Assignment 1: API

Camera Service (already implemented)

The camera services simulate the surveillance cameras at the airport. Each image produced signifies an event involving motion (e.g., a person entering or exiting a specific airport section), depending on the camera's location. When activated, this service streams images (frames) to a designated destination (URL). Captured images are represented in two ways:

- **Raw Image:** A plain raw JPEG image data with no other info attached ("image/jpeg").
- **Frame Object**: An object containing base64-encoded imaged data, the timestamp when the image was captured, ID of the section in which the photo was taken, random uuid, and whether the person was entering or exiting the area at the time of capture.

Note: Since real cameras are not available for this exercise, the service uses a dataset of human faces collected from the internet (see more http://vis-www.cs.umass.edu/fddb/). The Camera service selects up images from this dataset and sends them to the destination service via POST requests.

The Camera service API specification is as follows:

[Base URLs: http://localhost:311xx (host network), http://cameraX* (container network]
*X can be a number of the camera, or can be omitted.

GET /config

Returns the current configuration of the Camera service.

Response ("application/json"):
```
{
    "section": 1,
    "event": "entry",
    "name": "Camera X" (optional)
}
```

Codes: 2xx if the operation was successful, otherwise 4xx.

PUT /config

Configures the Camera service according to the configuration provided

Body ("application/json"):
```
{
    "section": 1,
    "event": "entry",
    "name": "Camera X" (optional)
}
```

Codes: 2xx if the operation was successful, otherwise 4xx.

POST /stream?toggle=on

Turns ON or OFF streaming to the specified destination (URL). This functionality will send Frame objects in the JSON format to the destination. If the toggle is not supplied, "on" is the default value. The "**max-frames**" value limits the number of frames that are sent to the destination. If it is not supplied, all available images are sent. The delay field is used to specify a delay in seconds between subsequent requests (default is 0 or no delay).

Body ("application/json"):

```
{
    "destination": "<destination-url>",
    "max-frames": 10, (optional)
    "delay": 0.1, (optional)
    "extra-info": ""  (optional)
}
```

Response ("application/json"):

```
{
    "code": 207,
    "type": "STATUS",
    "message": "Streamed 1 frame(s) in x seconds, with 0 failed requests."
}
```

Codes: 2xx if the operation was successful, otherwise 4xx.

GET /frame (primary for testing purposes)

Captures an image and returns it in a JSON object or as a raw image to the requestor. You can use HTTP headers to switch between the types.

Response (if the request header has **Accept: "application/json"**):

```
{
    "timestamp": "2020-10-24T19:28:46.128495",
    "section": 1,
    "event": "exit",
    "image": "<base64-encoded-string>",
    "frame_uuid": "<uuid>",
    "extra-info": ""
}
```

Response (if the request header has **Accept: "image/jpeg"**):

```
raw image, with "image/jpeg" content-type
```

Codes: 2xx if the operation was successful, otherwise 4xx.

You can pass environment variables when starting this service, “SECTION_ID” and “POSITION” automatically set the configuration of the section id (integer) and event (string) to specified values.

ImageAnalysis (already implemented)

This service analyzes a given image for human faces and estimates the age and gender of detected individuals (note: it does not perform person recognition). The resulting JSON data is either returned back as a response or forwarded via a POST request to the specified destination.

The ImageAnalysis service API specification is as follows:

[Base URLs: http://image-analysis (container network)]

POST /frame

Receives an image (i.e., frame in JSON) for analysis and returns a list with estimated information on detected persons.

Body ("application/json"):

```
{
    "timestamp": "",
    "image": "<base64-encoded-string>",
    "section": 1,
    "event": "entry",
    "destination": "<destination-url>", (optional)
    "frame_uuid": "<uuid>", (optional)
    "extra-info": "" (optional)
}
```

Response ("application/json"):

```
{
    "timestamp": "2010-10-14T11:19:18",
    "section": 1,
    "event": "entry",
    "persons": [
        {
            "age": "60-100",
            "gender": "Male"
        }
    ],
    "extra-info": "", (optional)
    "frame_uuid": "<uuid>", (optional)
    "image": ""<base64-encoded-string>"
}
```

Codes: 2xx if the operation was successful, otherwise 4xx.

POST /frame (raw image)

Receives an image for analysis and returns a list with estimated information on detected persons.

Request Headers: Content-Type: "image/jpeg"

Body ("application/json"):

```
raw image, with "image/jpeg" content-type
```

Response ("application/json"):

```
{
    "persons":[
    {
        "age": "8-12",
        "gender": "female"
    }
    ]
}
```

Codes: 2xx if the operation was successful, otherwise 4xx.

