

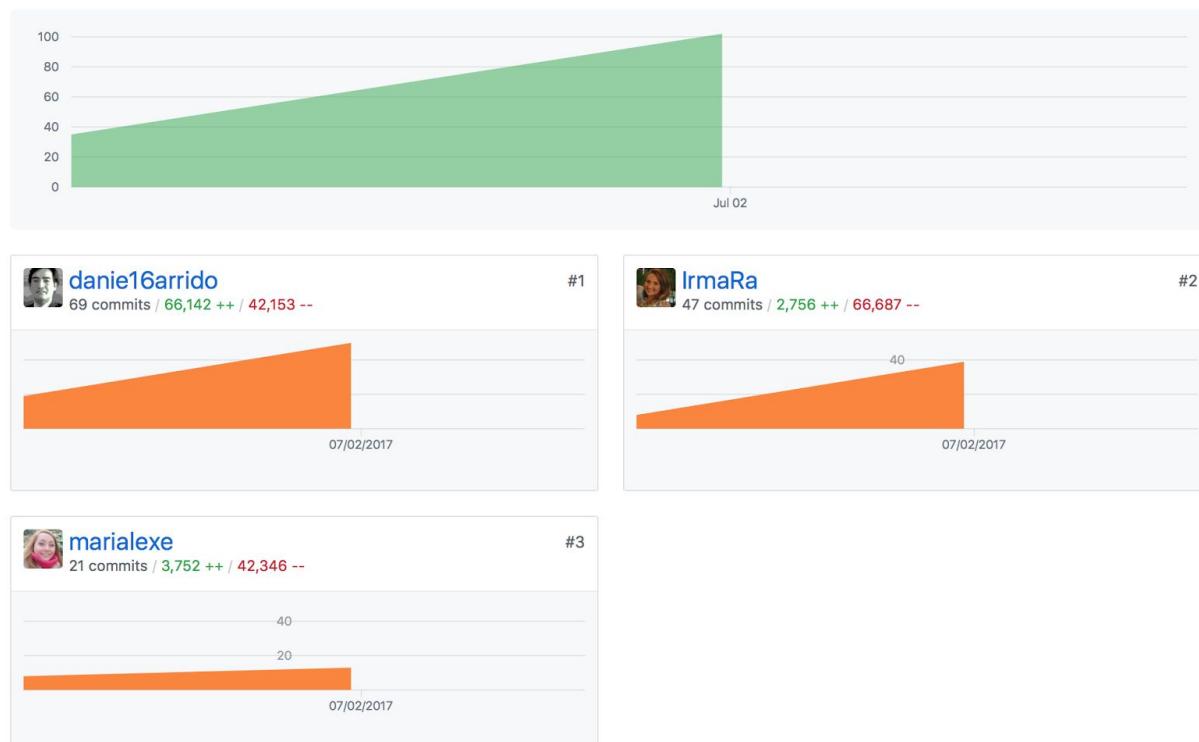
PDA Software Development: Project Unit part 1

P 1 Take a screenshot of the contributor's page on Github from your group project to show the team you worked with.

Jun 25, 2017 – Jul 7, 2017

Contributions: **Commits** ▾

Contributions to master, excluding merge commits



P2 Write a project brief which outlines the requirements of the program(project group)

Brief:

Festivals fans want to be able to view relevant festivals events on a dashboard.
Use and create a new API to display information about festivals,
news and travel information for events.

MVP

Display upcoming events on a map
Display results and ranking of best/Festivals
Allow users to add festivals to a favourites list.

P 3 Provide a screenshot of the planning you completed during your group project, e.g. Trello MOSCOW board.

The image displays two Trello boards:

MoSCoW Board:

- Must:**
 - Display a map with all the festivals marked
 - Display the most popular festivals(image, name, date, location) and button to add to favourites.
 - To be able to add a festival to favourite festivals list
- Should:**
 - Hover over festivals to display all details.
 - Add action to the eventListener of the search buttons
 - Save favourite festivals
 - Apply filter by type
 - Apply filter by location and time
- Could:**
 - summary description
- Would:**
 - to have a chart

Scrum Board Festivals Finder:

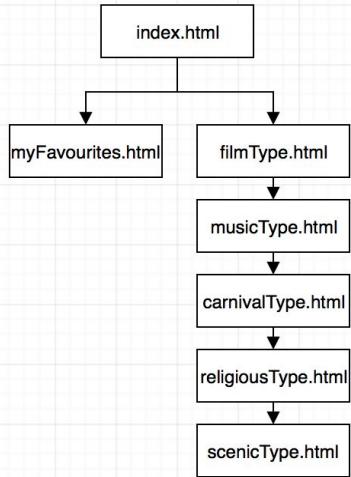
- Backlog:**
 - Colour markers by category
 - work out how to implement css files in bundle and not have them all listed in index.html header
 - decide how to measure popularity. Do we add extra info in the api?
 - add a function in festival_query.js to retrieve all the festivals happening in the next 30 days (use the current date as base)
- Stories:**
 - As a programmer I want to have a git repo so I can develop code.
 - As a programmer I need data so I can display data.
 - As a programmer I want to have an API so I can develop the app.
 - As a user I want to see the top 6 festivals (popularity) so I can choose.
- To DO:**
 - create seeds
 - create a db
 - create models
 - setup festival_query.js file (M)
 - add function to festival_query to retrieve the most popular festivals. (6) (M)
 - update seeds with popularity (I)
- Doing:**
 - setup git repo (M,D)
 - create folder structure (M,D)
 - create node.js (M,D)
 - create base skeleton for the project (M,D)
 - create development branch(M,D)
 - setup routes (D)
 - create base skeleton for the project (M,D)
 - create develop branch (M,D)
 - map wrapper (I)
 - css file with map settings (I)
 - div in html (I)
- Done:**
 - update the festival_query.js file create a function findByType, (M)
 - update the festival_query.js file create a function findByCountry, (M)
 - create landing page (D)
 - xmlhttprequest to connect to the api
 - move map related functions to MapWrapper. (I)
 - create functions which adds markers (I)
 - update the festival_query.js file with a function findByRating, (M)
 - create more than one css file. Make one for buttons/slider/hover over ,etc etc
 - add a function in festival_query to retrieve the most popular festivals, (M)
 - create new user database (D)
 - create buttons for each festival type in <div search>. (M)
 - add images to seeds.js file (I)
 - fix queries for type, country, ratings they are sending responding with only one item instead of an array of items (D)
 - show full details of the festival once user chooses one, (I)
- Stories Done:**
 - As a programmer I want a favourites api so user can store its favourite festivals.
 - As a user I want to have a landing page, so I can search for festivals.
 - As a user I want to see a map with events so I can choose one.
 - As a user I want to be able to save my favourite festivals.

