Flood feeder

[Update 16 Feb - (end of the day): demo-able app, demonstrates basic functionality! We won a grant from Nominet Trust (thanks!) and a generous hosting package from Google (thanks!). Work will continue ASAP to bring it to alpha/beta and open it up with documentation to other developers to contribute.]

Team:

- Dan Smith (@danpaulsmith)
- Brendan Quinn (@brendanquinn)
- Greg Nwosu (@greg_nwosu)

Demo: http://floodfeeder.cluefulmedia.com/
Github: https://github.com/gregnwosu/floodfeeder

If you would like to contribute to this project then get in contact with any of us/hang out in our Google Hangout etc.

Idea

A tool to aggregate flood-related datasets (spatial or non-spatial), and spit them out in JSON/GeoJSON/TopoJSON (a friendly and easy to use data format for developers to work with!). This makes the development process quicker, the data easier to grasp and increase the chances of apps and visualisations being developed.

It should offer geographic granularity, so you can aggregate data for a country, a ward, a town or any other type of geographic boundary.

The tool is in the form of a single page web page that provides a list of geographic layers and data feeds to include or exclude from your agreggated feed, which - (if they contain geographic data such as point locations [cellphone masts, river guages] or shapes [flood areas, cellphone coverages]) - can be previewed live on a map and be interacted with.

Once finished building your feed, you can choose to export it as JSON/GeoJSON (or other formats).

Aggregating the data

1. The original idea posed, was for all feeds to be returned as a unified feed

```
export = {
   geoJSON:{
           …        // everything in here
   }
}
```

currently we have a list of the form

```
[

 { "type": "Feature",
  "geometry": {
    "type": "Point",
      "coordinates": [lon, lat, alt]
   },
  "properties": {
    "title": ad,
    "description": "<p>"+msg+"</p>",
   “source” : “shoothill”,
   “extra_json” : {
           “blah1”: “”,
           “blah2” : “”,
   }
˜}

]
```

2. But if the feeds are a mixture of spatial and non-spatial data, they won’t fit the geoJSON model if merged, so would have to be included alongside each other:

```
export = {
   geoJson:{}, // spatial (multiple spatial feeds as a FeatureCollection of points/shapes...)
   json: {
            feed3:{}, // non-spatial
            feed4:{} // non-spatial
   }
}
```

3. Perhaps the user just wants each of the feeds individually exported?

```
{
   feed1:{}, // spatial = geoJSON (FeatureCollection probably)
   feed2:{}, // spatial = geoJSON (FeatureCollection probably)
   feed3:{}, // non-spatial, perhaps timestamped data (Tweets etc)
   feed4:{}, // non-spatial ...
}
```

As a front-end dev, I can say that **#3** is more useful, to have them individually. That way I can combine them/add them as individual layers to my app if I want to. But then it might not hurt to have a "merge" spatial feeds as a checkbox before exporting? Simply because GeoJSON supports FeatureCollections.

What's been used

- Ordnance Surveys boundary data (to enable granularity and boundary restriction), converted and compressed to JSON & TopoJSON
- Shoothill's API - serving up Environment Agency's flood warning/alert data.
- OpenStreetMap's map tiles & Leaflet.js to create the map & its markers
- ~~D3.js to overlay boundary data on to the map~~
- Fake cellphone mast data (demo purposes)
- Fake export data (demo purposes)
- Bootstrap for the UI
- http://mapit.mysociety.org/ to auto-detect areas

How it works

1. Map loads Open Street Map tiles
2. Geographic areas available to include as overlays on map as checkboxes under the "Geography" list (an area could then be selected to focus & aggregate data for)
3. Available data sources are listed on page with checkboxes next to them
4. On clicking a checkbox, the data is retrieved, cached locally, if contains geodata it's added to the map (markers, areas...)
5. User can export the data as JSON (no geo data) or GeoJSON/TopoJSON.

Demo walkthrough

1. Load the app
2. OSM map should load
3. Check “countries” to overlay boundary data (zoom out a bit to see extent)
4. Enter “Staines” into the search box to zoom to area
5. Check “Flood alerts” or “Flood warnings” to retrieve data from Shoothill’s API and add it to the map.
6. Zoom out 2 levels
7. Check “Cellphone masts” to add some fake mast locations to the map.
8. Click “Export” to simulate an export of the aggregated feed.

