Test task: meeting room occupancy API

Abstract

The goal of this task is to implement a simplified back-end application that may be used to both collect and provide the data from an IoT network.

Use case and assignment

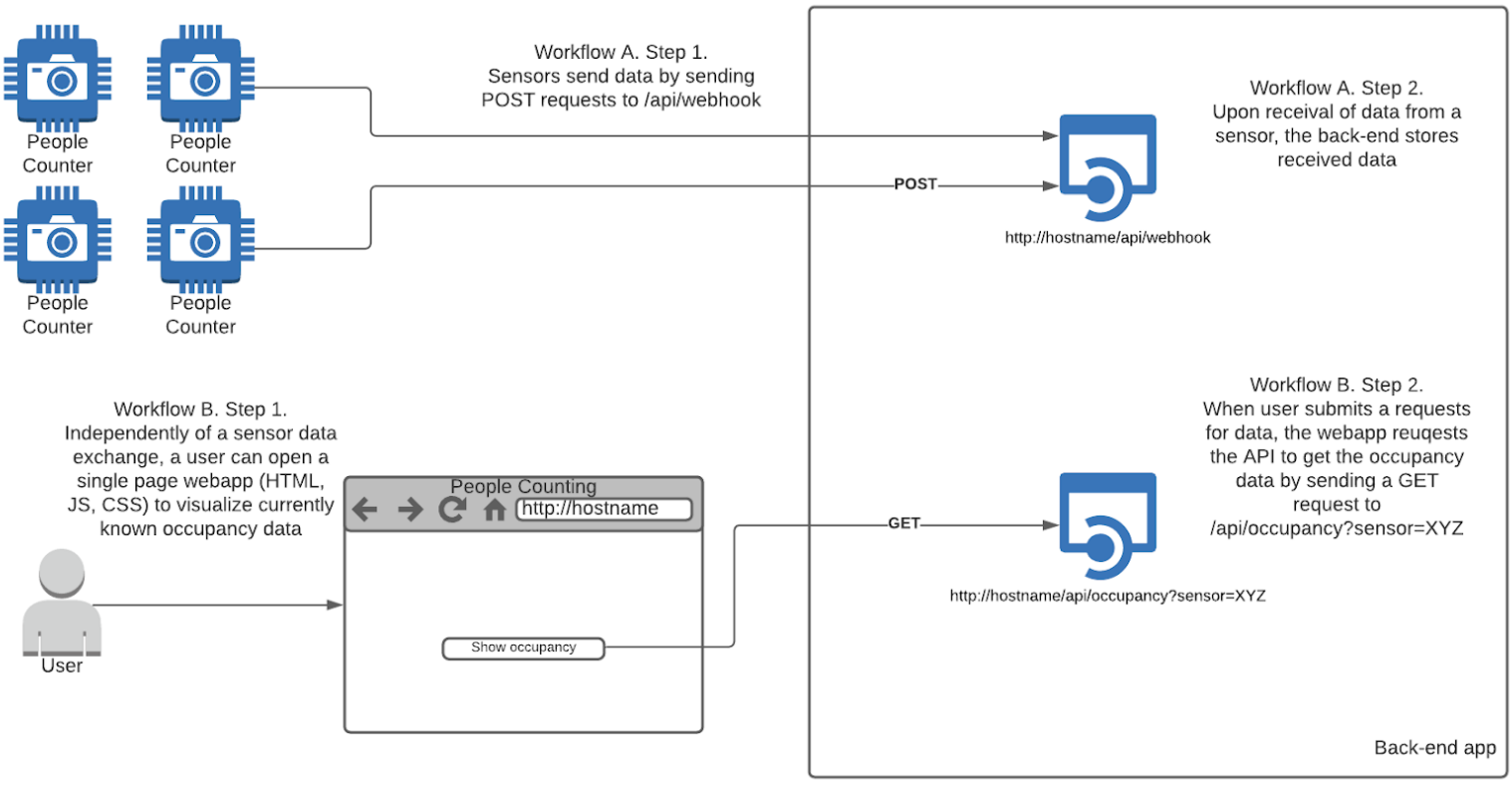
Context

An office building is equipped with sensors at entrances and exits of every meeting room. These sensors count the number of people who go into or out of the room. These sensors are further referred to as “People Counters”. All People Counters are connected to the Internet. A People Counter can be configured with a “webhook listener URL” and a sensor will send HTTP requests with the results of the observations to this URL. An HTTP request is produced by the sensor every time it detects a single person or multiple people; it indicates the number of people detected and their direction.
Building manager wants to monitor the number of people in the meeting rooms in real time.

Objective

The goal of this task is to implement a back-end web application to collect the data from the People Counters (HTTP webhook), and to expose collected data (HTTP RESTFul API)

Following diagram (next page) presents an example of a data flow in an imaginary system that collects and shows the collected data. The objective of this exercise is the implementation of the back-end application only.



The application should provide the following APIs:

1. Webhook Listener. An endpoint to be called by People Counters to send the results of the observations. Requests produced by People Counters may be emulated with the following command:
   ```
   curl --header "Content-Type: application/json" \
       --request POST --data \
       '{"sensor":"abc","ts":"2018-11-14T13:34:49Z","in":3,"out":2}' \
       http://hostname/api/webhook
   ```
   where `http://hostname/api/webhook` is this endpoint URL.

2. RESTFul API to provide the accumulated people counting data. These endpoints can be assumed to be used by a front-end Web or mobile application, for example.
   a. Get a list of known sensors.
      Sample request:
      `curl --request GET http://hostname/api/sensors`
      Expected HTTP response body:
      `{ "sensors": ["abc", "def", ...] }`
   b. Get a meeting room occupancy. (See *Functional Requirements* below for details of how the result is calculated).
      Sample request (**abc** below is an example of a sensor id):
      `curl --request GET http://hostname/api/sensors/abc/occupancy`
      Expected HTTP response body:
      `{ "sensor": "abc", "inside": 42 }`

Functional requirements

- At any point in time, occupancy of a meeting room is a sum of all entries minus a sum of all exits registered by a sensor since the start of the application. Occupancy is calculated for each room separately. Every meeting room is measured by a single dedicated sensor.

Technical requirements

- A solution to this exercise is the source code of a standalone application that may be eventually served from a web server.
- A solution may be written with any technology of choice and any programming languages, provided that the choice can be justified.
- The application may keep all data in memory. For the purpose of this exercise, there is no need to implement any persistent storage or use a database.

Optional assignments

- The implementation may be accompanied by unit tests.
- `occupancy` endpoint may take an optional query parameter: `atInstant`. When this parameter is defined the API is expected to provide the number of people that were present in the room at a given moment in time:
  ```
  curl --request GET
  http://hostname/sensors/abc/occupancy?atInstant=2018-11-14T14:00:00Z
  ```
  example output:
  ```
  { "inside": 9 }
  ```

Solution evaluation

Please, note that a solution will be evaluated for the following criteria, in the given order (former items have more importance than the latter ones):

- Correctness (the solution corresponds to the given requirements).
- Code quality.
- Simplicity, yet effectiveness of the software design.

The choice of a technology used to implement a solution is not a part of evaluation criteria, provided that it meets the Technical Requirements above.