P4 Write an acceptance criteria and test plan.

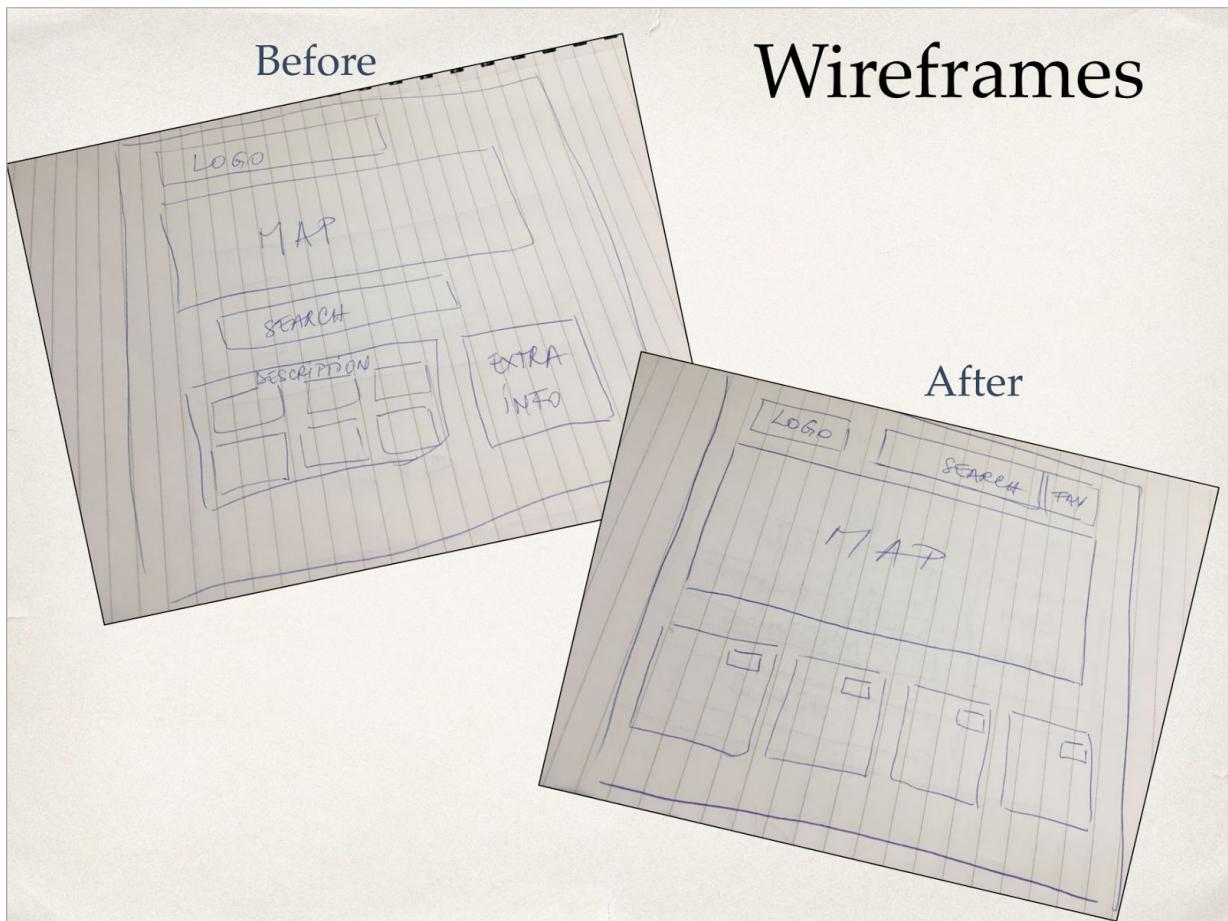
Acceptance Criteria	Expected Output/Result	Pass/Fail
User is able to see all festivals in a map	When opening the landing page a map must be displayed with markers to all the festivals happening from now on.	Pass ✓
User is able to filter festivals by type	When user clicks on one of the menu categories the data display area lists all the festivals of the menu type	Pass ✓
User is able to get more info of a festival	When user hovers over the festival component it automatically displays date about the data and more info of selected festival	Pass ✓
User is able to save festivals to favourites	When user hovers over the festival component it automatically displays a button to save said festival being able to later click on my Favourites button and see all the saved festivals	Pass ✓
User is able to delete festival from favourites	When user is his/her favourites area he/she should be able see the saved festivals and click on the delete button and have that item deleted from favourites.	Pass ✓

