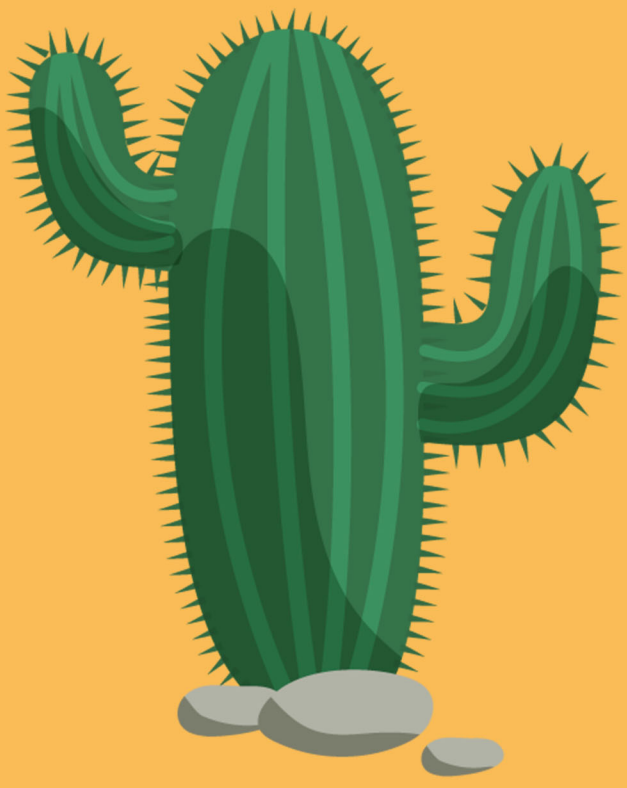
CACTUS HEAVEN



Small assignment IV

We have been contacted by a new retailer in the cactus biz called **Cactus Heaven**. They love cactus and want everyone to feel the same way! They started out as a small company with around 500-1000 daily users to their website https://cactus-heaven.com (*i'm not sure if this actually exists… but you can check!*) but they are rapidly expanding and their daily users are now around 4000-5000. The codebase they currently have is a monolithic application which is neither scalable nor easy to deploy, and they want to move to the world of microservices! That's where we come in, let's step in to the land of wonders called Microservice-Land!

Template

This assignment comes with a template which can be downloaded from **Canvas** (*template.zip*). The template includes the following:

- **api_gateway/** *- This is the API gateway which encapsulates all the services and the client can access. This service is partially implemented. It is accessible through port 3000*
- **email_service/** *- This is the email service used to send a confirmation email to the user when he has successfully ordered. This service is already implemented.*
- **logging_service/** *- This is the logging service used to log all transactions made within the system, both error and successful.*
- **order_service/** *- This is the order service used to store the orders within a MongoDb database. The schema is already setup in **data/schema** and includes two schemas Order and OrderItem. These schemas should be used to create the order and the items associated with the order.*

Dependencies

This assignment depends on **RabbitMQ** and that needs to be setup locally. Follow this walkthrough to setup **RabbitMQ** to your system: https://www.rabbitmq.com/download.html. After you setup **RabbitMQ** you need to run it within a command line: **rabbitmq-server** and it should run on port 5672 and the management UI on port 15672. All services depend on this server so it needs to be running to test the other services. You should also setup **Python** because the **email_service** is written in **Python** and needs to be executed. Setup **Python 3.x** (*for gods sake…*) and use execute **pip install requests pika** in the **email_service/** folder.

Assignment description

All services should use a common exchange called "order_exchange". Below the functionality of the application is described:

API GATEWAY (20%)

1. (**5%**) Add a route **/api/orders [POST]**

   The data sent to the route should look like this:

```
{
    "email": "arnarl@ru.is",
    "items": [
        {
            "description": "A",
            "quantity": 1,
            "unitPrice": 2000
        },
        {
            "description": "B",
            "quantity": 2,
            "unitPrice": 3000
        }
    ]
}
```

2. **(15%)** When this route is called the following should happen:
    2.1. **(5%)** An order created event should be emitted with the routing key “create_order”
    2.2. **(5%)** The body of the request should be sent as data with the events
    2.3. **(5%)** It should always return 200 OK (*it doesn’t care what happens*)

LOGGING SERVICE (40%)

It should be implemented as a console application in **.NET Core** using **RabbitMQ**

1. **(10%)** It should consume the order created event using the queue “logging_queue”
2. **(30%)** It should store the data within the logging event prefixed with “**Log: “** within a file called **log.txt**

ORDER SERVICE (40%)

It should be implemented as a **NodeJS** application using **RabbitMQ**

1. **(10%)** It should consume the order created event using the queue “order_queue”
2. **(15%)** It should create a new order using information from the event
3. **(15%)** It should also create order items using information from the same event

Submission

A single compressed file (***.rar, *.zip**) containing all your code should be submitted to **Canvas**. Don’t forget to exclude **node_modules/** for the projects that contain that folder! Also don’t forget to comment the names of each group member (*excluding the one who submitted*).