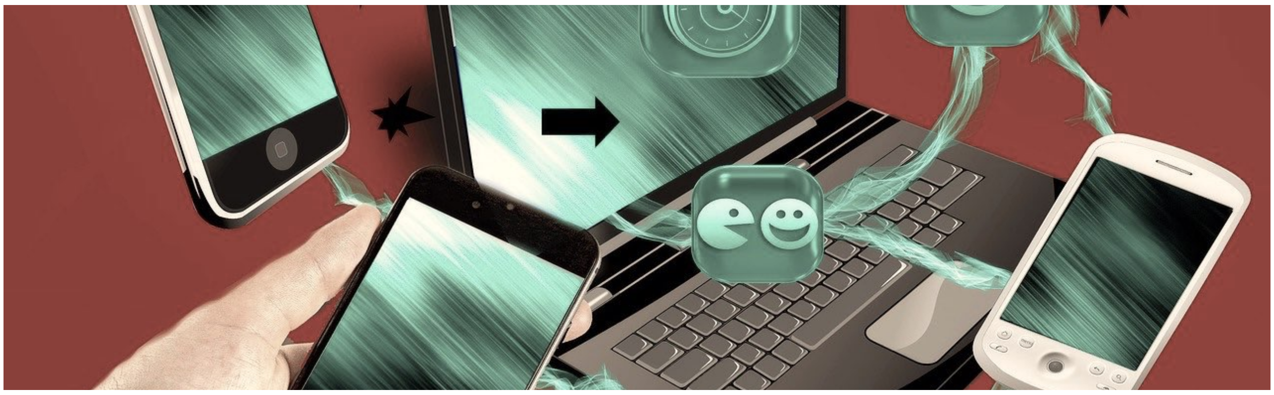
Richard Heward
Tame Blue Lion Ltd
11th May 2017

Hacking a Strategic Option

A hackathon approach to bring a new concept to life



Introduction

This paper describes how it is possible to quickly build a solution to a real business problem, in order to see it is a viable strategic course of action. This is at the 'other end' of a previous post I have published on Digitising The Strategy.

The real-world requirement was to find an option to improve how customer ordered packages were managed. These packages are physically small and are pre-bagged and stored for customer to be picked up. This is similar to 'click and collect' but with the exception that the order is usually a pre-organised subscription mixed with additional requests, plus the actual chosen products of the order are defined by the business and third parties.

As this concept was built I realised that the same solution to find the packages could also be used in the rest of the workflow to assemble the bag and also to hand them out.

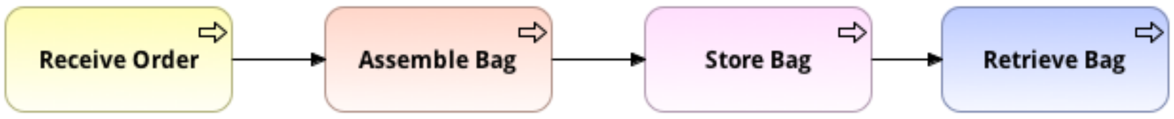
The objective therefore was to quickly and cheaply mock-up a barcode solution to bring clarity as to what an enterprise solution might look like and how it might work, in order to stimulate stakeholder engagement and help strategic thinking: "Hacking a Strategy".

In doing this we realised that multiple business stakeholders were looking for a similar solution in isolation, we could have an enterprise wide capability which would enable all projects, and also be a platform for additional opportunities. It could be elegant, enterprise, be quick to deliver and save money.

Overview

The following high level process summarises the points at which an order is managed through the workflow. Two of us realised we had enough technical skills and ambition to easily bring a part of this to life.

Here is the simple overall "level zero" process. In reality there are several sub-processes.



Store Bag and Retrieve Bag

The last two steps of the process were used as a start points this is where clarity was needed most. An assembled bag is stored and sometime later retrieved.

The existing process used a manual card index deals to look after the bags too be placed into a pigeon-hole style shelving system .

A new automated retrieval system would use a barcode already printed on the bag label and also a new barcode that is attached to each pigeon-hole shelf location; this assigns a location in the storage area.

When the barcode is scanned to add a bag to storage a "ready to collect" text message would be sent to the customer. Then when the customer arrives to collect, a customer search screen can be used to show where the bag is located in storage making it easier, reducing waiting time and increasing customer-facing time.

The automated system will also be able to automatically contact customers who have not collected their order at any given interval. This would simplify the way storage is managed and potentially result in less stock being tied up unnecessarily.

General Workflow

It is easy to assign barcodes to any location in the bag preparation area, not just the storage shelves, so we could extend the same solution leftwards into the 'Receive Order' and 'Assemble Bag' processes. This will lead to greater insight as to order throughput times and more opportunity to search for where a prescription is in the process.