Next steps

1. Organise a face-to-face/google hangout soon to discuss how we can distribute work load to get the app to alpha level.
2. ~~Improve code and remove dummy demo stuff~~
3. Decide on/enable aggregation of feeds (backend?) and thus the ability to export aggregated data
4. ~~Enable geographic granularity (wards/districts/what might be the most useful? maybe a “enter your location” > lat/long > you are contained within?)~~
5. Enable flood warning/alerts data retrieval (Shoothill’s API) for not just Staines!
6. Add a useful README &/ Wiki on github
7. Add more datasets & overlays
8. Add JeniT’s TopoJSON for granularity - https://github.com/JeniT/boundaries
9. Add Gareth’s Twitter scraping layer - https://twitter.com/godawful
10. Find @floodvolunteers API/locations?
11. ~~Find cellphone masts locations?~~
12. Include “Feedback” tab for app
13. Develop one or more visualisations using Flood Feeder
14. Submit app to CodeForResilience (http://www.codeforresilience.org/)

Front-end to-do-list

1. ~~Get markers working on the map again~~
2. Tidy up auto-detection
3. ~~Change to new API URL structure~~
4.
5. Test on chrome - tooltips hiding behind map
6. Use layer groups in leaflet for the layer
7. Add browser-location detection

8. Add GA to page
~~9. Add feedback plugin (https://www.uservoice.com/)~~
10. Display features from layers as pretty printed JSON (for previewing information)
11. BETA (Allow users to import their own geoJSON)
~~12. Buy floodfeeder.co.uk~~

Backend to-do-list

1. Move to Google's hosting platform
2. Decide on how to query spatial datasets given bounding box values
3. Look at installing some sort of geo querying package? (a la http://geodjango.org/ & http://www.mkgeomatics.com/wordpress/?p=500)
4. Make sure all information from feeds is incldued
5. Include source as a field in each geojson record

Resources

GeoJSON spec
http://geojson.org/geojson-spec.html

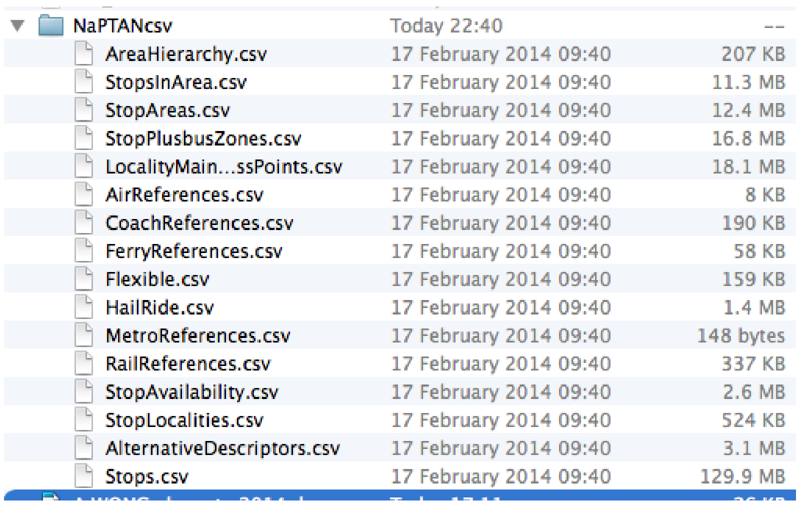
GeoJSON lint & validator
http://geojsonlint.com/

Online file converter to/from GeoJSON with POST API
http://ogre.adc4gis.com/

Related datasets & APIs (import.io)
https://docs.google.com/spreadsheet/ccc?key=0AkH2kR2ghqxWdElYZU9KYTFuWnZhZW9nTFVGazNDbVE

Shoothills API
http://dbec32afb59243e0a83d0216b56eccce.cloudapp.net/api/floods/1
& doc
http://apifa.shoothill.com/help

Transport dataset (NaPTAN) (Bus stops, train stops etc)
http://data.gov.uk/dataset/naptan/resource/e3d0c00c-abb7-4159-b512-5e3ac394780a
I've downloaded the NaPTAN set, it contains Northing/Eastings for the following ("Stops.csv" is a comprehensive list of all transportation "stops", containing 420,000 rows of bus stops, rail stations, tube stations, and their different entrances etc...):



