Back-end [Laravel]

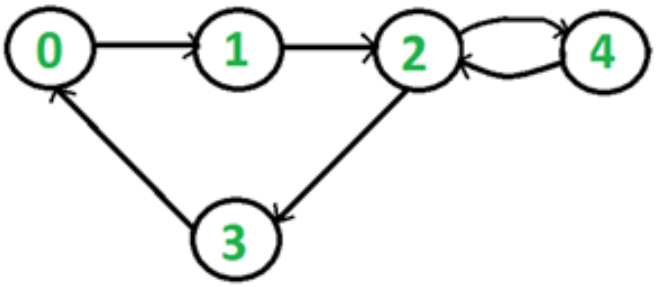
Goals :

Show your back-end coding skills, including design quality and how readable the source code is.

Duration: 24 hours.

Description: Build a REST API to manage graphs using **Laravel 8.x**

Definition: A graph is a set of nodes in which certain pairs of nodes are linked.



A. Terminology and resources

Graph

- Each graph has an id, a unique name, description, created_at and updated_at;
- Each graph can have a set of nodes.

Node

- Each node has an id;
- Node has only one graph;
- Nodes can be linked with relation.

Relation

Each relation has parent node and child node.

B. Specifications

1. Endpoints

The API should be able to:

1. Create an empty graph;
2. Edit graph meta data (name, description);
3. Delete graph;
4. Get all graphs (only meta data);
5. Add node to a specific graph;
6. Add relation to a specific graph;
7. Update the graph shape (all nodes and relations)
8. Get single graph with its nodes and relations;
9. Delete a specific node.

> You should document all endpoints in its controller action (request/response body)

2. Artisan commands:

Create 3 artisan commands:

2.1. Generate a random graph

```
php artisan graph:gen --nbNodes={$nbNodes}
```

This command should create a random graph with `$nbNodes` nodes and random relations.

2.2. Clear empty graphs

This command should delete all empty graphs.

```
php artisan graph:clear
```

2.3. Graph stats

This command display graphs stats (display the graph meta data, number of nodes, number of relations) by graph id.

```
php artisan graph:stats --gid={$graphId}
```

C. Global specifications

1. Use git and a gitHub Repository;
2. Prepare the application design (database schema, app modules)
3. Apply design patterns to solve the app design problems;
4. Create a reusable modules, repositories…;
5. Support REST API and HTTP standards;
6. Support mysql;
7. Unit tests are not required.