property that says how many cards are available for limited cards, are required in order to publish the card. But they are not required if the card is not published.

The application that consumes this **API needs a way to**:

* Create a card.
* Update properties of a card like name, image, rarity or available number of cards for limited cards.
* Publish / unpublish many cards in bulk.
* Retrieve the cards of a given owner.
  + When the owner retrieves the cards, she will have access to the following statistics: For each card, the total number of cards obtained and the total number of cards used by all the users.
  + When another user retrieves the cards, she will have access to the following statistics: For each card, the total number of cards obtained and the total number of cads used by herself.

You have not to implement anything regarding the obtained and used cards domain. But we are interested in discussing the way they could be retrieved or composed. You can mock or use pseudocode to refer to other parts of the system you imagine you will need but are not built yet.

It is not necessary to implement a user system. Instead, you can use the following JSON Web Tokens for privileged access, and implement rules based on the userId.

Owner of the resource:

|  |
| --- |
| eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdHJlYW1sb290cy5jb20iLCJ1c2VySWQiOiI1YTUwMTU5MzA4ZjVhODAwMTExZGU3NTkiLCJpYXQiOjE1MTYyMzkwMjJ9.mj8-t--lfImQGg8HoA\_9XOvDlunl3YJoPttkIbOHNMU |

Another user:

|  |
| --- |
| eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdHJlYW1sb290cy5jb20iLCJ1c2VySWQiOiI1YTUwMTU5MzA4ZjVhODAwMTExZGU3NTAiLCJpYXQiOjE1MTYyMzkwMjJ9.ArXF6iD5tX0DkKiS0EG3y30Bl3g\_E8iLPkk98hJw0Pc |

You should take into consideration the performance and scalability of the system. Consider that the cards retrieving could be accessed by 200,000 users in a period of 2 minutes.

Also, there are users with more than 1,000 different cards.

Finally, each time an action like a card is created or a card is published, we should **send** the details of the action **to two different analytics platforms**.

You can mock the calls to the analytic platforms.

**We’d like you to:**

* Analyze the problem.
* Design a solution.
* Implement the most relevant parts.
* If there are some parts or architecture elements / considerations you find interesting to talk about but there is not enough time to implement them, describe them and propose an implementation path delivering value to the user at least each 2 weeks.

**Bonus (optional)**

* Follow port and adapters / hexagonal architecture.
* TypeScript.
* Dockerize the solution.

**Notes**

* You can make all the assumptions you want or ask for more information.
* We don’t expect you to spend more than 5 - 8 hours in the resolution. We encourage you to work in the most valuable things for the position.
* You could choose the persistence engine you prefer for the challenge. At Streamloots we use MongoDB.

**Resources**

* MongoDB Cloud Atlas Free Tier Cluster <https://docs.atlas.mongodb.com/tutorial/deploy-free-tier-cluster/>