



DigitalLabs

@MMU

# DigitalLabs

@MMU



Digital Labs

@MMU

# What are we doing today?

# Today



DigitalLabs  
@MMU

Today we demonstrate tools and platforms that  
let you focus on the job in hand.

These tools remove boiler-plate coding and  
configuration effort.

They aren't a substitute for understanding!  
They need investigation before being used.

# Today



DigitalLabs

@MMU

## Tools Two:

- **JAM – Javascript, APIs, Markup - new Stack!**
  - Javascript throughout the stack
  - API fronts server-side functionality
  - Single Page Web App / Progressive Web App / Cordova App
- **Introducing REST APIs – make micro services in seconds! (small lie)**
  - RESTful Web Services
  - IDLs and the Open API Specification
  - IDL Tools
  - Security
- **Introducing PaaS – people making amazing useful stuff for free, so you don't have to!**
  - GitHub Pages
  - Restlet
  - Heroku
  - Auth0
- **Introducing your Case Studies - look at amazingly useful examples!**
  - The Urban Wild
  - Time Series Data Capture
  - LittleList



Digital Labs

@MMU

# Let's Fly!

# JAM Stack



DigitalLabs

@MMU

Yet another way to develop web applications:



Javascript, APIs, Markup

# JAM Stack



DigitalLabs  
@MMU

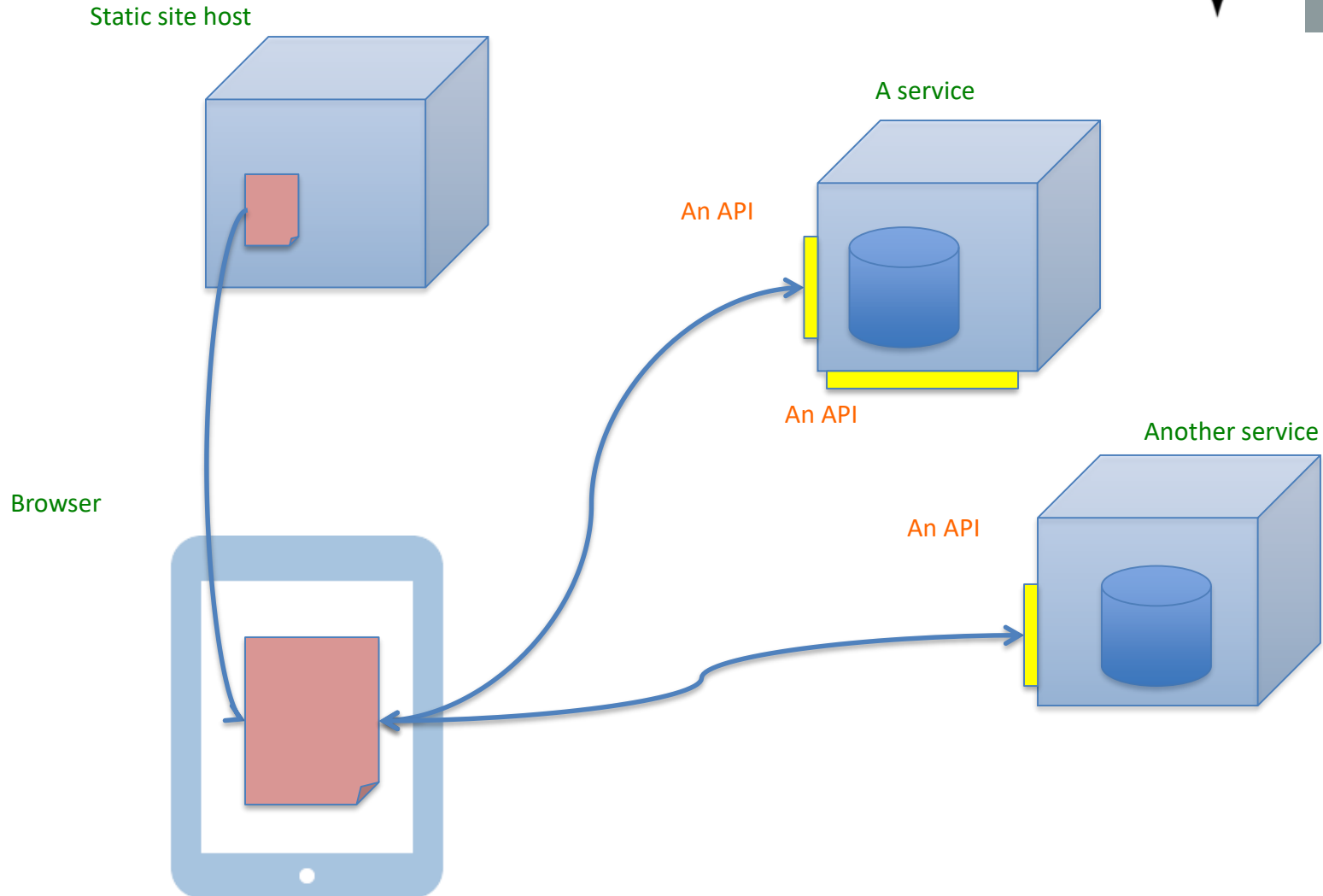
- A single set of Markup files (a web app) is served to a browser
  - Or
- Client is pre-built (mobile app)
- Javascript is used to control the application and its UI.
- The web app gets its data from web services, via their APIs