Receive Order

The ‘Receive Order’ process involves downloading the details of the order or capturing a live order in-store into the system such that we can quickly produce a barcode label that will go into the ‘Assemble Bag’ process.

Assemble Bag

During bag assembly, adding packages to the bag is achieved by scanning each package (as they have their own product barcodes) to build up a list of what’s in each bag.

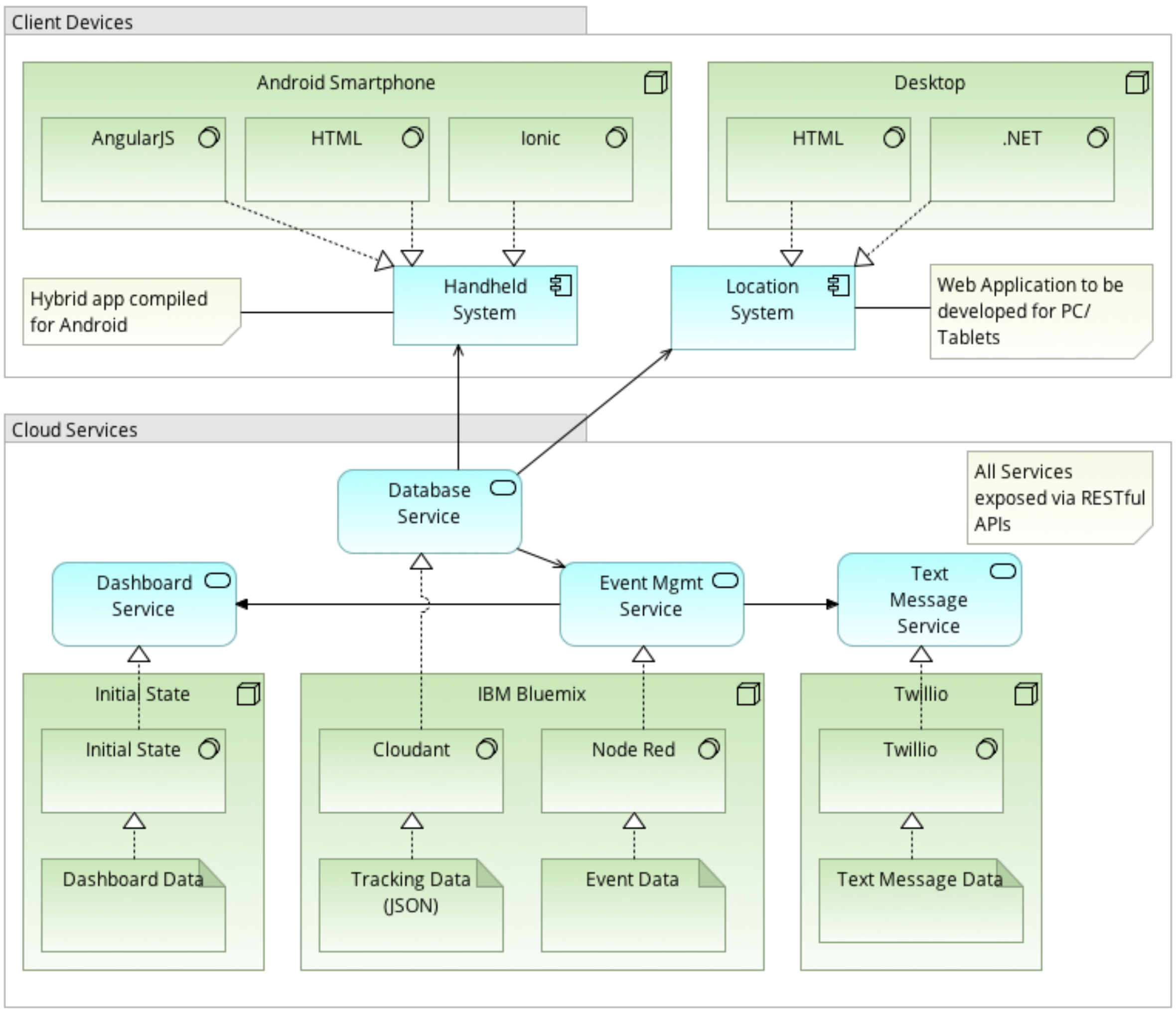
Validation Checks

There is a use case where the system needs to report and check what packages are in each bag. We know this because of the Assemble Bag process and then it is just the event of scanning the bag off the shelf (as scanning the shelf itself is unnecessary) is enough to trigger a final message which inspects the bag data object to see what is in it (without having to open it).