P 5 Create a sitemap for the project

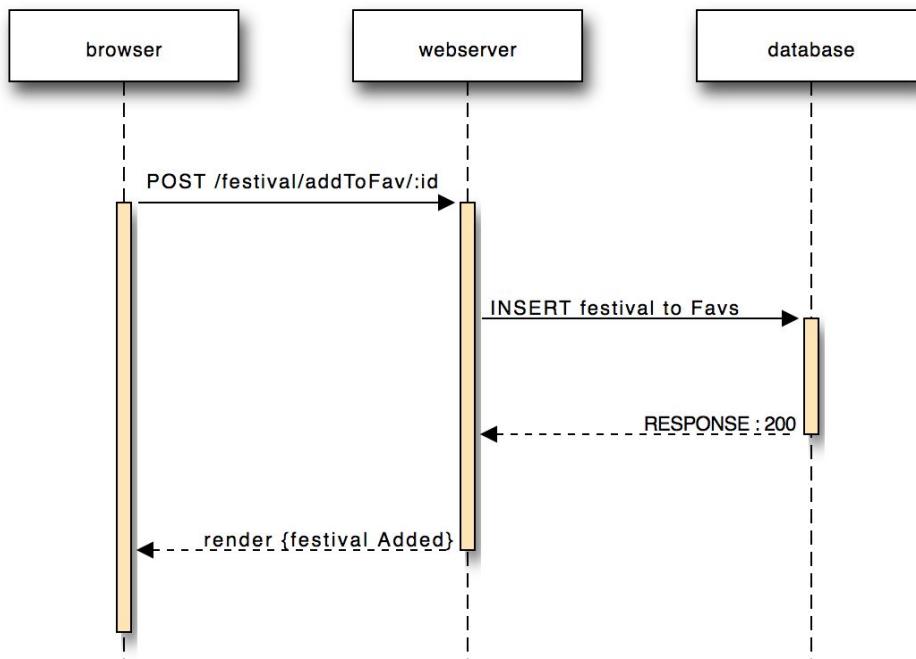
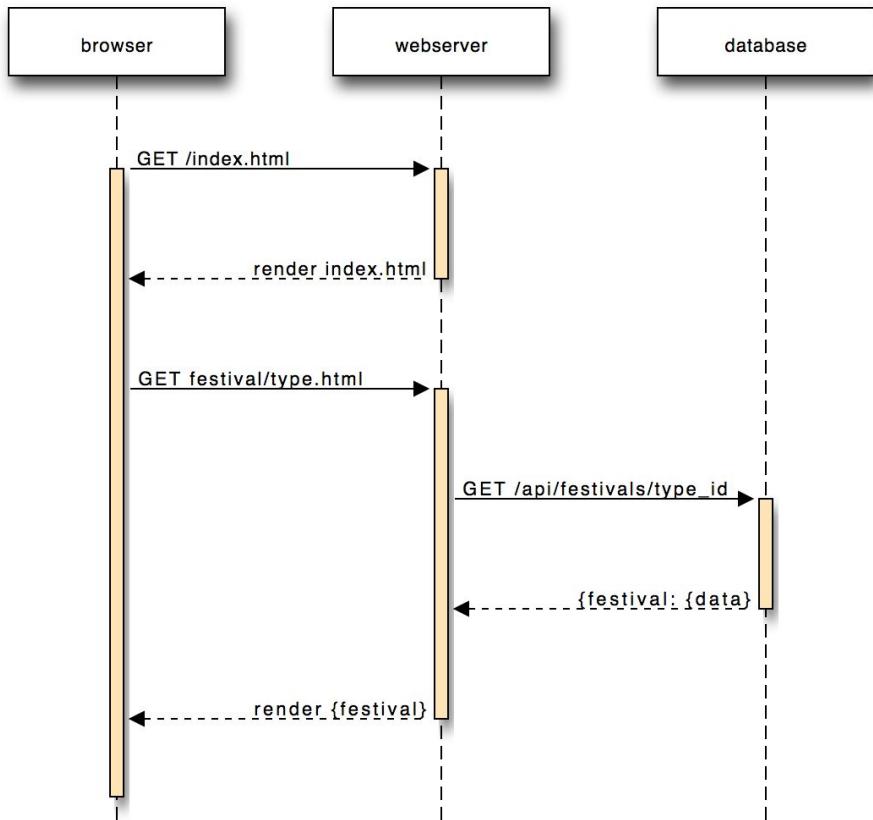
Festival Finder SiteMap



P 6 Produce two wire frame designs showing user interface and interactions, what you have hoped it would look like.