# JAM Stack



Digital Labs

@MMU





# JAM Stack



DigitalLabs

@MMU

What's an API?

- Application Programming Interface
- Sits between a service and its clients
- Imposes a predictable communication between them
- More [here!](#)

# JAM Stack



DigitalLabs

@MMU

## What's an API?

Quick example

- [The Urban Wild](#) - Wildlife logging app
  - User logs wildlife sightings at their location
  - User searches wildlife sightings at other locations



# JAM Stack



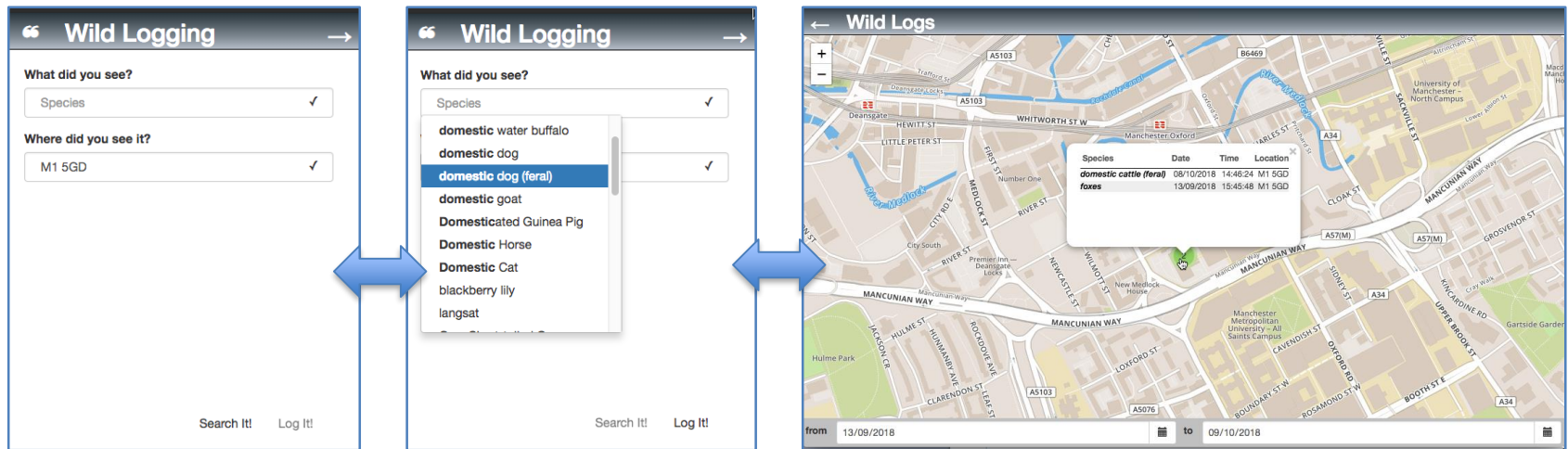
Digital Labs

@MMU

## What's an API?

### Quick example

- [The Urban Wild](#) - Wildlife logging app

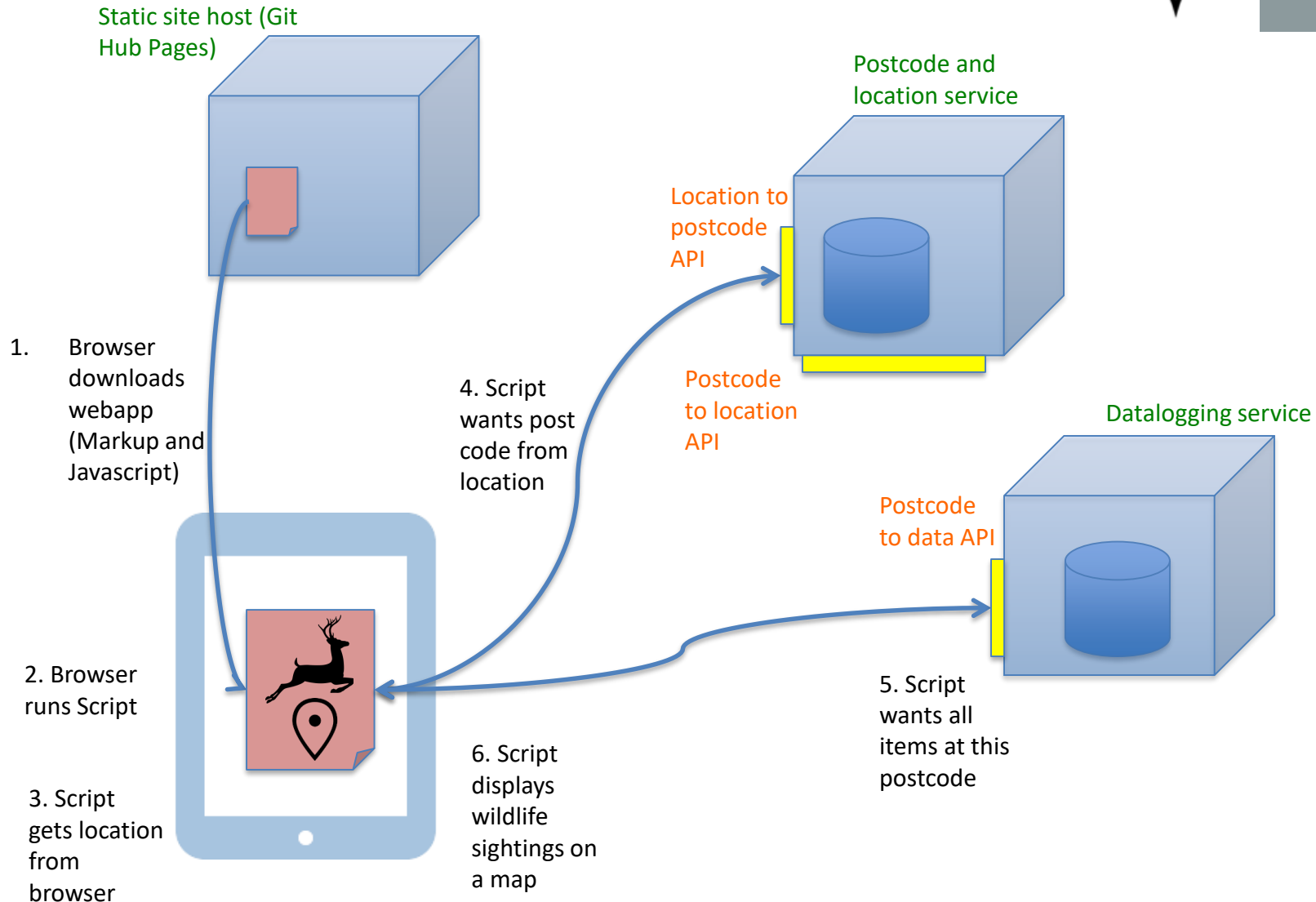


# JAM Stack



DigitalLabs

@MMU



# JAM Stack



Digital Labs

@MMU

## What makes a 'Good' API?

- **Efficient**
  - Minimise calls
  - Minimise bandwidth
- **Scalable**
  - Larger datasets
  - Many users
- **Well documented**
  - Essential
  - Accurate
- **Easy to use**
  - KISS \*

APIs are  
*designed:*  
Operations  
Data

# Introducing REST APIs



Digital Labs  
@MMU

## Communicating with the Service

- Clients communicate with a service via a protocol:
  - REST (Representational State Transfer)
  - SOAP (Simple Object Access Request Protocol)
  - Lots of Others
- Pass data over the API as a payload of the protocol:
  - JSON
  - XML
  - Lots of Others
- (you can even use CSV)

# Introducing REST APIs



Digital Labs  
@MMU

## RESTful Web Services

There are lots of protocols which can be used to communicate between clients and web services.

We use **REST**, because it makes development easier:

- Simple, yet rich
- Everywhere
- Well-tooled
- Not fast, not slow
- Not too verbose
- Easily debugged

# Introducing REST APIs



Digital Labs

@MMU

## RESTful Web Services

What's REST?

- REST (Representational State Transfer)
  - set of architectural principles
    - Sits on HTTP 1.1