Solution Architecture

In order to develop a trial solution quickly and cheaply it was decided to use technology we had to hand which we had some knowledge of. That meant a Smartphone as a Barcode scanner, connected to cloud technologies for the Database, Integration, Messaging and Dashboard analytics layers.

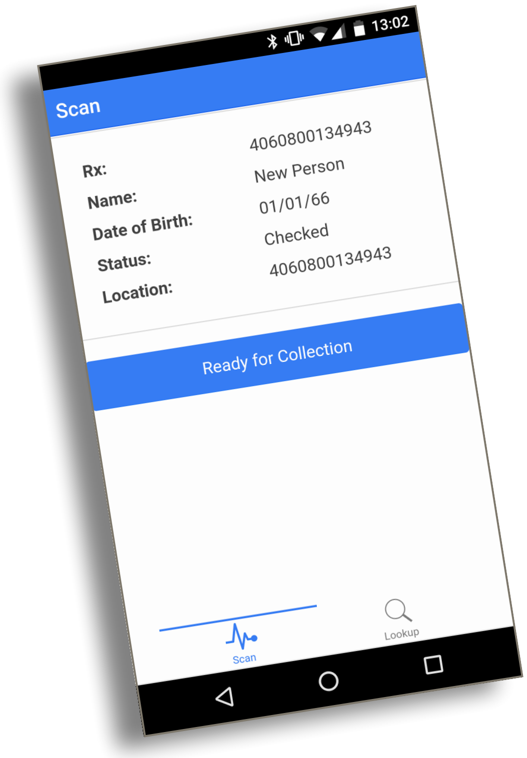
The following ArchiMate view gives a high level overview of the components and services used.



These elements are examined in more detail below.

Handheld System

The application for the mobile handheld system was developed locally in HTML5 (presentation), AngularJS (logic) and the Ionic Framework (UI), and then packaged into a native Android application using Apache Cordova, only because we had an Android phone to hand.



Rx: 4060800134943
Name: New Person
Date of Birth: 01/01/66
Status: Checked
Location: 4060800134943

Database Service

The data was stored in JSON files on an IBM Cloudant (NoSQL CouchDB variant) database hosted on IBM Bluemix. This was accessed via an IBM Bluemix account, which was easily accessible from anywhere.

JSON gave us a great deal of flexibility such that it was easy to define addition elements of data, including timestamps, statuses, arrays of associated items, image attachments and so on without redesigning the database schema. For example, a single record is represented as follows:

```
{
  "_id": "01293787b2g13g1g2f3h1g2",
  "_rev": "5-40b3bcea2b0e5db7614bfab9bb617143",
  "bagid": "123321",
  "name": "Julie Jane",
  "dob": "01/01/66",
  "status": "Ready",
  "updated": "03/04/2017 12:32:33",
  "location": "B1",
  "notify": true,
  "type": "live",
  "items": [
    "123321",
    "133322"
  ]
}
```

Messaging

The messaging layer was implemented on the Twillio cloud messaging service. This sent custom text with context to a mobile number of our choice. The trial version also added its own prefix header text, but this did not matter.

Integration

The Mobile application communicated with the Cloudant database using a Restful HTTP API.

The Dashboard and Messaging components were triggered using IBM Node-RED hosted on IBM Bluemix. Node-RED (https://nodered.org/) is a graphic representation of all the various end points and connections between. It allowed for a flexible (drag-and-drop) way of building up the solution with many nodes available to connect to Cloudant, to investigate and transform message content and so on. In addition, javascript functionality was written for some of the more tricky manipulations. New nodes and flows are easily found in the free library.

The flows can be triggered manually, or on a regular interval (so the dashboard looks more up-to-date). The dashboard library provided useful input nodes, which allowed the flows to be triggered directly from the dashboard. It was also possible to pass in data from on-screen sliders and so on.