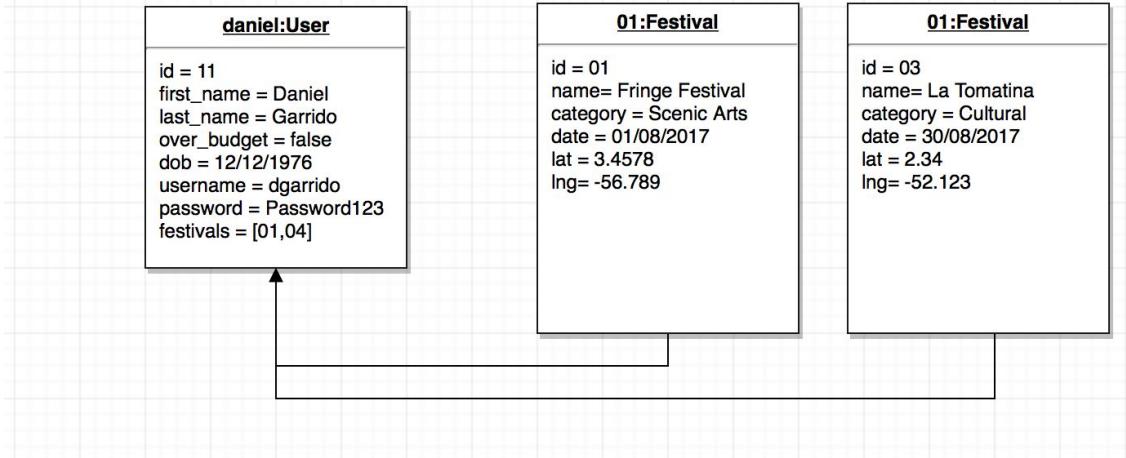


P 6 Produce two system interaction diagrams(sequence and/or collaboration diagrams)



P 8 Produce object diagrams

Object_diagram



P 9 Select two algorithms you have written. Take a screenshot of each and write a short statement on why you have chosen to use those algorithms.

```
Game.java x

47     pHand.addIcard(this.deck.next());
48 }
49 }
50 }
51
52 public int handPlayerSum(Player player){
53     ArrayList<ICard> pHand = player.getHand().getCards();
54     int sum = 0;
55     int cardRank;
56     int aces = 0;
57     for (ICard card: pHand){
58         cardRank = card.getRank();
59         if (cardRank == 1){ //Ace card
60             aces++;
61             sum += 11;
62         } else if (cardRank > 10){ //face card
63             sum += 10;
64         } else {
65             sum += cardRank;
66         }
67     }
68     while (sum > 21 && aces > 0){
69         sum -= 10;
70         aces--;
71     }
72     return sum;
73 }
74
75 private int checkBlackJackState(Player player){
76     int playerHandSum = handPlayerSum(player);
77     if( playerHandSum == 21){
```

handPlayerSum() is a function that describes an algorithm that sums the cards in the player's hand by initially adding the value of the card and then checking if the hand has aces and the total value is more than 21 then and only then the total hand's value will be subtracted 10 for each existing ace.

```

78     |     this.addMarker( loc, crime.category );
79     |     loc = "";
80   },
81   |     this.updateCategories()
82 },
83
84   circlePath: function(center, radius, points){
85     var newPolygon=[];
86     p=360/points,d=0;
87     for(var i=0;i<points;++i,d+=p){
88       newPolygon.push(google.maps.geometry.spherical.computeOffset(center,radius,d));
89     }
90     return newPolygon;
91   },
92   createRequestAddress: function( polygon ) {
93     var requestAddress = "https://data.police.uk/api/crimes-street/" + this.selectedCategory + "?";
94     requestAddress += "poly=";

```

CirclePath() is a function that helps to generate a polygon that simulate a circle based on a center(lat, lng in googlemaps API), radius(m) and the number of points needed.
I needed it to query my API with a polygon instead of an actual circle.

P 10 Take a screenshot of an example of pseudocode for a function.

```

1 //Blackjack game logic
2 //ToDo|
3 Create a deck of cards
4 Shuffle a deck of cards
5 Create user's Hand
6 Create dealer's Hand
7 Deal two cards to each hand
8 Check if dealer has got blackjack (if so, game ends)
9 Check if user has got blackjack (if so, game ends)
10 User draws cards (if user goes over 21, game ends)
11 Dealer draws cards
12 Check for winner

```

P 11 Take a screenshot of one of your projects where you have worked alone and attach the Github link.



CrimeSpottinghttps

An app to track crimes within a chosen area and filter it by date and type of crime.