NaPTANcsv	Today 22:40	--
AreaHierarchy.csv	17 February 2014 09:40	207 KB
StopsInArea.csv	17 February 2014 09:40	11.3 MB
StopAreas.csv	17 February 2014 09:40	12.4 MB
StopPlusbusZones.csv	17 February 2014 09:40	16.8 MB
LocalityMain...ssPoints.csv	17 February 2014 09:40	18.1 MB
AirReferences.csv	17 February 2014 09:40	8 KB
CoachReferences.csv	17 February 2014 09:40	190 KB
FerryReferences.csv	17 February 2014 09:40	58 KB
Flexible.csv	17 February 2014 09:40	159 KB
HailRide.csv	17 February 2014 09:40	1.4 MB
MetroReferences.csv	17 February 2014 09:40	148 bytes
RailReferences.csv	17 February 2014 09:40	337 KB
StopAvailability.csv	17 February 2014 09:40	2.6 MB
StopLocalities.csv	17 February 2014 09:40	524 KB
AlternativeDescriptors.csv	17 February 2014 09:40	3.1 MB
Stops.csv	17 February 2014 09:40	129.9 MB

I've also found this Guardian Data blog fusion table with the stops as lat/longs (52MB csv):
https://www.google.com/fusiontables/data?docid=15n7Rpi190vjrPcmbiT_hRcZ9JbXE8a_I1euHyg&pli=1#rows:id=1

Transport API (DGUK)
http://transport.data.gov.uk/

TransportAPI (Placr)
https://developer.transportapi.com/documentation
(I've signed us up to this, ask me for login - 1,000 calls a day, but they might waive us if we explain its short term for a flooding relief effort).

Thunderforest tile layers (Transport, Outdoors, Landscape)
http://www.thunderforest.com/terms/

Network rail feed (Live train positions)
http://www.networkrail.co.uk/data-feeds/
(I have just signed us up to use this, ask me for login details - account is now "Active")

List of related open datasets
https://github.com/theodi/shared/wiki/Finding-Open-Data

Cellphone masts (with grid references)
https://docs.google.com/file/d/0BxTmUMstpZKwdkIYM2pZZDNGVXc/edit
https://drive.google.com/folderview?id=0BxTmUMstpZKwcmFoNGFPY1RHb1U&usp=sharing

Cell phone masts (lat/longs - converted by George v D.) (10MB/144,000 rows)
https://www.google.com/fusiontables/DataSource?docid=1Snw53xYaBu4dZVyMl-nFTsjdsfSJe3

KDddpCYz4#rows:id=1

GeoDjango
http://geodjango.org/

Twitter Geo API
https://dev.twitter.com/docs/api/1/get/geo/search

ONS Census API
http://www.ons.gov.uk/ons/about-ons/who-ons-are/programmes-and-projects/enhancing-access/ons-api/index.html
I’ve signed up to this API, looks like we can query the census per ward/district and get back JSON. Could be a nice feature to hover over an area that’s got flood alerts/warnings to see what the gist is of the population there.

flood volunteers
transport use
tweets

GeoDjango formatting blog post
http://www.mkgeomatics.com/wordpress/?p=500

Constructing the GeoJSON object

Types of GeoJSON objects

Each GeoJSON structure is either a

- “FeatureCollection” (a set of Features)
- or a "GeometryCollection” (a collection of pure geometries with no metadata)
- A "Feature" which must contain a Geometry and a set of “properties”.
- Geometry can be a:
    - "Point" (one position)
    - "MultiPoint” (an array of positions)
    - "LineString" (an array of positions)
    - "MultiLineString” (an array of arrays of positions)
    - "Polygon" (an array of arrays of positions)
    - "MultiPolygon” (a multidimensional array of positions)

Every GeoJSON record can have extra metadata to enhance a record.

Collection of points (flood warnings/cell towers etc)

```
{
  "type":"FeatureCollection",
  "features":[
    {
      "type":"Feature",
      "geometry":{
        "type":"Point",
        "coordinates":[
          102.0,
          0.5
        ]
      },
      "properties":{
        "operator":"O2",
        "power":"12db"
      }
    },
    {
      "type":"Feature",
      "geometry":{
        "type":"Point",
        "coordinates":[
          103.0,
          0.4
        ]
      },
      "properties":{
        "operator":"Orange",
        "power":"15db"
      }
    },
    {
      "type":"Feature",
      "geometry":{
        "type":"Point",
        "coordinates":[
          104.0,
          0.7
        ]
      },
      "properties":{
        "operator":"EE",
        "power":"9db"
      }
    }
  ]
}
```

