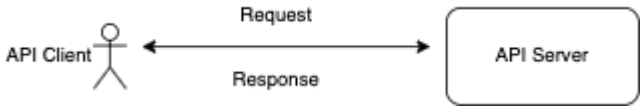
Edument Initial Assigment for NodeJS developer

This document outlines the initial assignment to carry out as part of Edument's recruitment process for the role of NodeJS developer.

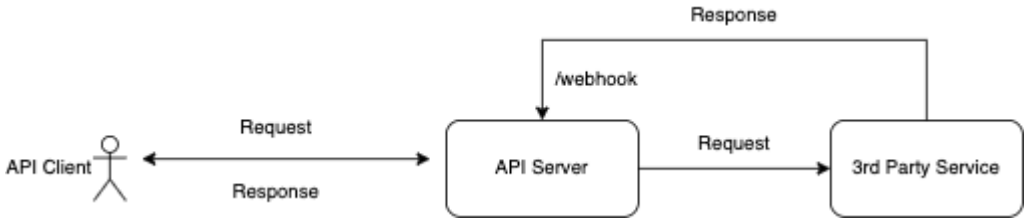
Overview

As a NodeJS developer you've begun an integration project at a company where the following infrastructure already exists:



When a client makes a (synchronous) HTTP request to the API, the API server executes application logic and returns a HTTP response.

The company has now decided to let a 3rd party service handle a large part of the application logic; the intended integration will result in the following architecture:



Notice that while the API client and the API server communicate *synchronously*, the API server has to interact with the 3rd party service *asynchronously*; the complete flow is as follows:

- The API client sends a request to the API server.
- The API server accepts the client request and *forwards* it to the 3rd party service.

  It *does not* return a response to the API client immediately, rather it awaits a notification from the 3rd party service with a response to send to the API client.
- The 3rd party service accepts the forwarded request and returns immediately.
- After the 3rd party service completes processing of the forwarded request, it notifies the API server with a result via a **webhook**.
- The API server receives the notification and sends the result as the final response to the API client.

The API server thus has to act as a *bridge* between the API client and the 3rd party service, and their respective modes (synchronous vs asynchronous) of communication.

Tasks

To complete this assignment, implement the following:

- A web server with these API endpoints:

  `POST /client/:id`

  Accepts a POST request with an `id` path parameter.

  `POST /webhook/:id`

  Accepts a POST request with an `id` path parameter.

  > You may use a web framework such as express.

- When accepting a client request, *do not* return a response immediately; you have to await a webhook notification before sending a response to the client.
- When accepting a webhook request, use the `id` path parameter to match a client that is waiting for a response.

Verify your implementation by invoking the API server as follows (assuming a port of 3000):

```
curl -X POST http://localhost:3000/client/1
curl -X POST http://localhost:3000/client/2
```

You'll now have two clients waiting for a response.

Next, post webhook notifications:

```
curl -X POST -H 'Content-Type: application/json' -d '{ "result": "ok" }'
http://localhost:3000/webhook/2
```

Client with ID 2 should now receive a response (the JSON data) and complete.

```
curl -X POST -H 'Content-Type: application/json' -d '{ "result": "ok" }'
http://localhost:3000/webhook/1
```

Client with ID 1 should now receive a response (the JSON data) and complete.

Delivery

Create a Github repository for your assignment. Ensure that the API server can be started via a script, e.g. `npm start`.

Contact

If you have any questions about the assignment, send an email to: marc.klefter@edument.se.