- GoogleMaps API
 - data.police.uk API

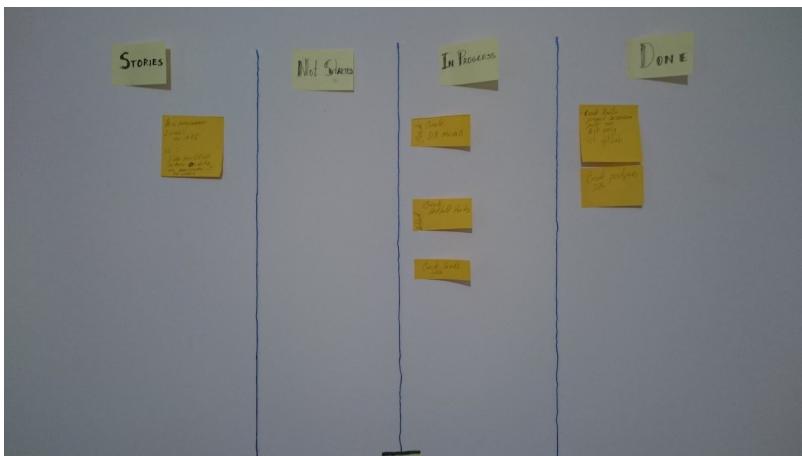
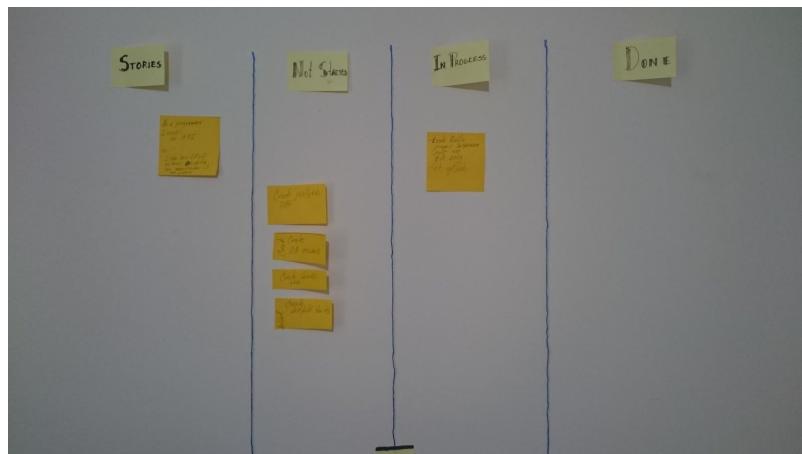
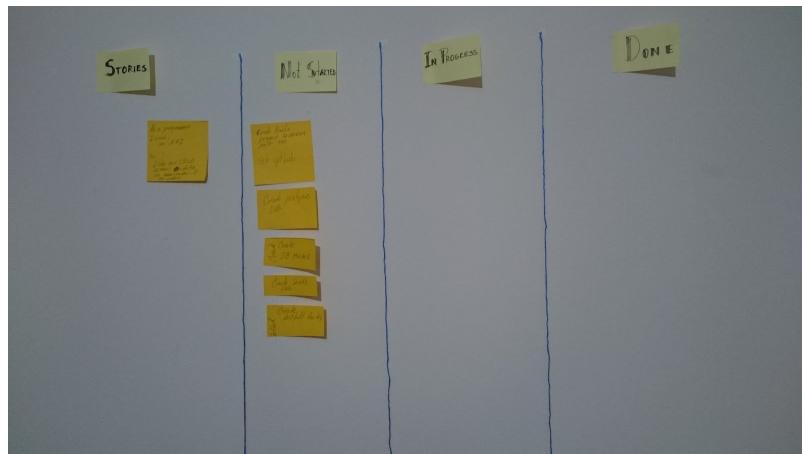
Key learning to practise and/or revise:

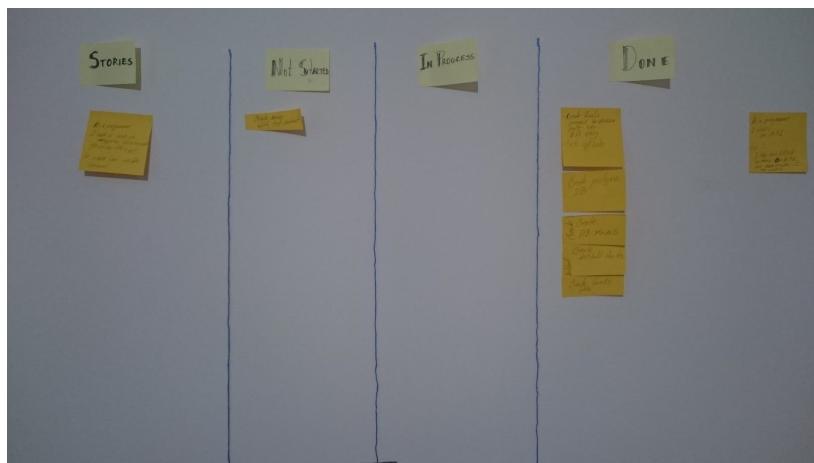
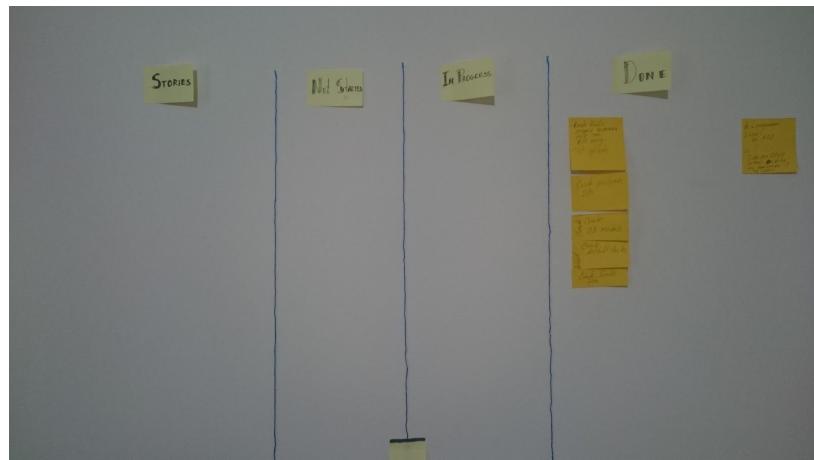
- Manipulating the DOM - selecting/creating/appending elements
 - Using event listeners to make our apps interactive
 - Making an HTTP request to retrieve some data and then using it