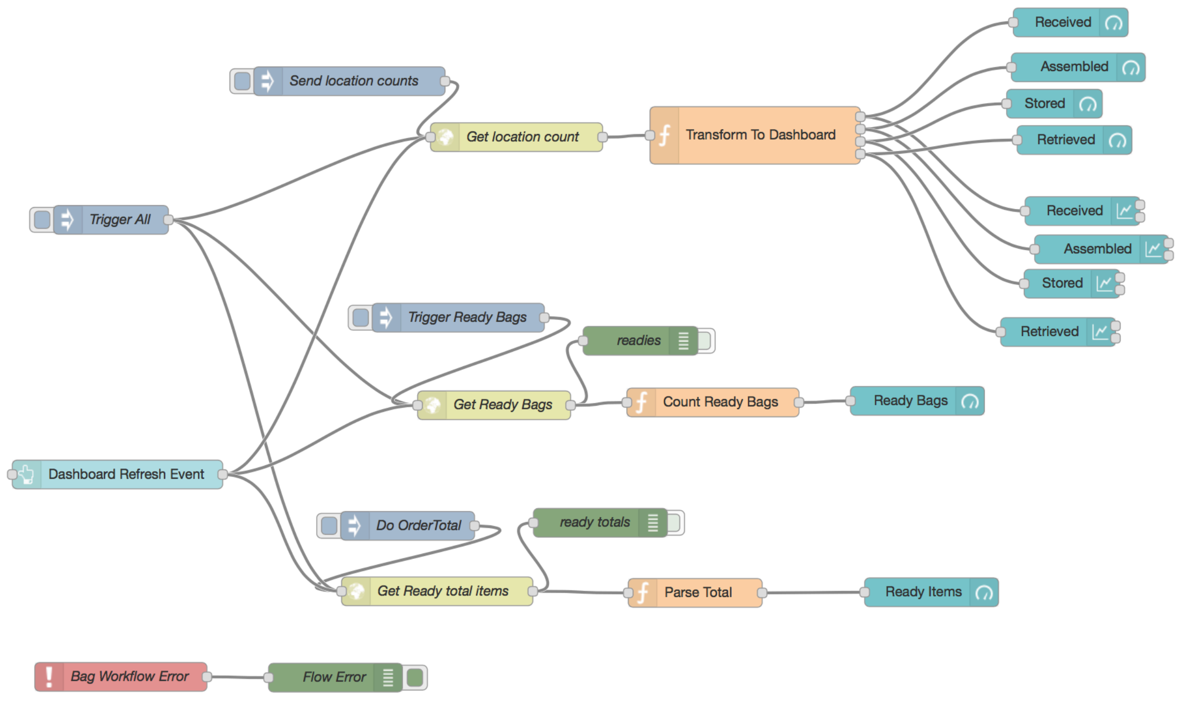
Node-RED itself made it easy to create new restful API's directly, so more complex interactions can be exposed. This can even create web pages directly, but required more time than we could warrant.

Here are the details of the flows used:

Bag Workflow Flow

The Bag Workflow Flow does three things based on triggered queries to the Cloudant database:

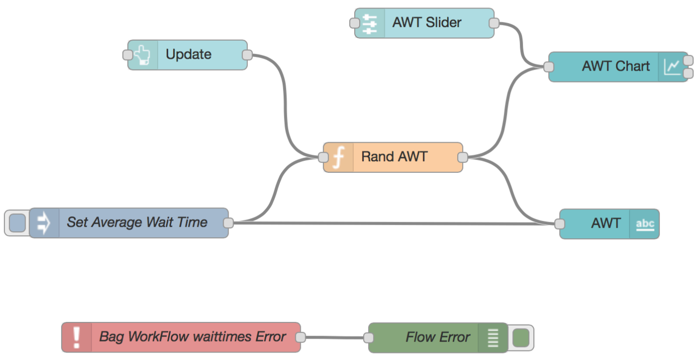
- Counts how many bags are in each location. It displays these on gauges and on level meters in the dashboard.
- Counts how many bags are ready and displays this on a gauge.
- Counts how many packages are in the bags and displays this on a gauge.



Customer wait times Flow

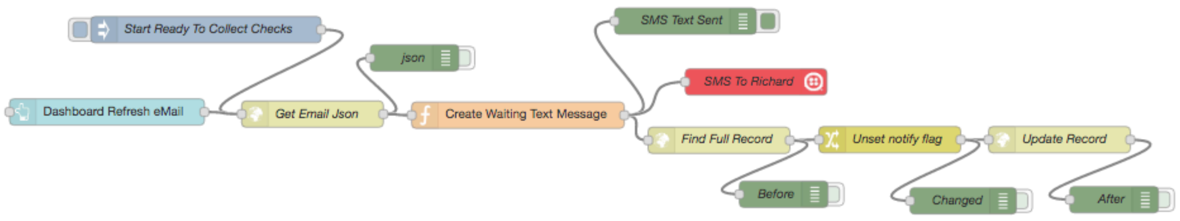
The following flow merely generates a random number and shows this on a chart and as a value on the dashboard. It can also be overridden using a slider on the dashboard. Given time, the intention was to calculate the waiting times from the history records in the database.

We also changed the database document structure halfway through to include the concept of ‘live’ and ‘history’ document. This could lead to the ability to scan the history documents in order to derive wait time.