Remarks: If the "**destination**" field is provided, the result is forwarded to the destination.

FaceRecognition (already implemented)

This service compares the received image against the predefined database of persons of interest to identify potential matches.

The FaceRecognition service API specification is as follows:

[Base URL: http://face-recognition (container network)]

POST /frame

Receives a frame for analysis in JSON format and returns a list with information about detected persons in JSON. If the "destination" field is provided and some individuals were recognized, the result is forwarded via POST to the destination. If the operations is successful but no individuals were recognized, a 204 status code is returned with an empty response.

Request Body ("application/json"):

```
{
    "timestamp": "2010-10-14T11:19:18",
    "image": "<base64-encoded-string>",
    "section": 1,
    "event": "entry",
    "destination": "<destination-url>", (optional)
    "frame_uuid": "<frame_uuid>", (optional)
    "extra-info": "" (optional)
}
```

Response ("application/json"):

```
{
    "timestamp": "2010-10-14T11:19:18",
    "section": 1,
    "event": "entry",
    "image": "<base64-encoded-string>",
    "known-persons": [
        { "name": "Mr. Y" }
    ],
    "frame_uuid": "<frame_uuid>", (optional)
    "extra-info": "" (optional)
}
```

Codes: **200/204** if the operation was successful, otherwise **4xx**.

POST /frame (raw image)

Checks if the provided image contains any known persons of interest. Accepts raw JPEG image data and returns a list of detected persons' names.

Request Headers: Content-Type: "image/jpeg",

Request Body:

```
raw image, with "image/jpeg" content-type
```

Response ("application/json"):

```
{
    "known-persons": [
        { "name": "Ms. X" }
    ]
}
```

Codes: **2xx** if the operation was successful, otherwise **4xx**.

Remarks: It may take a while until some individuals are recognized. In case the "**destination**" field is provided, the result is forwarded to the destination (unless no one was recognized).

Section (already implemented)

Section service(s) take care of the statistical data for airport sections, recording all individuals who pass through each section using MongoDB. By default, each Section service stores its data in a local MongoDB database. If you decide to use a single database for all Section service instances, then you can configure that with the following environment variables:

- EXTERNAL_MONGODB_URL=<mongodb://<hostname:port/>
- EXTERNAL_MONGODB_DB=<db-name>
- EXTERNAL_MONGODB_COLLECTION=<collection-name>

The Sections service API specification is as follows:

[Base URLs: http://localhost:314xx (host network), http://section (container network)]

POST /persons

Creates a new entry in the list of detected persons passing through this section.

Body ("application/json"):

```
{
  "timestamp": "2010-10-14T11:19:18",
  "section": 1,
  "event": "exit",
  "persons":[
      {
        "age": "8-12",
        "gender": "male"
      }
  ],
  "image": "<base64-encoded-string>", (optional)
  "frame_uuid": "<uuid>", (optional)
  "extra-info": "" (optional)
}
```

Codes: 2xx if the operation was successful, otherwise 4xx.

GET /persons?from={from}&to={to}&aggregate=count&event=exit&section=1

Returns a JSON containing a list of individuals in the specified section within a given time frame (using the time format shown below). Additionally, it allows using aggregate=count for returning the total number of persons in the section, aggregate=time for returning the total time between the first and the last received event, and filtering by event type. Filtering by section number is optional and used only when all Section service instances are using the same database.

Response ("if query is: ?from={from}&to={to}&event=entry"):

```
{
  "persons":[
      {
        "uuid": "0aa57c62-68db-4d38-bdd2-5ea47dds3d2e",
        "age": "8-12",
        "gender": "male",
        "timestamp": "2010-10-14T11:19:18.039111",
        "section": 1,
        "event": "entry",
        "frame_uuid": "<uuid>" (optional)
        "extra-info": "" (optional)
      }
  ]
}
```

Response ("if query is: ?from={from}&to={to}&aggregate=count"):

```
{
    "count": 42
}
```

Codes: 2xx if the operation was successful, otherwise 4xx.

Alert (already implemented)

This simple service manages system alerts, which indicate detection of persons of interest.

The Alert service API specification is as follows:

[Base URLs: http://localhost:315xx (host network), http://alert (container network)]

POST /alerts

Creates a new alert by providing an array of person names, as well as the section number, event type, image of the face, along with the timestamp when it was captured.