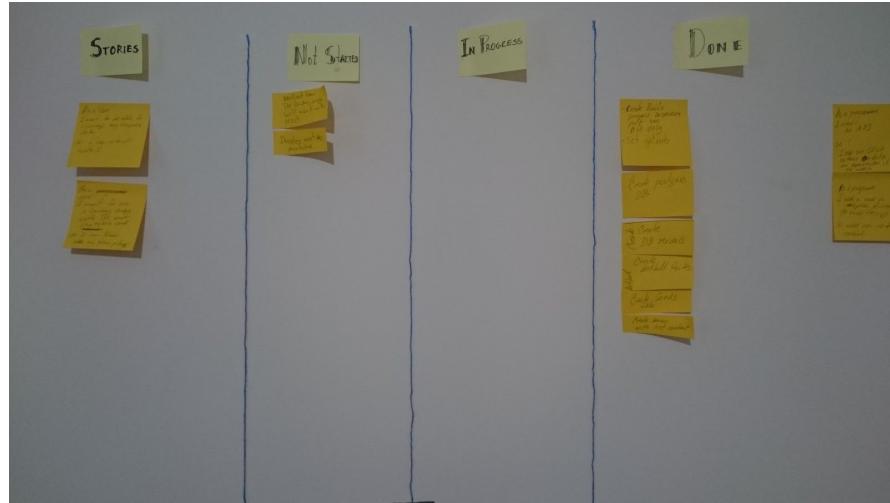
<https://github.com/danie16arrido/CrimeSpotting>

P12 Take screenshots or photos of your planning and the different stages of development to show changes

Agile(Scrum)







P 13 Show user input being processed according to design requirements. Take a screenshot of:

- The user inputting something into your program
User selects a filter(bicycle-theft)



- The user input being saved or used in some way

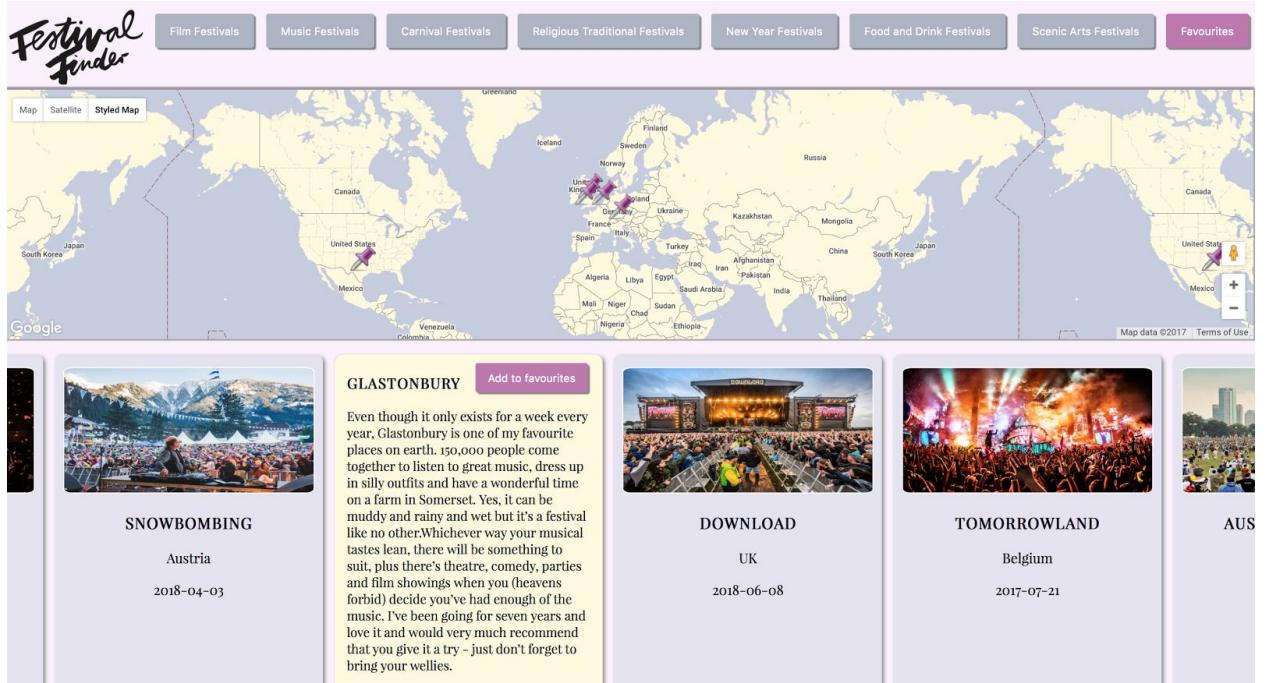
User selection saved and query send to the API which populates the map with the data.