Messaging Flow

This flow queried the Cloudant database to see which live prescriptions need to have a reminder sent (a simple boolean variable on each document). When it found one, it would get the full record details, send a contextual reminder message via the Twillio SMS service to a fixed mobile phone number (the "customer's"), Then unset the 'send an email' flag. This effectively de-queued the waiting prescriptions one by one. Finally, we added a refresh button on the dashboard if we didn't want to wait for the next automatic interval.

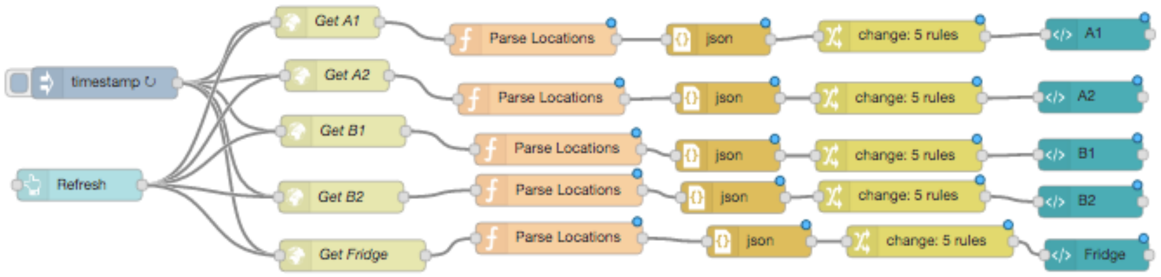


```
// Create Waiting Text Message
Msg1 = { payload: msg.payload.length };
var rows = msg.payload.rows;
var mymsg = "";

if (rows.length === 0) {
    mymsg = Msg1;
}
else {
    var row = rows[0];
    var v_bagid = "";
    v_bagid = row.fields.bagid;
    Msg1.topic=v_docid;
    mymsg = "Bag "+v_bagid+" is ready to collect.
Please come to collection point A. ";
    Msg1.payload = mymsg;
    return Msg1;
}
```

Bag locations Flow

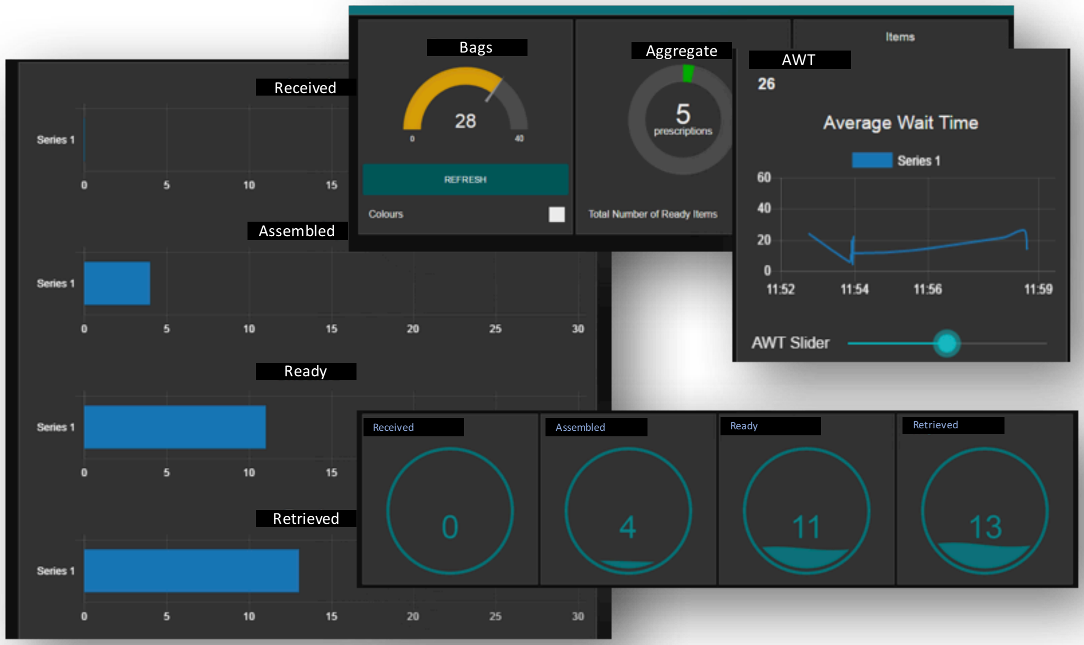
This flow is less elegant in that it gets the key data from each prescription, by location. It could be rewritten to work without knowledge of the locations, but there wasn't enough time and it just had to work. Note we only had five locations (A1, A2, B1, B2 and Overflow).