<u>CRS (Coordinate reference system)</u>

Is a CRS value necessary?

```
"crs": {
         "type": "name",
         "properties": {
                  "name": "urn:ogc:def:crs:OGC:1.3:CRS84"
         }
}
```

<u>GeoJSON bbox (Bounding box)</u>

Is a bounding box necessary?

```
{ "type": "Feature",
   "bbox": [-180.0, -90.0, 180.0, 90.0],
   "geometry": {
     "type": "Polygon",
     "coordinates": [[
       [-180.0, 10.0], [20.0, 90.0], [180.0, -5.0], [-30.0, -90.0]
       ]]
     }
   ...
   }
```

Hackday notes

- Arrive at site
- Choose an area if required
- Select from a categorised list of data feeds (1 spatial dataset + * other informative datasets)
- View
- Export as JSON/geoJSON
- Develop app / visualisation

Demo: http://floodfeeder.cluefulmedia.com/

Github: https://github.com/gregnwosu/floodfeeder

Data feeds available:

Shoothill API: http://dbec32afb59243e0a83d0216b56eccce.cloudapp.net/api/floods/1
Documentation: http://apifa.shoothill.com/help

https://hackpad.com/Data-APIs-for-floodhack-zd2MTv6chas

- EA's flood warning data
- EA's flood alert data
- Flood volunteers data
- …

step 1:
request: lat, long, radius, IDs for APIs to be queried

until the DNS propgates, put this in your /etc/hosts file:
74.50.52.21 floodfeeder.cluefulmedia.com

http://floodfeeder.cluefulmedia.com/api/floods?lat=51.431480&lon=-0.515525&radius=50
(that's the lat/long for Staines)
http://floodfeeder.cluefulmedia.com/api/floods?lat=51.431480&lon=-0.515525&radius=50

response: GeoJSON including lat, long, altitude, title, HTML (varies depending on the API being queried)

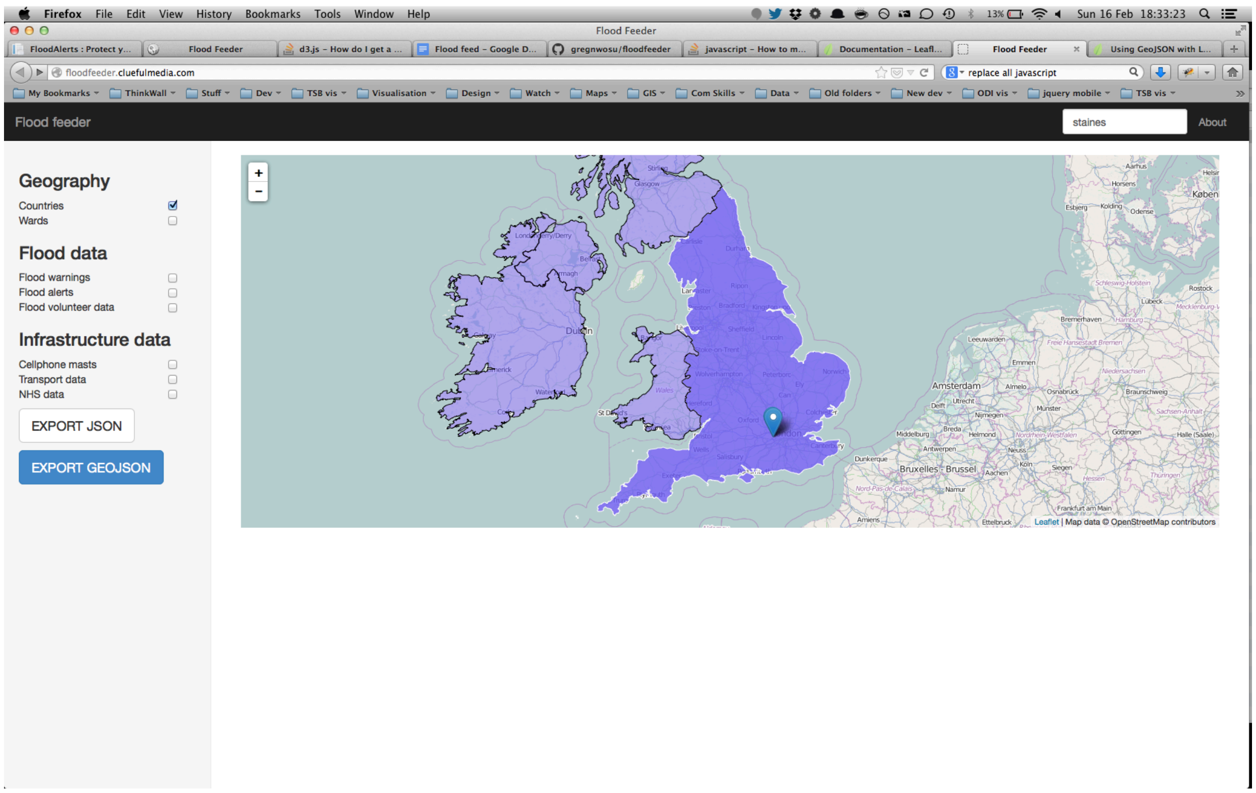
```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [-0.515525, 51.431480, 0]
  },
  "properties": {
    "title": "Dinagat Islands",
    “html”: “</blah”
  }
}
```