    - Developed with HTTP 1.1 in 2000 by Roy Feilding
    - Clients interact with the service, using links and operations
  - A URL points to a **resource**:
    - Here's one for a collection:
      - <https://www.cars.com/registrations/>
    - Here's one for an item:
      - <https://www.cars.com/registrations/AB45PRZ/>
    - How about a postcode lookup?
      - <https://postcodesrus.com/postcodes/>



# Introducing REST APIs



Digital Labs  
@MMU

## RESTful Web Services

- HTTP used to interact with service's resources
- Request methods used to do this:
  - POST – creates
  - PUT – updates completely
  - GET (safe method)
  - DELETE – removes
  - PATCH – changes
- Standard HTTP responses returned

# Introducing REST APIs



Digital Labs  
@MMU

## RESTful Web Services

- The MIME type tells the client what data to expect in the response.
- Often:
  - [application/json](#)
  - [application/xml](#)

# Introducing REST APIs



Digital Labs

@MMU

## RESTful Web Services

POST <http://postcodesrus.com/postcodes>

Request Body:

```
{
  "lat": "53.474300",
  "lon": "-2.246820"
}
```

Response Header:

```
{
  "content-type": "application/json"
}
```

Response Body:

```
{
  "composite": "M1 5GD"
}
```

# Introducing REST APIs



DigitalLabs  
@MMU

## RESTful Web Services

Q: How did I know the service would do that?

# Introducing REST APIs



Digital Labs  
@MMU

## RESTful Web Services

Q: How did I know the service would do that?

A: It was documented, using an Interface Definition Language

# IDL

# Introducing REST APIs



Digital Labs

@MMU

## REST IDL

- Interface Definition Language
- Formal definition of an API
  - Operations
  - Data
- Accompanies the protocol
  - SOAP -> WSDL
  - REST -> OAS (swagger)
  - gRPC -> Protocol Buffers
  - Lots of others!
- *Advertises the capabilities of a service*
- *Enables generation of boiler-plate code*