Dashboards

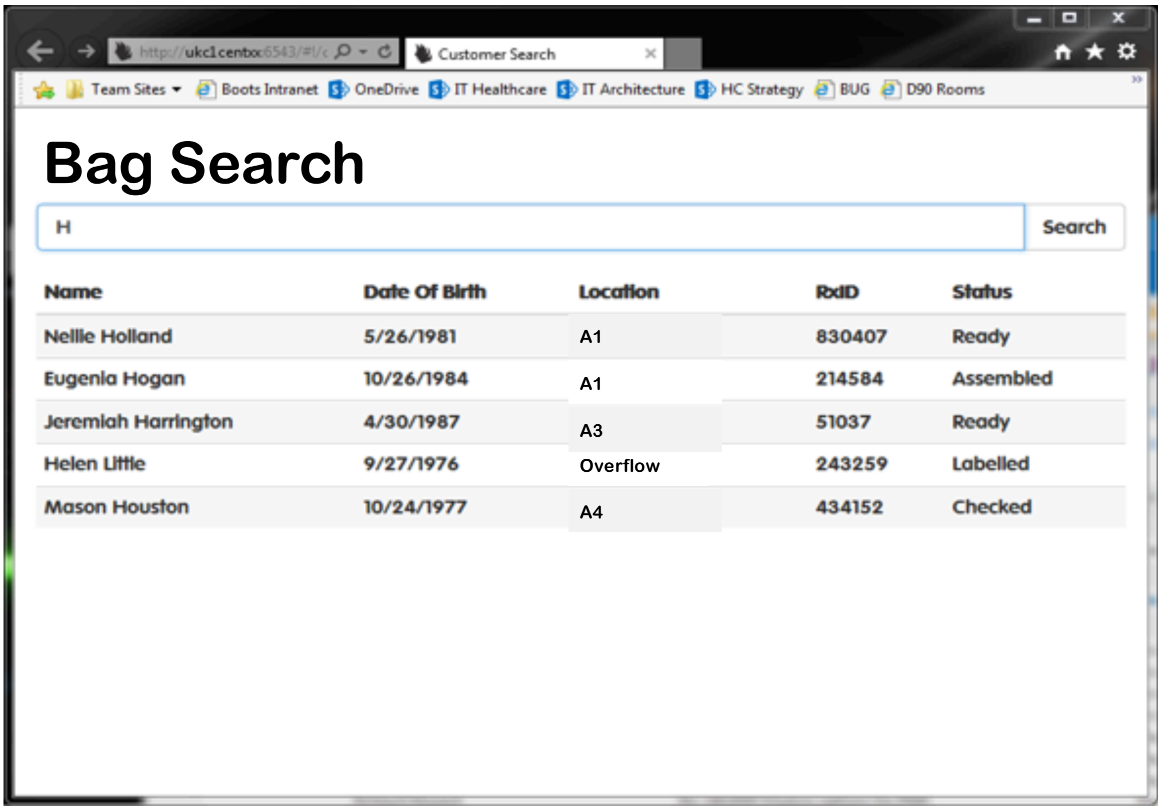
The Dashboard was developed with Node-RED Dashboard. This provides a simple API that can be called from Node-RED itself.

At first we used Initialstate to visualise some of the key data from the database. However this is more geared towards Internet Of Things device data and only gave us something else that could go wrong. Then we discovered that a new set of nodes were published onto the Node-RED library providing exactly what was needed. As it was using the same technology approach the changes took a matter of hours, and actually simplified the flows.



Bag Search Website

The Bag Search Website connected to the Cloudant database using the same REST Web APIs as used by the mobile client. In fact, the lookup was also performed from the mobile device.



Bag Search

H

Search

Name	Date Of Birth	Location	RxID	Status
Nellie Holland	5/26/1981	A1	830407	Ready
Eugenia Hogan	10/26/1984	A1	214584	Assembled
Jeremiah Harrington	4/30/1987	A3	51037	Ready
Helen Little	9/27/1976	Overflow	243259	Labelled
Mason Houston	10/24/1977	A4	434152	Checked

The assumptions for deploying a web application for hosting within a store were that:

- It needed to be lightweight
- It needed to be easy to deploy.
- Ideally we want to be able to Unit Test it as much of it as possible.
- Ideally backwards compatible to earlier browser versions.
- We want to deliver the good UI experience.
- To perhaps expose an API for other applications to communicate with.

For a rich UI we'd probably lean towards using a Single Page Application (SPA). The industry leader is Angular which version 1 is still compatible with Internet Explorer 8. Angular has unit testing built into its design so you can test more functionality than you could with an equivalent MVC solution. In addition, Bootstrap was used to quickly leverage responsive controls in IE8.