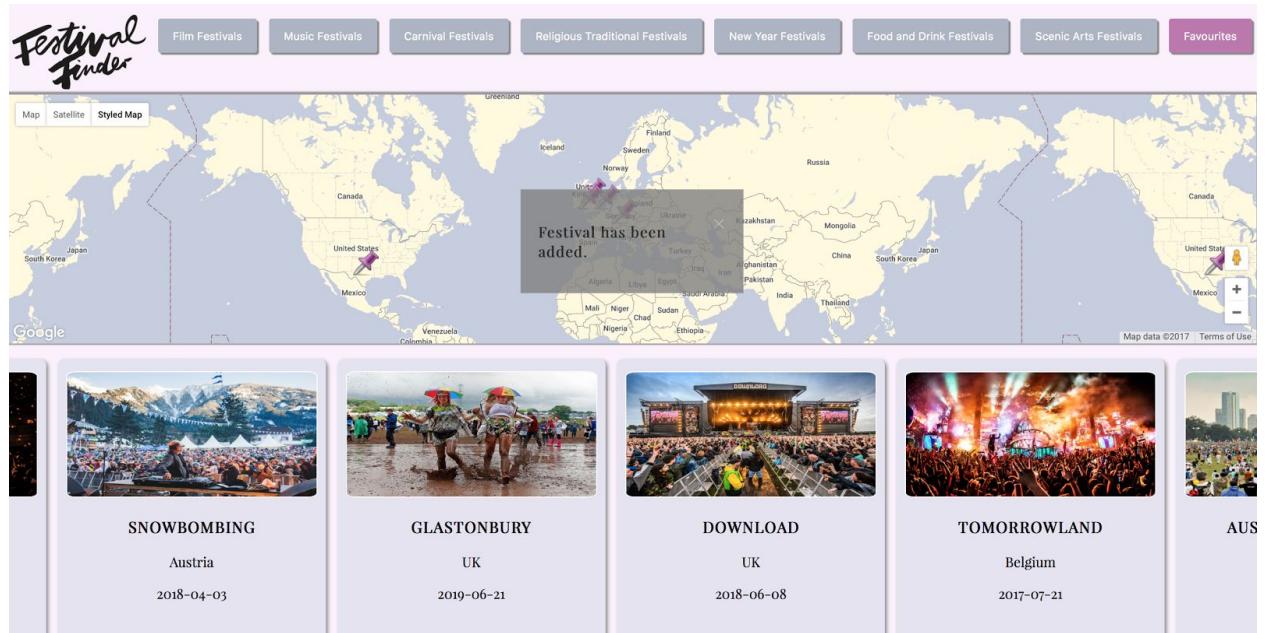
P 14 Show an interaction with data persistence. Take a screenshot of:

- Data being inputted into your program

User saving a festival



- Confirmation of the data being saved



P 15 Show the correct output of results and feedback to user. Take a screenshot of:

- The user requesting information or an action to be performed
User requesting data about league with ID=3
- The user request being processed correctly and demonstrated in the program
API responding with the data for league with ID=3

Tournament Manager		GET http://localhost:9100/api/v1/leagues/3				Send	200 OK	TIME 14.5 ms	SIZE 104 B	
No Environment	Cookies	Body	Auth	Query	Header	Docs	Preview	Header 12	Cookie	Timeline
Filter <input type="checkbox"/> Goals <input type="checkbox"/> Cards <input type="checkbox"/> Matches <input type="checkbox"/> Referees <input type="checkbox"/> Locations <input type="checkbox"/> Players <input type="checkbox"/> Teams <input checked="" type="checkbox"/> Leagues								<pre> 1 * { 2 "id": 3, 3 "name": "FA Cup", 4 "created_at": "2017-07-22T21:09:14.699Z", 5 "updated_at": "2017-07-22T21:09:14.699Z" 6 } </pre>		
DEL DESTROY /api/v1/leagues/:id PUT UPDATE /api/v1/leagues/:id POST CREATE /api/v1/leagues GET SHOW /api/v1/leagues/:id GET INDEX /api/v1/leagues		 Select a body type from above								

P 16 Show an API being used within your program. Take a screenshot of:

- The code that uses or implements the API
- The API being used by the program whilst running

```
1 import axios from 'axios'
2 const apiServer = "http://localhost:9100/api/v1"
3 const leagueID = 11
4 const getApiLeagueName = leagueID => {
5   return dispatch => {
6     return axios.get( apiServer + "/leagues/" + leagueID)
7       .then( response => {
8         dispatch(
9           {
10             type: "API_LEAGUE_NAME",
11             league_name: response.data.name
12           }
13         )
14       }
15     )
16   }
17 }
```

User requesting details for Chelsea team

FA Cup

The screenshot shows a web interface for the FA Cup. At the top, there are three tabs: 'MATCHES' (selected), 'TABLE', and 'STATS'. Below the tabs, a table lists two matches for the Chelsea team:

Date	Time	Home	Away	Venue
2017-07-23	10:30	Chelsea	Liverpool	Edinburgh Uni B
2017-07-23	15:30	Sao Paolo	Boca Juniors	Holyroot

User getting results for Chelsea team after the front end server querying the API server and rendering data.

FA Cup

The screenshot shows a web interface for the FA Cup. At the top, there are three tabs: 'MATCHES' (selected), 'TABLE', and 'STATS'. Below the tabs, a section titled 'Chelsea s Players' displays three player cards:

- Daniel Garrido**
Height: 1.81m Weight: 82.33Kg Age: 40
- Leo Messi**
Height: 1.03m Weight: 60.8Kg Age: 33
- Thibaut Courtois**
Height: 1.83m Weight: 78.8Kg Age: 34

P 17 Produce a bug tracking report

Description	Status	Solution	Status
User can not get results when filtering by “New Year” category	FAILED	In the database field name changed from “newyear” to “new_year” for easy parsing.	PASSED
Map not being updated with the selected category	FAILED	Added a method that updates the map with new results when rendering.	PASSED
User can not save festivals	FAILED	Added user ID to the API query.	PASSED
Data of festival not being displayed on hover over.	FAILED	#festival-container class edited in main.css file to be able to handle hover over.	PASSED
Logo image link not sending back user to home page	FAILED	Added link to the homepage to the logo image.	PASSED

P18 Demonstrate testing in your program . Take screenshots of:

- Example of test code



```
1 var assert = require('assert');
2 var Record = require('../record.js');
3 var RecordStore = require('../record_store.js');
4
5 describe("RecordStore", function(){
6   var record;
7   var recordStore;
8
9   beforeEach(function(){
10     record = new Record("test", "x&y", "chillout", 9.99);
11     record1 = new Record("test2", "x&y&z", "chillout-more", 19.99);
12     recordStore = new RecordStore("dans record store", "Gotham city");
13   });
14
15   it("should have a name", function(){
16     | assert.strictEqual(recordStore.name, "dans record store");
17   });
18
19   it("should have a city", function () {
20     | assert.strictEqual(recordStore.city, "Gotham city")
21   });
22
23   it("should have an inventory", function(){
24     | assert.deepEqual(recordStore.inventory, []);
25     | assert.strictEqual(recordStore.inventory.length, 0);
26   });
27
28
29   it("should have a balance 0 at init", function(){
30     | assert.strictEqual(recordStore.balance, 0 );
31   });
32
33   it("should be able to add records to inventory", function(){
34     | recordStore.add(record);
35     | assert.strictEqual(recordStore.inventory.length, 1);
36   });
37
38   it("should be able to list the inventory", function(){
39     | recordStore.add(record);
40     | var result = "1. test, x&y, chillout, £9.99"
41     | assert.strictEqual(recordStore.listInventory(), result);
42   });
43
44   it("should list multipley items ", function(){
45     | recordStore.add(record);
46     | recordStore.add(record1);
47     | var result = "1. test, x&y, chillout, £9.99\n2. test2, x&y&z, chillout-more, £19.99";
48     | assert.strictEqual(recordStore.listInventory(), result);
49   });
50
51   it("should be able to remove a record from inventory", function(){
52     | recordStore.add(record);
53     | recordStore.remove(record);
54     | assert.strictEqual(recordStore.inventory.length, 0);
55   });
56
57   it("should be able to modify balance", function(){
58     | recordStore.modifyBalance(10);
59     | assert.strictEqual(recordStore.balance, 10);
60   });
61
62   it("should be able to sell a record", function(){
63     | recordStore.add(record);
64     | recordStore.sell(record);
65     | assert.strictEqual(recordStore.balance, record.price);
66     | assert.strictEqual(recordStore.inventory.length, 0);
67   });
68
69   it("should be able to sell a record that exists", function(){
70     | assert.strictEqual( false ,recordStore.sell(record));
71   });
72
73   it("should be able to print finance data", function(){
74     | var result = "Balance: £0 Inventory: £0";
75     | assert.strictEqual(recordStore.financeData(), result);
76   });
77
78   it("should get the inventory value ", function(){
79     | recordStore.add(record);
80     | assert.strictEqual(recordStore.inventoryValue(), record.price);
81   });
82
83   it("should get the inventory value with multiple items", function(){
84     | recordStore.add(record);
85     | recordStore.add(record1);
86     | assert.strictEqual(recordStore.inventoryValue(), 29.98);
87   });
88 })
89
```

- The test code failing to pass

```

28      ...
29      it("should have a balance 0 at init", function(){
30          | assert.strictEqual(recordStore.balance, 0 );
31      });
32
33      it("should be able to add records to inventory", function(){
34          | assert.strictEqual(recordStore.inventory.length, 1);
35      });
36
37      it("should be able to list the inventory", function(){
38          recordStore.add(record);
39          var result = "1. test, x&y, chillout, £9.99"
40          assert.strictEqual(recordStore.listInventory(), result);
41      });
42
43      it("should list multipley items ", function(){
44          recordStore.add(record);
45          recordStore.add(record1);
46          var result = "1. test, x&y, chillout, £9.99\n2. test2, x&y&z, chillout-more, £19.99";
47          assert.strictEqual(recordStore.listInventory(), result);
48      });
49

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: z:

```

18 passing (20ms)
1 failing

1) RecordStore should be able to add records to inventory:
   AssertionError: 0 === 1
   + expected - actual

   -0
   +1

  at Context.<anonymous> (specs/record_store_spec.js:34:12)

npm ERR! Test failed. See above for more details.
→ recordStorejs git:(master) ✘

```

- Example of the test code once errors have been corrected

```

28      ...
29      it("should have a balance 0 at init", function(){
30          | assert.strictEqual(recordStore.balance, 0 );
31      });
32
33      it("should be able to add records to inventory", function(){
34          recordStore.add(record);
35          assert.strictEqual(recordStore.inventory.length, 1);
36      });
37
38      it("should be able to list the inventory", function(){
39          recordStore.add(record);
40          var result = "1. test, x&y, chillout, £9.99"
41          assert.strictEqual(recordStore.listInventory(), result);
42      });
43

```

- The test code passing

```
32
33     it("should be able to add records to inventory", function(){
34         recordStore.add(record);
35         assert.strictEqual(recordStore.inventory.length, 1);
36     });
37
38     it("should be able to list the inventory", function(){
39         recordStore.add(record);
40         var result = "1. test, x&y, chillout, £9.99"
41         assert.strictEqual(recordStore.listInventory(), result);
42     });
43
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

> mocha specs/

```
record
  ✓ should have an artist
  ✓ should have a title
  ✓ should have a genre
  ✓ should have a price
  ✓ should have print a record summary

RecordStore
  ✓ should have a name
  ✓ should have a city
  ✓ should have an inventory
  ✓ should have a balance 0 at init
  ✓ should be able to add records to inventory
  ✓ should be able to list the inventory
  ✓ should list multipley items
  ✓ should be able to remove a record from inventory
  ✓ should be able to modify balance
  ✓ should be able to sell a record
  ✓ should be able to sell a record that exists
  ✓ should be able to print finance data
  ✓ should get the inventory value
  ✓ should get the inventory value with multiple items
```

19 passing (15ms)

→ recordStorejs git:(master) ✘