Body ("application/json"):

```
{
  "timestamp": "2010-10-14T11:19:18",
  "section": 1,
  "event": "exit",
  "known-persons":[
      {
        "name":"Ms. X"
      }
  ],
  "image": "<base64-encoded-string>", (optional)
  "frame_uuid": "<uuid>", (optional)
  "extra-info": "" (optional)
}
```

Codes: 2xx if the operation was successful, otherwise 4xx.

GET /alerts?from={from}&to={to}&aggregate=count&event=entry

Returns a JSON with a list of alerts in a given time frame. Additionally, it allows aggregate=count for returning the total number of created alerts and filtering by event names.

Response ("if query is: /?from={from}&to={to}"):

```
{
  "alerts":[
      {
       "uuid": "5aa57c62-66db-4d38-bdd2-5ea37dd9352e",
       "timestamp": "2010-10-14T11:19:18.039111",
       "known-persons":[
              {
                "name": "Mr. Y"
              }
       ],
        "section": "1",
        "event": "entry",
        "image": "<base64-encoded-string>", (optional)
        "frame_uuid": "<uuid>", (optional)
        "extra-info ": "" (optional)
      }
  ]
}
```

Response ("if query is: /?from={from}&to={to}&aggregate=count"):

```
{
    "count": 42
}
```

Codes: 2xx if the operation was successful, otherwise 4xx.

GET /alerts/{uuid}

Return the alert details, i.e., person's info, timestamp, event, section, and the corresponding image.

Response ("application/json"):

```
{
  "uuid": "5aa57c62-66db-4d38-bdd2-5ea37dd9352e",
  "timestamp":"2010-10-14T11:19:18",
  "section": 1,
  "event": "exit",
  "known-persons":[
      {
         "name":"Ms. X"
      }
  ],
  "image": "<base64-encoded-string>", (optional)
  "frame_uuid": "<uuid>", (optional)
  "extra-info ": "" (optional)
}
```

Codes: 2xx if the operation was successful, otherwise 4xx.

DELETE /alerts/{uuid}

Removes the specified alert.

Codes: 2xx if the operation was successful, otherwise 4xx.

Collector (to be implemented)

This service primarily handles the communication flow within the system (at least in part). For example, when it receives an image frame (via POST /frame), the frame should eventually reach the Section service(s), and some frames need to reach the Alert service(s). However, each image also needs to be processed by the ImageAnalysis and FaceRecognition services beforehand. Depending on your design, the Collector service may first forward the received image (*frame*) for analysis to the ImageAnalysis service, which returns an estimated age and gender of the person (if a person is found on the image), then transform this data if necessary and sends it in the appropriate format to the Section service (see the Section service description for /persons) for storage of statistical information. Additionally, the service should also send the frame to the FaceRecognition service, which will try to identify any persons of interest. Note that if the "destination: <URL>" field is sent to the FaceRecognition or ImageAnalysis service, these services will try to send the result to the specified destination; otherwise, the result will be returned in response to the original requestor, which is the Collector service in this case. This service may interact with all or some of the other services, depending on your design decisions.

The Collector service API specification is as follows:

[Base URLs: http://collector (container network)]

POST /frame

Adds a new frame to the Collector service.

Body ("application/json"):

```
{
    "timestamp": "2010-10-14T11:19:18.03",
    "image": "<base64-encoded-string>",
    "section": 1,
    "event": "entry",
    "frame_uuid": "<uuid>",
    "destination": "<destination-url>", (optional)
    "extra-info": "" (optional)
}
```

Response ("if destination is provided"):

```
remote service response or just a code
```

Codes: 2xx if the operation was successful, otherwise 4xx.

POST /persons (optional)

Allows the user or another service to push detected person information.

Body ("application/json"):

```
{
  "timestamp": "2010-10-14T11:19:18",
  "section": 1,
  "event": "exit",
  "persons":[ { "age": "8-12", "gender": "male" } ],
  "destination": "<destination-url>", (optional)
  "image": "<base64-encoded-string>", (optional)
  "frame_uuid": "<uuid>", (optional)
  "extra-info": "" (optional)
}
```

POST /known-persons (optional)

Allows the user or another service to push detected information on persons of interest.

Body ("application/json"):

```
{
  "timestamp": "2010-10-14T11:19:18",
  "section": 1,
  "event": "exit",
  "known-persons":[  { "name": "Ms. X" }  ],
  "destination": "<destination-url>", (optional)
  "image": "<base64-encoded-string>", (optional)
  "frame_uuid": "<uuid>", (optional)
  "extra-info": "" (optional)
}
```

Response ("if destination is provided"):

```
remote service response or just a code
```

Codes: 2xx if the operation was successful, otherwise 4xx.