At the back end, we need to host the static pages that constitute the angular web site and some form of REST service to serve data to the Angular app when it's running on the client. IIS is overkill as we'd only be serving data to the store. This makes it a perfect candidate for self-hosting (i.e. hosting the web application as a windows service) which is lightweight and easy to deploy. NancyFX does just this with the added bonus that it's "low-ceremony" (you don't have to write much code to get the job done) and it appears that it has a really good API for Unit Testing.

Analytics

Full analytics were not needed for this trial, although we did expose the NoSQL data through DashDB. This can expose more SQL like queries again through a Restful HTTP API.

Ideally we would want to measure the time between messages to reflect how long customer transactions took to be processed (See the Average Wait Times Flow above), but we were able to cover this concept with the presentation and dashboard demonstration.

Barcodes

We drew inspiration from a typical existing click-and-collect pattern which use barcodes on shelf edges to associate to the parcels being placed there.

A quick google of free barcode generators allowed us to mock up some bag labels (on real bags) and shelf labels (or, as in one demo, on an in-box office tray). Even mockup packages were used for additional scanning of packages against a bag, which was a useful demonstration for the validation concept.

A1

Time Taken

The first generation was developed in a week (1 man week effort & 1 week elapsed) at which point it was demonstrated to initial senior stakeholders. It was well received and further changes suggested before it could be presented more widely. These changes took another 3 weeks to develop (1 man week effort & 3 weeks elapsed), at which point it was validated with the business practitioners before being demonstrated to the business leadership teams.

A good proportion of the time taken was in gaining familiarity with new technology, much of which was done in our own time (lots of youtube training), but had this been a true hackathon, with assigned skilled resources (in these technologies), the delivery time would have been even quicker.

It should be noted that time would have been spent on this activity anyway, and experience has shown that standard workshop-PowerPoint-presentation cycle often repeated beyond expected levels of efficiency.

Stakeholder Response

The senior stakeholder response to the show and tell was very positive. The meeting itself started with our usual PowerPoint, then. the offer of a hands-on demonstration was welcomed. The attendees stood-up around our make-shift "storage shelves" with the barcode edges and our mobile phone. Projected onto the wall was the dashboard, showing live changes as our bags we stored, searched and handed out.

One issue we did not expect was how the stakeholders suggested how it could be extended to bring even further business benefit. Comments such as "we can see how this will work" and "can we include financial information on the dashboard?" and "this would be great to show to a store". Afterwards we were in a stronger position to discuss how to take it to the next level and how a business case was now closer.

IT stakeholders have also commented how the approach shaped project thinking and how it demonstrated how quickly things can be done (for example, spinning up a platform as a service). Although not the primary objective it provided great insights into new technologies, which can be taken into consideration for future architecture thinking.

Stakeholder types

This approach will generally work for practical, visually-led stakeholders, or anyone who likes a more engaging fast-track approach to creating a vision. Those wanting to explore detailed requirements would be best engaged in the traditional manners.

Lessons Learned

What went well?

- The speed to develop and test theories quickly
- Business Involvement through regular show-and-tell sessions, helped avoid going off in the wrong direction.
- The ability to chose any free technology that we had (or quickly obtain) skills in.
- Creating the concept was productive and therefore enjoyable. Much of the time spent on learning or doing the more complex delivery was done in our own time, because we wanted to get to the next level. Like a personal hackathon without the Pizzas.
- Node-RED was a revelation. We were already used to it from our own Raspberry Pi based hobbies, but its ability to adapt to whatever scope we challenged it with was impressive. Direct access to the database, JSON message manipulation, dashboarding, sending text messages / eMail, text and image analysis and detailed JavaScript coding. This is unlikely to scale at an enterprise level, but that's not the point of the hackathon work.

What could have been better?

- Greater availability to technology resource - It would be better to have some of these tools and capabilities already in-house
- Some of the software we wanted was trial-ware or low cost. Any costs were paid for by the concept builders (around £5 for processing time).
- The ability to query the data in a historic manner to get a time-based view of each prescription. NoSQL makes this harder, but not impossible, however we ran out of time for this.
- The use of Cloudant (or NoSQL/CouchDb type database) was highly flexible around adopting new ideas. For example, adding of items for bag validation was just a simple array structure to existing documents, and adding photos to a bag order was also possible as just a document attachment. However, the indexing and views is complex and seemed to resort to re-watching a youtube or just plain trial-and-error (then success). At least we learnt the concepts behind "map reduce"!

What types of project benefits from this approach?

- Potentially any project concept, but it is likely the criteria will be similar to that of choosing the Agile approach, i.e. projects where requirements are unclear, the solution is uncertain etc.
- High complexity projects. Specific key design decisions of such a project could be trailed up front, just to confirm specific technology elements, for example.
- Projects where requirements are uncertain. By building something, even if it is not the final concept or even if it turns out to be wrong is still helpful to driving requirement thinking.
- Low complexity, simple or short timescale projects are less likely to be a good candidate.

