➔ Technical Test Ruby developer

Many apartments have *a central heating system* and due to different room isolation properties it's not easy to keep the same temperature among them. To solve this problem we need to collect *readings* from IoT *thermostats* in each apartment so that we can adjust the temperature in all the apartments in real time.

The goal of this task is to build **a basic web API** for storing readings from IoT thermostats and reporting a simple statistics on them.

We are going to have two models with the following properties:

Thermostat:

- ID
- household_token (text)
- location (address)

Reading:

- ID
- thermostat_id (foreign key)
- number (sequence number for every household)
- temperature (float)
- humidity (float)
- battery_charge (float)

The API consists of 3 methods. In addition to its own parameters **each method** accepts a *household token* to authenticate a thermostat. Serialize an HTTP response body using JSON. The methods are:

1. POST Reading: stores *temperature*, *humidity* and *battery charge* from **a particular thermostat**. This method is going to have a very high request rate because many IoT thermostats are going to call it very **frequently and simultaneously**. Make it as fast as possible and **schedule a background job** for writing to the DB (you can use Sidekiq for example). The method also returns a generated sequence *number* starting from 1 and **every household has its own sequence**.

2. GET Reading: returns the thermostat data using the *reading_id* obtained from POST Reading. The API **must be consistent**, that is if the method 1 returns, the thermostat data must be immediately available from this method **even if the background job is not yet finished**.

3. GET Stats: gives the *average*, *minimum* and *maximum* by temperature, humidity and battery_charge **in a particular thermostat across all the period of time**. Again, make sure this method is **consistent** in the same way as method 2. For extra points, make it execute in O(1) time.

For simplicity, you can seed the DB with different thermostats containing household tokens and locations. Make sure your code is properly tested with *RSpec* as well. You can use any tools and gems to build and optimize your API. For extra points, handle bad requests with a JSON error message and a non success response code.

Submission

Please submit a fully completed answer . Please provide source code and full GIT repository alongwith tests. Please commit often and with good commit messages. That will allow us to see how you've approached the problem. Don't worry about changing things around often