# Introducing REST APIs

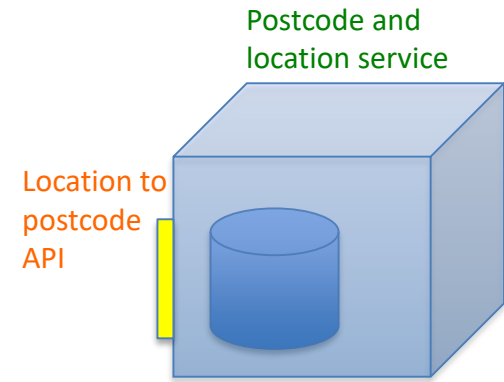


Digital Labs  
@MMU

## REST IDL

### Example

- Postcode and Location Service
  - Location to Postcode API
    - Defines the data types
      - 'location'
      - 'postcode'
    - Advertises resources
      - /postcode
    - Advertises parameters
      - Location
    - Advertises responses
      - Postcode



# Introducing REST APIs



Digital Labs

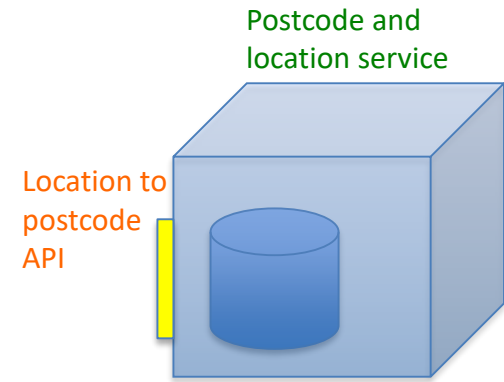
@MMU

## REST IDL

### OAS

- Open API Specification
- Previously called Swagger 2.0
- Based on YAML
- (YAML Ain't Markup Language)

```
---
swagger: "2.0"
info:
  description: "A simple API for looking -up postcodes, given a location"
  version: "1.0.0"
  title: "Postcode Lookup"
  host: "postcodesrus.com"
  schemes:
    - "https"
  consumes:
    - "application/json"
  produces:
    - "application/json"
  paths:
    /postcodes:
      post:
        summary: "lookup request"
        description: "creates a request to lookup a postcode"
        consumes: []
        parameters:
          - name: "body"
            in: "body"
            required: true
            schema:
              $ref: "#/definitions/Location"
        responses:
          200:
            description: "Lookup successful"
            schema:
              $ref: "#/definitions/Postcode"
          400:
            description: "Not a recognised location"
          404:
            description: "No postcode at this location"
        definitions:
          Postcode:
            type: "object"
            required:
              - "composite"
            properties:
              composite:
                type: "string"
                description: "the postcode as seen on an address"
                description: "A UK Postcode"
          Location:
            type: "object"
            required:
              - "lat"
            properties:
              lat:
                type: "number"
                description: "Latitude"
              lon:
                type: "number"
                description: "Longitude"
            description: "A geographical location"
```





# Introducing REST APIs



Digital Labs  
@MMU

## Tools

- [swagger.io](https://swagger.io)
- [RESTlet studio](#)

These will help you define an API, using an IDL.

These will generate code from an IDL.

You can create a Java EE / NodeJS server from an interface definition.

RESTlet Studio lets you define your API graphically

# Introducing REST APIs



Digital Labs  
@MMU

## Tools

- [RESTlet cloud](#)
- [PostgREST](#) (Haskell)

These will help you create an API from a database  
They will generate the service to wrap the database, too.

["It's so fast to develop, it feels like cheating!"](#)

# Introducing REST APIs



DigitalLabs  
@MMU

## Security

HTTP supports authentication

HTTPS allows secure transmission

IDL allows definition and advertising of the security scheme

# Introducing REST APIs



Digital Labs  
@MMU

## Security

HTTP supports authentication

HTTPS allows secure transmission

IDL allows definition and advertising of the security scheme

A properly defined and constructed API will ensure:

- secure transmission of data
- immediate rejection of unauthorised requests

# Introducing REST APIs



DigitalLabs  
@MMU

## Security

### Why do I need security?

- If you deal with any kind of user data, then you need to be aware of:
  - [The Information Commissioner's Office \(ICO\)](#)
  - [The Data Protection Act](#)
  - [The General Data Protection Regulations](#)

If you don't need to, don't store personally attributable / identifiable information AT ALL. EVER.

