Queensland University of Technology

CAB432

Assignment 2 Proposal

Scalable Traffic Counting Application

Author:
Luke Busstra [] Joshua Miles[n7176244]

Tutor:

10^{th} October 2016

Contents

1 Introduction

2 Use Cases

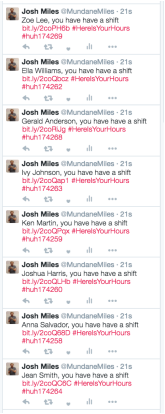
Mashup use cases and services this section should outline explicitly the use cases supported by the mashup (these must be illustrated using screen shots) and the service API calls used to support them. This is a semi-technical description, probably occupying about a page per use case.

2.1 User Case 1

As an employee I would like to have evidence direct from the employer that I have worked for a certain time so that when I go to another position within the same industry I can verify that I have worked in the industry before.

2.1.1 Screenshot

The screenshot shows tweets with the links and names showing evidence they worked where they said they did.



2.1.2 Twitter

Endpoint: `https://api.twitter.com/1.1/statuses/update.json?status=`
The twitter API was used to provide a longterm solution to storing the employees hours for searching in the future, the specific employer made the

tweet and can be simply search when combining the hash of the company with the employee id.

2.1.3 Bitly

Endpoint: `''https://api-ssl.bitly.com/v3/` The Bitly Link shortener was used to shorten the URL to fit into the typical size of a tweet, without it there was a that the crucial information wouldn't fit

2.1.4 Tanda

Endpoint: `https://my.tanda.co/api/v2/schedules?user_ids=theUserID` This will get the employee with the given id so that their schedule can be taken out inorder to post the information inside of the url

2.1.5 Google Calendar

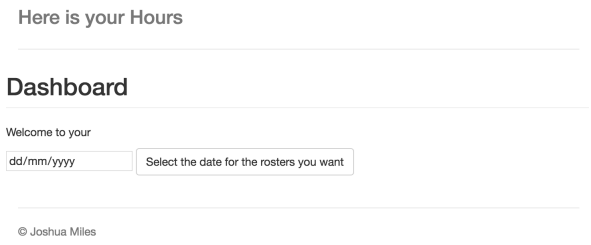
Endpoint: This will get the employee with the given id so that their schedule can be taken out inorder to post the information inside of the url

2.2 Use Case 2

As an employer I want reward my employees in another way by providing evidence to them of working at my esablishment so that they are incentivised to a greater extent than just wage.

2.2.1 Screenshot

The screenshot shows that the employer can select dates where the employer has worked and thean the ability to send the email to employees.



2.2.2 Twitter

Endpoint: `https://api.twitter.com/1.1/statuses/update.json?status=`

The twitter API was used to provide a longterm solution to storing the employees hours for searching in the future, the specific employer made the tweet and can be simply search when combining the hash of the company with the employee id.

2.2.3 Tanda

Endpoint: `https://my.tanda.co/api/v2/schedules?user_ids=theUserID` `https://my.tanda.co/api/v2/locationsThelocationID` `https://my.tanda.co/api/v2/departments/`

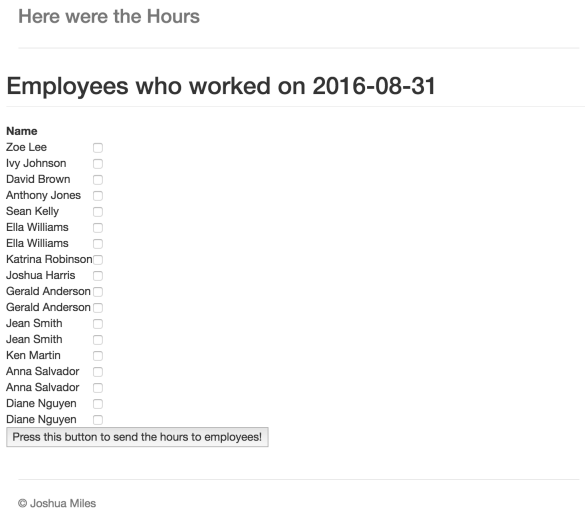
In order to get the specific details about where the employee the following api calls need to be made, first the all of the ids are gotten and than for every employee id needs to get the schedules they are given. The location is then needed by getting the location id using the deparment API which needs the user ID.

2.3 Use Case 3

As a manager I want to be able to seelct who I send their previous hours to so that I can keep the privacy of the employees that do not wish their hours and places they have worked to be public.

2.3.1 Screenshot

The screenshot shows that the manager can make a selection of who they wish to send the hours to using the checkbox.



2.3.2 Tanda

Endpoint: `https://my.tanda.co/api/v2/schedules?user_ids=theUserID`
`https://my.tanda.co/api/v2/shifts?from=Datefromto=To`

To get the employees and when they will be working the tanda shift API is used with the input of the manager to get the date they wish to select to send to the employees

3 Technical Description

Right now the application is relying on tanda input data which was working previously but now only has null inside all of the key identifying traits, this

was not expected at all so if you get a demo account with tanda and get another key it may have pre-populated data.

3.1 Technologies

3.1.1 Express

Express was used as a server to handle the requests and routes throughout the pages. It also was able to handle the view engine of the application which was set to handlebars.

3.1.2 Handlebars

Handlebars what was used to post transfer information between the backend and the front end. It made the transfer relativly simple and easy for making a somewhat dynamically rendered application when used in conjuection with express.

3.1.3 Request

The request node package was used to make the API calls and requests, it is asynchrounous by nature so to use it effectivly one needed to treat it as such. This was not first apparent when initially using the package so the naieve solution was to treat it like it was doing the operations linearly. When the values I was trying to populate started coming back empty even though they were already called, this is when the issues became apparent.

3.1.4 Promises

To solve the issues faced using the the request package promises where ustilised to force something somewhat synchrounous. The solution forged was a chain of Promise.then statements which called took the API request, completed it and when it was resolved it sent a object (shift object) which collected the data throughout the process and the collection of results on the object was used to make the tweet.

There was an opputunity to use the Promise.all function but was not implemented due to time restrictions however the problem that was to be solved with it was to simply collect the names of the users and than send them over to the next page after the employer selects the employees to send

the information to. This was going to be done by getting a final array of promises containing calls to the user api in tanda getting the names using the employee ID given the selection made by the manager which would have been asyncronous. Promise.all would have resolved all of the promises and collected the names of the employees and posted to the handlebars document.

4 Discussion of the use of Docker

http://13.73.202.109:2020/

The trouble the docker image made was the fact that you weren't able to just push a new change to the 'hub' if anything was wrong in combination the node version that was used was not consistant with the actual version being built so it was throwing errors. The code was refactored to meet the specifications of Argon because it was a well renowned node version but this ended up being quite troublesome as I kept getting errors from docker telling me I had an unsupported version.

5 Extensions

In the future, this application would be better with a permanent type of universal id to make it easier identify which employees are who and not just rely on one workplaces employer identifier. In addition, not rely on twitter, google and bitly to store data needed to verify the employees experience would mean if one broke it would be able to relocate the API use to another service. On a minor note, being able to use the Promise.all function would improve the usablity on the managers side, as was noted previously.