EVERYTHING FAILS.
ALL THE TIME.

Person of contact:

Eva Nelissen

Tech Recruiter
eva.nelissen@coolblue.nl

BE developer assignment

So you want to be a Backend Python Engineer at Coolblue Planning Tech? Great! We need you. In this assignment you will demonstrate your skills and creativity as it is required from all developers in Planning Tech. It reflects real use cases at our company and a technological stack that is representative.

You should be able to complete this assignment in an estimated 6-8h of work.

Purpose

The purpose of the assignment is to assess your

- programming skills,
- creativity,
- knowledge of the field ("what is out there"),
- clearness of thought.

Problem

You work for a planning team and the algorithm guys have a library for solving what is called the **Travelling Salesman Problem**: finding the cheapest (shortest) path or sequence in order to visit a set of N locations, numbered 0, 1, ..., N with 1 vehicle.

As a back-end engineer you are responsible for developing a wrapping service around this library that communicates through publish/subscribe queues. The service subscribes to an inbound queue to receive instances of the problem as input and publishes the solution to the problem in the outbound queue. You are free to design the contract of the messages and the protocol format used in the messages. Keep in mind that the input is a set of N lat-lon geocoded locations. You will need the pairwise distance of these locations (it is input to the problem-solving library. For assignment purposes you can randomly generate these locations and use the Euclidean distance: for points p,q,

```
d(p,q) = sqrt((p_x - q_x)**2 + (p_x - q_y)**2)
```

Requirements

- A python project, that is, not only Python code but an actual project (structure, packaging, etc...) set up in a github repo.
- There should be tests but in the interest of time, for this assignment, there is no need for full test coverage, just a few complete tests.
- Use docker: we are an events and micro-services oriented company.
- The project should be functional. An `example.sh` script should install and run the application.
- Use or-tools as the underlying optimization engine. To be more to the point you can use this tutorial (use the default parameters, the quality of the solution is *not* what we are interested in here in this assignment).
- Write *some* documentation explaining the design.

Bonus

If the basic assignment has been completed (we estimate 6h-8h work for this assignment) and you have time over, you can try to go after the following hints for extra points (in any combination):

- Make a RESTful web-service with a few endpoints allowing the customer to input problem instances and obtain results from the underlying pub/sub queues. For simplicity, this service should be stateless.
- Make the service deployable via `docker-compose`.
- Implement logging.
- Add support for more than 1 vehicle (the underlying optimization package supports it).
- Add time-constraints to the visits, i.e., location `i` *has* to be visited between times `a_i` and `b_i` .

Tips

- Feel free to use `setuptools` and pip for packaging but keep in mind the new developments in this area ;)