# Introducing REST APIs



Digital Labs  
@MMU

## Security

Securing websites and databases is tricky

- It is an arms race.
- Use Platforms As A Service to delegate:
  - Complex configuration
  - User Authentication

# Introducing PaaS



Digital Labs  
@MMU

## Platform as a Service

A server system which provides a set of services, accessible online.

In particular for us:

- Database Hosting (Mongo, MySQL, Postgres...)
- API services hosting
- User accounts and authentication

# Introducing PaaS



Digital Labs

@MMU

We use [RESTlet](#) to create simple data services.

## Why?

- For simple database storage for a prototype, it gives us what we need really quickly.
  - **define** the API:
    - Define the operations it exposes
    - Define the data it returns
  - **generate** the **service** which implements it:
    - Wait for requests from clients
    - Fetch database data
    - Fetch data from other services
    - Return data



# Introducing PaaS



DigitalLabs

@MMU

We use [Heroku](#) as our easy-to-use service provider.

## *Why?*

- Much of the configuration previously seen with web servers is hidden.
- That means it's more difficult to make mistakes

# Introducing PaaS



Digital Labs

@MMU

We use [Auth0](#) as our easy-to-use authentication provider.

## *Why?*

- We can delegate all user accounts to experts
- [They provide a robust method by which we can authenticate users.](#)
- That means it's more difficult to make mistakes

# Introducing PaaS



DigitalLabs  
@MMU

We use [GitHub Pages](#) as our easy-to-use front-end provider.