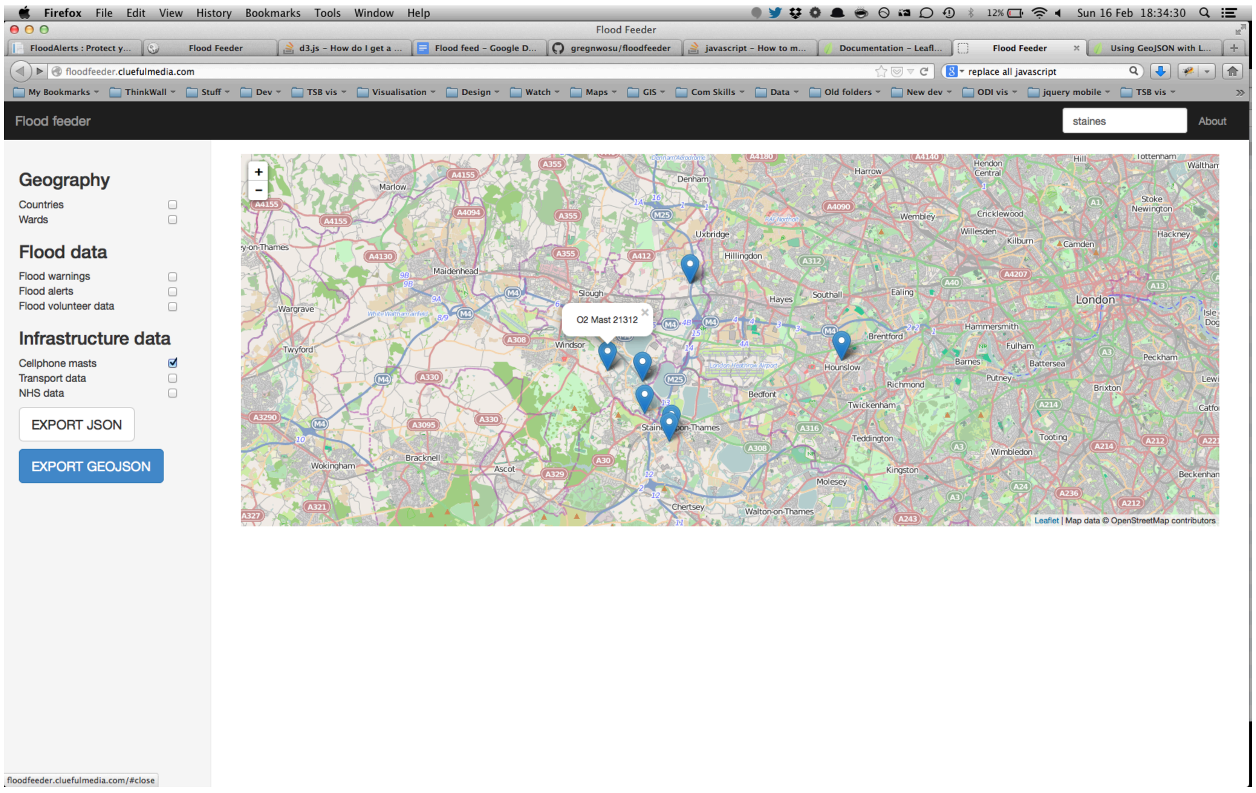
github

https://github.com/gregnwosu/floodfeeder.git

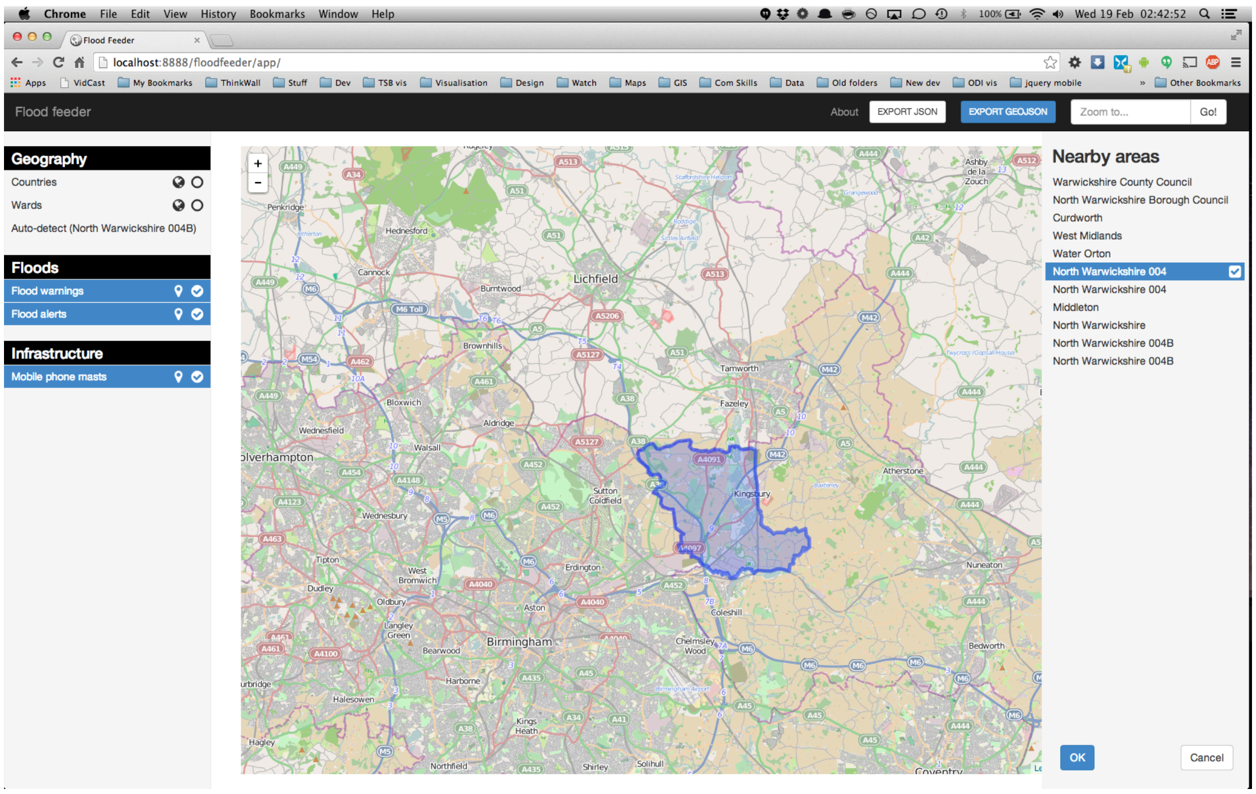
Screenshots

16th Feb


Flood feeder
staines
About
Geography
Countries
Wards
Flood data
Flood warnings
Flood alerts
Flood volunteer data
Infrastructure data
Cellphone masts
Transport data
NHS data
EXPORT JSON
EXPORT GEOJSON
Leaflet | Map data © OpenStreetMap contributors


Flood feeder
staines
About
Geography
Countries
Wards
Flood data
Flood warnings
Flood alerts
Flood volunteer data
Infrastructure data
Cellphone masts
Transport data
NHS data
EXPORT JSON
EXPORT GEOJSON
O2 Mast 21312
floodfeeder.cluefulmedia.com/#close
Leaflet | Map data © OpenStreetMap contributors