Alternative approaches

PowerPoint

It's quick and if done well can easily portray a concept or way of working. However, it is open to interpretation and is not always engaging or hands on enough

Prezi

Animated presentations are a much more visual and engaging way of presenting a concept. However, although better than Powerpoint it still isn't as engaging as real demonstration

Third party consultants

Making use of partner organisations or even investing in someone to build a makeshift solution has its place. Many partners will invest for free as there is a chance it will convert to an delivery opportunity for them.

Research

Check whether someone already proven it? Don't repeat it again, unless you want to localise it in some way.

Mini Architectures

Traditional project shaping doing some high level Enterprise Architecture views. This can help clarify but has to appeal to the stakeholders.

Requirements capture and analysis

Traditional project shaping in a Mandate relies on the Business Analyst capturing initial requirements so that the EA can design an Outline Architecture.

Agile Workshops

Early workshops with stakeholders, BA's, EA's, SA's, Testing and Service will provide a joined up view and often lead to workable solutions.

Wire-framing
A great way to visualise workflows and screens and it doesn't have to be complex.

Mind Mapping
A visual means of capturing disparate competing concepts at once in order to arrive at a unified way forward.

Alternative technologies
MySQL instead of NoSQL.
This may be easier to query. However this will require a less flexible scheme design to occur.

Using Software AG's Apigee framework.
This offers a Backend as a Service (BaaS) including its own Cassandra No SQL database. I tried a simple database and JSON documents just for academic interest, but it did seem to be too API oriented.

Hosting on Raspberry Pi
As a novel safe standalone off network solution, most elements could be hosted on Raspberry Pi, including node-RED, CouchDB (instead of Cloudant) etc. Node-RED can also be implemented in Microsoft Azure, Amazon Web Services and under Docker.

Python Applications
The Bluemix version of the Python SDK was downloaded a simple database actions explored, but Node-RED was enough for now.

NodeJS
The Bluemix version of NodeJS SDK is also a good method of delivering applications, in a Java style. Node-RED was good enough so this was not explored.

Dashboards
An alternative dashboard suggestion was PrimeNG – a port of the PrimeFaces project. This was more flexible, but wasn't something we could use with the time and skills available. Google Charts may have a place too, but this wasn't investigated.

Pitfalls
Governance
- Simple housekeeping and controls are still necessary in a hackathon world. The last thing you need is for simple to clear down a database and loose all the indexes and views as well.
- Keep versions so that when you're in the middle of change and adding extra functionality you can always demonstrate a working version.

Getting carried away

- It's always possible to add a crazy feature "just because it's possible". It probably isn't necessary to prove the concept at hand.
- Also remember that it isn't the actual solution. If the code works, it doesn't have to be elegant. The Node-RED flows are a good example of that.
- Making the applications looking too pretty and free flowing. This isn't essential, unless that's the very core of what you're proving.
- Know when to stop - "It's just a demo!"

Building the real thing

- You're not building the real thing, so don't get adding real-world features or interfaces.
- There is no need to create frameworks or many reusable concepts. You should expect to throw it all away.
- Be consistent about the messages to stakeholders. This isn't ever going to go to production.

Summary

Crucially, by adopting the principle that the solution would never see real-world use, it was possible to be *very* agile in the way it was assembled and the technology choices that were used. It was also important to stress this to any stakeholders and decision makers along the way. The effort required to hot-house real solution components takes greater rigour, investment and commitment.

Whilst traditional alternatives such as requirements-driven specifications, workshops and PowerPoint or employing third party consultancy have their place, we found easier consensus was made sooner in this case.

The solution could be extended to other functionalities or the mobile app to capture a photo and attach it to the JSON document. The search application could be modified to only show those bags with photos waiting to be assembled.

It should be noted that it was the solution concept itself that was being investigated, not the system or the method being used to create it (as discussed in this paper). In effect, stakeholders arrive at the same outcome, but in a faster, different, more engaging way. In our real world case, we expected to have to use the standard design and delivery documents, only we can be clearer about the vision and can complete such documentation more easily.

The real-world win with this example was the realisation that a solution that was hacked-up for one part of a process, could apply to other parts too. So not only was the process approach proven, so was its ability to work elsewhere.

This paper can act as a pattern or guidance for other such trial solutions. The choice of technologies used can be varied depending on skills and expertise, however a common set of core products could be identified to further speed-up concept realisation.

Richard Heward
m: richard@tamebluelion.co.uk