Proposal for APA Databases
Dec 2019
Nathaniel Tagg

This is a very basic set of working notes trying to map out the right way for the DB to work.

Requirements:
(1) Keep track of components by some kind of serial numbers
(2) Keep track of what components make up what assemblies
(3) Submit testing data, both as database form values AND saved files (PDF/ROOT/whatever)

Constraints:
(A) Multi-site access.
(B) Straightforward GUI; user training should be minimal
(C) Simple authentication (use existing or simple group passwords). I
(D) Very responsive to changes in data structure as system evolves
(E) Data submission systems are more urgent than data retrieval

Thoughts:

Constraint (A) means we web-based access is probably the best (and happens to be what I'm good at anyway). The current industry-standard for rapid web application development is to use node.js as the front end (which again, I've got lots of experience with).

Constraint (D) is the most crucial point: the data structures are likely to evolve a lot over time. Therefore, I believe we should go with a No-SQL solution, to avoid the problems of rapid schema changing.

Constraint (B) with (D) requires some way of having a rapidly-changing gui. I think forms.io looks best, although I am uneasy about using the forms.io database backend for data submission.

The rapidly changing nature of the database, combined with constraint (C) means that we have to be careful about the data model. All transactions should should be saved even if they are overwritten by other changes in future.

Real-world proposal:

Create a QR code for every single object that gets put into an APA. Stick to object if possible; otherwise stick to label attached. Also can be put on external box.

QR code is very robust to scratches and dings. This can encode the unique ThingID used for every component in the system.

If scanned with a smartphone, this code can geotag and timestamp the object, allowing a

Here is your shiny new QR code! Print the image below and paste it

http://sietch.xyz/d068da29-ea6f-4623-abf2-9125f9ee4a24

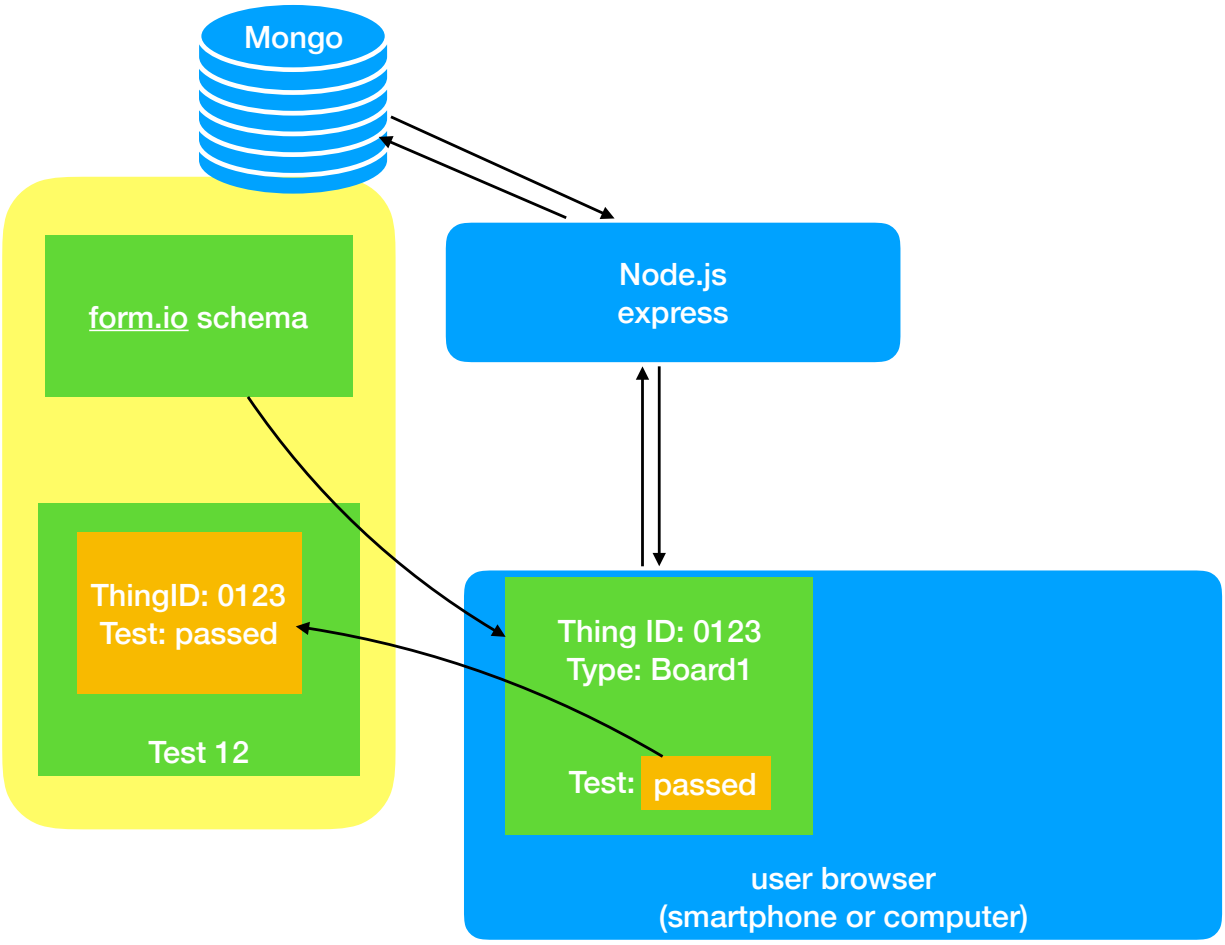
http://sietch.xyz/d068da29-ea6f-4623-abf2-9125f9ee4a24

http://sietch.xyz/d068da29-ea6f-4623-abf2-9125f9ee4a24

http://sietch.xyz/d068da29-ea6f-4623-abf2-9125f9ee4a24

http://sietch.xyz/d068da29-ea6f-4623-abf2-9125f9ee4a24

realtime tracking inventory.



Infrastructure:

Single server (hosted at Otterbein?) running Node.js HTTP server for access and MongoDB back-end. Ubuntu operating system seems easiest. Single URL.

Regular database backup to other sites, but do not attempt database replication or synchronization.

Save files either with DB path hashing, or directly with GridFS in Mongo.

Database:

In Mongo, a “Table” is called a “collection”. Not all elements of a collection need be identical, nor do they need to obey a schema. Each element is called a “document” and is composed in JSON (allowing for tested structure).

One collection will house a basic tracker of the list of thing ID numbers and their types. This is the “Thing” table.

There will be one entry per thing in each of it’s associated collections, like an “APA” table or a “cold board” table or whatever. Each can have different fields. These should be used to list the provenance and ID numbers of that object.

There will be a collection for each kind of test that can be run on each object. One such test can be a “location test” which simply maps when objects were created and geotagged.

There will be a collection for the forms that are displayed to the user. This actually will be the thing that defines the schemas used in the other tables!

Files will be stored in GridFS buckets with ObjectID references associated with the tests.

Database collections (“tables” in SQL parlance:)

Thing		
	ThingID	UUID (or similar
	Type	TEXT (from dropdown options)
	Components	list of ThingIDs
Depending on type (documents in a collection can by polymorphic)	Serial number	
	Lot number	
	Manufacturer	
	Procurement date	
	Assembled at site	
	Last updated	date

testForm		
	Form type	type
	Form items	Dropdown - Assembled At
		Serial number - text
		etc

test		
	ThingID	UUID
	passed	true/false
	test data	test.root

Transaction		
	insert	date submitted
	IP address	from client
	credentials	username
	url	
	form schema	
	submission	document submitted