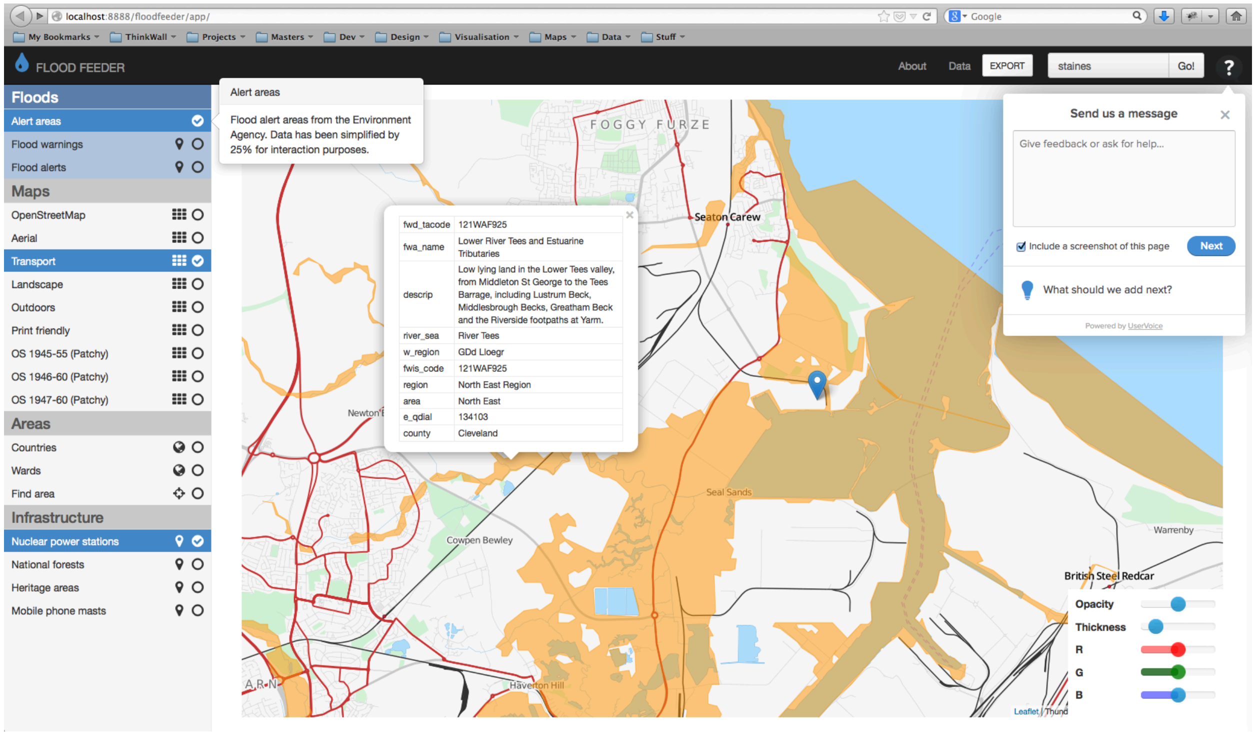
18th Feb



Nearby areas

- Warwickshire County Council
- North Warwickshire Borough Council
- Curdworth
- West Midlands
- Water Orton
- North Warwickshire 004 ✓
- North Warwickshire 004
- Middleton
- North Warwickshire
- North Warwickshire 004B
- North Warwickshire 004B

3rd March



Alert areas

Flood alert areas from the Environment Agency. Data has been simplified by 25% for interaction purposes.

fwd_tacode	121WAF925
fwa_name	Lower River Tees and Estuarine Tributaries
descrip	Low lying land in the Lower Tees valley, from Middleton St George to the Tees Barrage, including Lustrum Beck, Middlesbrough Becks, Greatham Beck and the Riverside footpaths at Yarm.
river_sea	River Tees
w_region	GDd Lloegr
fwis_code	121WAF925
region	North East Region
area	North East
e_qdial	134103
county	Cleveland

Send us a message

Give feedback or ask for help...

☑ Include a screenshot of this page

Next

What should we add next?

Powered by UserVoice

Meeting notes

/api/warnings?params1=&param2=
/api/mobilephones?params1=

/api/?dataset=warnings
/api/?dataset=mobilephones&lat=50&lon=42

UI calls:
/api/?warnings=true&bottomleft=50&topright=555
/api/?mobilephones=true&lat=50&lon=42

/api/?warnings

export URL: /api/?warnings=true&mobilephones=true

Export URL
http://floodfeeder.cluefulmedia.com/data?warnings=true&alerts=true&mobilephonemasts=true&bounds=[23.15352,34.23534]

installing stomp
pip install git+https://github.com/jasonrbriggs/stomp.py

Welcome to Network Rail Data Feeds.

Your account is now Active and you are now able to connect to feeds and receive information.

Please display the following security token in developed source code: 77274791-cebf-4a5d-af57-4a7a6272965c.