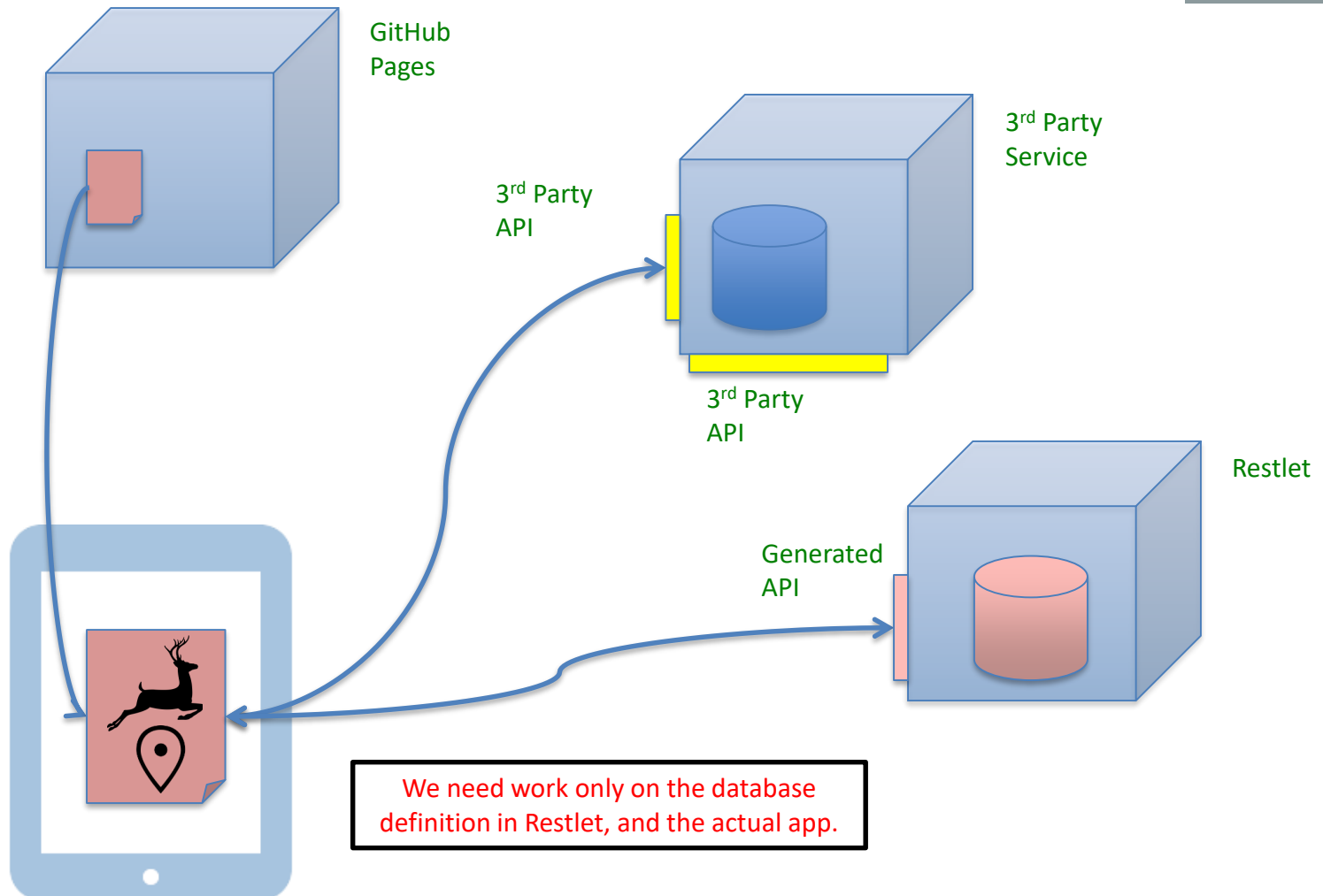
## *Why?*

- It's easy.
- We can host static web pages which authenticate through a third-party, and then access data via an API. Oo! JAM!
- Pushed immediately from our [GitHub repo](#)

# Introducing PaaS



Digital Labs  
@MMU



# Case Studies



DigitalLabs

@MMU

We have produced a set of Case Studies and example code to help you as you design and construct your Live Project:

- **The Urban Wild**
  - Project: [Github](#)
  - Diary: [DigitalLabs Post](#)
  - Web App: [Angular JS](#)
  - Host: [GitHub Pages](#)
  - APIs
    - [Postcode](#)
    - [IT IS](#)
    - [Restlet PaaS](#)
- **Time Series Data Capture**
- **LittleList**

## The Urban Wild

- Demo Project
- Let's try and push the limits of what we can do without creating our own server, just using PaaS.
- Allow casual users to add wildlife sightings to a map.
- Allow them to search on location and date for other sightings.
- 65 Hour rush job.

# Case Studies



DigitalLabs

@MMU

We have produced a set of Case Studies and example code to help you as you design and construct your Live Project:

- The Urban Wild
- Time Series Data Capture
  - Project: [Github](#)
  - Diary: [DigitalLabs Post](#)
