Take-home project

This take-home challenge aims to assess the technical ability of candidates applying for a back-end software engineering position at Root. Completing this exercise should demonstrate that you can:

1. *Understand and interpret requirements*
2. *Demonstrate proficiency in a programming language*
3. *Build an HTTP Web API*
4. *Write clean, well-structured code*

The deliverables are a ZIP file containing:

1. *The created project/code*
2. *readme.md file answering the following:*
    a. *How to set up and run the code*
    b. *Briefly explain how you approached the problem*
    c. *What you would do differently if you had to do it again*
    d. *What you learned during the project*
    e. *How you think we can improve this challenge*
    f. *Technical questions as set out in the description of the challenge below*

As a guideline, this challenge should ideally take 4-6 hours, but actual completion times may vary.

PS: please do not share this challenge with anyone.

Brief

Dinopark covers an expansive area. As such, maintenance of facilities and security features (fences, cameras, etc.) are a constant effort. Here at Park Maintenance, we take our jobs very seriously. If we didn't, someone would get hurt! We need you to create a tool to help make our jobs safer and more efficient. Last year we lost eight workers. We really want to get that number down.

Let me tell you a little bit more about our operations:

- Our team divides the park into a 26 by 16 grid of zones. The letters A-Z represent the columns and the numbers 0-15 represent the rows. For example, A13 is the first column and the 14th row.
- Each zone in the grid needs maintenance every 30 days.
- Maintenance staff can only perform maintenance in a zone when it is *safe* to do so
    - *Safe* means: No carnivores are in a zone or all carnivores in the zone are still digesting their last meal (all dinosaurs in the park have an estimated digestion time which is logged when the animal is first moved into the park)

We'd like you to design a backend which processes logs from our park's monitoring system (NUDLS™) and exposes the status of dinosaurs over a RESTful API. It's important to know whether a zone needs maintenance or is safe to enter.

Our outsourced front-end developers will then use this system in our operations dashboard, implementing this design that plots out the grid, colour coding the zone on the map according to how safe they are to enter and which areas need maintenance. Due to the lack of communication we have with our front-end developers, please make sure the API is simple to understand, and well documented (in the readme). **Please note: The NUDLS™ feed to the park is unreliable and goes down from time to time. Please ensure that your solution takes this into account.**

NUDLS™ offers a webhook integration point which POST's events to a given endpoint. Unfortunately, due to very backwards reasoning, Park Security doesn't want to give you NUDLS™ access until the software has been completed. We, however, found an exposed event-feed endpoint that you can consume using a GET request (https://dinoparks.net/nudls/feed). I've attached the NUDLS™ developer docs (such that they are). Hopefully, this should be enough to get you going.

From what we could gather, speaking to Park Technical and the outsourced front-end developers, the system should entail the following things:

- RESTful CRUD endpoints for the front-end developers to consume park data
- A persistence layer (DB) to store and maintain the park and dinosaur state
- Tests, to ensure workers aren't accidentally eaten upon entry

Finally, Park Technical has asked for some insight about the following things (please include this in the README):

- Given that lives depend on this software, Park Technical has insisted on a 99.99% uptime guarantee. What would you do to make the service more resilient? (i.e. to optimise uptime/high availability)
- The Mainland Park, which houses the largest number of dinosaurs, wants to roll this out assuming the first version is a success. It's expected that the park will house more than 1 million dinosaurs in the next 12 months. What would you change to handle this level of scale on the system. Or rather, what do you expect to break, given the solution you provided?
- Someone recommended that we use GraphQL. Park Technical has no idea what this is, but has asked for your advice on whether to implement this or not. What is your recommendation?

NUDLS (New Unified Dinopark Logging System) Developer Documentation

Version 1.4.7-a5, build 10

NUDLS is a system for allowing employees of <u>*Dinoparks Amusement and Betting Company GMBH*</u> access to an unfiltered view of park operations. NUDLS uses the new <u>*JSON*</u> (JavaScript Object Notation) spec which was just ratified.

NUDLS uses the idea of a global persistent log as a unifying abstraction. After adding a new system, NUDLS sends the newly integrated system *all* previously persisted events one by one. NUDLS uses the HTTP (Hypertext Transfer Protocol) as a transport mechanism, sending logged messages using a POST (Personal Object Store Transfer) request.

NUDLS uses ISO standards for all datatime needs.

The body of this request may be any one of the following kinds (along with a sample message attached)

Dino Added

This event is called when new animals are added to the park. Digestion period should be used to inform feeding schedule.

```
{
"kind": "dino_added",
"name": "McGroggity",
"species": "Tyrannosaurus rex",
"gender": "male",
"id": 1032,
"digestion_period_in_hours": 48,
"herbivore": false,
"time": "2018-04-20T11:50:53.460Z",
"park_id": 1
}
```

Dino Removed

This event is called when an animal is removed from the park

```
{
"kind": "dino_removed",
"id": 1032 ,
"time": "2018-04-20T11:50:53.460Z",
"park_id": 1
}
```

Dino Location Updated

This event is called when an animal moves to a different cell in a standardized park grid.

```
{
"kind": "dino_location_updated",
"location": "E10",
"dinosaur_id": 1032,
"time": "2018-04-20T11:50:53.460Z",
"park_id": 1
}
```

Dino Fed

Indicates when a dinosaur was fed

```
{
"kind": "dino_fed",
"id": 1032,
"time": "2018-04-20T11:50:53.460Z",
"park_id": 1
}
```

Maintainence Performed

Indicates that a routine maintenance pass was made at the given cell in the standardized park grid. Please see Winged Avengers: The Dinopark Employee Handbook and Survival Guide, Page 107 for full details on what routine maintenance activities activities entail and how it may affect **you**

```
{
 "kind": "maintenance_performed",
 "location": "E7"
 "time": "2018-04-20T11:50:53.460Z",
 "park_id": 1
}
```