TRUE NORTH

SR NODEJS TEST

Restonode is an restaurant management system, actually works handling all the administration and now they want to implement a new delivery system module, this new module is going to be in charge of all the new orders from the customers via a website

NOTE: For this challenge only we are going to cover the backend part of the implementation

The solution consists in two services one backend api and one order messaging service

The Backend API must be able to do the following actions

- Possibility to users add rating to a restaurant
- List restaurants and filter by rating
- Create a new delivery order, the new order should have one or more **meals**, **total cost**, **address** and a **latLng** position of the place, this should save the order and return ETA (estimated time of arrival) based on user location and restaurant location (transport media being a motorcycle), this time should have in count traffic at the moment, also when a order is triggered a message needs to be created and queue in RabbitMQ server 1 message for the notification and another for the order

Order messaging service

- Recover messages queued by Backend API handling order messages and notification messages
  Action desired for each type of message
  - **Order messages** needs to send an email informing to the restaurant a new order is in place
  - **Notifications messages** need to send sms to the customer with the confirmation of the order (for practical reasons can be mocked and replaced with a write on log)

Stack

- NodeJS , you can use Hapi.js or Express
- Mocha (Lab if you are using Hapi.js) and Chai for unit testing and Sinon for mocking and spies
- PostgreSQL or any relational database
- You can use google maps or any API or any you feel more convenient
- RabbitMQ or similar

Acceptance Criteria

- All setup scripts should run without errors (including any custom pre/post install and sql scripts)
- Prefered to use async / await over promises when it’s apply
- Swagger can be used as documentation but not to generate parts of the App
- Unit Testing: at least a couple of methods should be unit tested
- All methods should handle exceptions
- Include Postman collection for calling all the API endpoints you created
- Include a high level architecture documentation. If you have AWS experience then please include how you would utilize AWS resources in your documentation.

You may send any questions directly to your recruiter.

Submission

Send us a zip file with entire project or a git public repo - In your submission, please include clear instructions for running the solution that you created.