  - Web App: [Angular JS](#), [D3](#)
  - Host: [GitHub Pages](#)
  - APIs
    - [TimeSeriesDataCapture\\_ImportSource](#)
    - [TimeSeriesDataCapture\\_BrowseData](#)
    - [Microsoft OneDrive](#)
    - [Auth0](#)
- LittleList

## Time Series Data Capture

- Scientists like to save their data in CSV files
- Then they put it on OneDrive
- ... can't find anything!
- This system imports data to Mongo DB
- Adds preview, tagging, annotation, search, visualisation.
- This was a 60 Day project, by a Mentored Summer Student

# Case Studies



DigitalLabs

@MMU

We have produced a set of Case Studies and example code to help you as you design and construct your Live Project:

- The Urban Wild
- Time Series Data Capture
- LittleList
  - Project: [Github](#)
  - Mobile App: [Github](#), [Ionic1](#), [Cordova](#)
  - Web App: [Github](#), [Angular JS](#), [Ionic1](#),
  - Host: [Github Pages](#)
  - APIs

## LittleList

- A tiny project for you to fork
- Use it as a basis for anything which gets a bunch of items from a search
- Easy demo of concepts like
  - Changing UI States
  - Asynchronous calls
- How-to for mobile works out-of-the box on Lab PCs.

# Things To Do



DigitalLabs

@MMU







DigitalLabs

@MMU

# DigitalLabs

@MMU