Sietch
APA construction database system

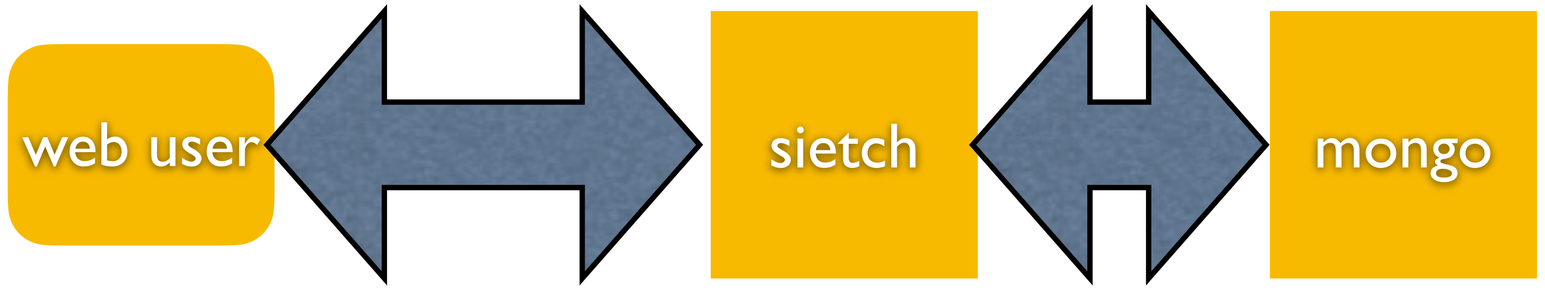
Nathaniel Tagg
Otterbein University
June 2020

Sietch is a tool designed to

easily and flexibly track data about APA components.

- It's a web application, can be accessed by most web browsers
- Connects to a MongoDB database backend
- I've been developing it for about 6 months. Based on open source: Node.js, FormIO.js, Express, Passport. Uses auth0.com for authentication.

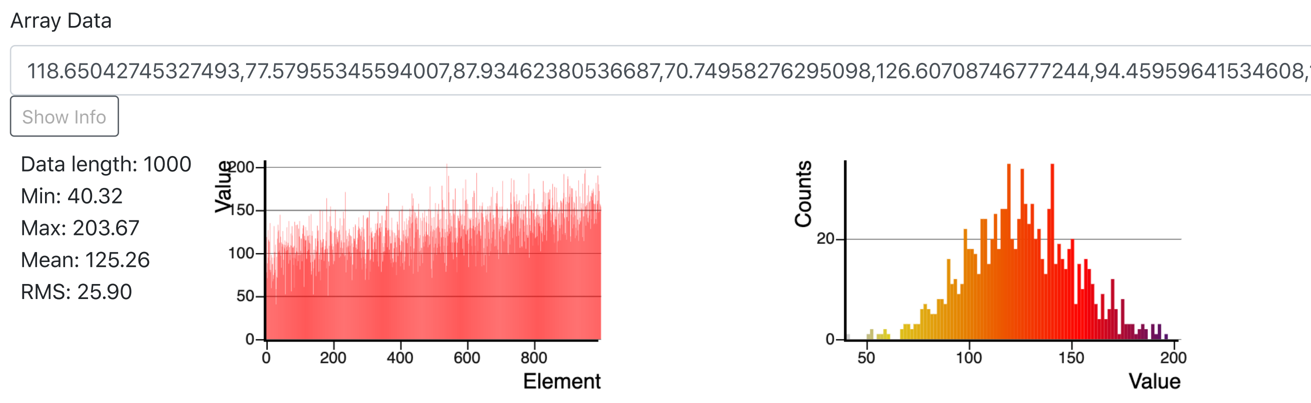
Development server live at https://sietch.xyz



What is it?

Use cases I’ve had in mind:

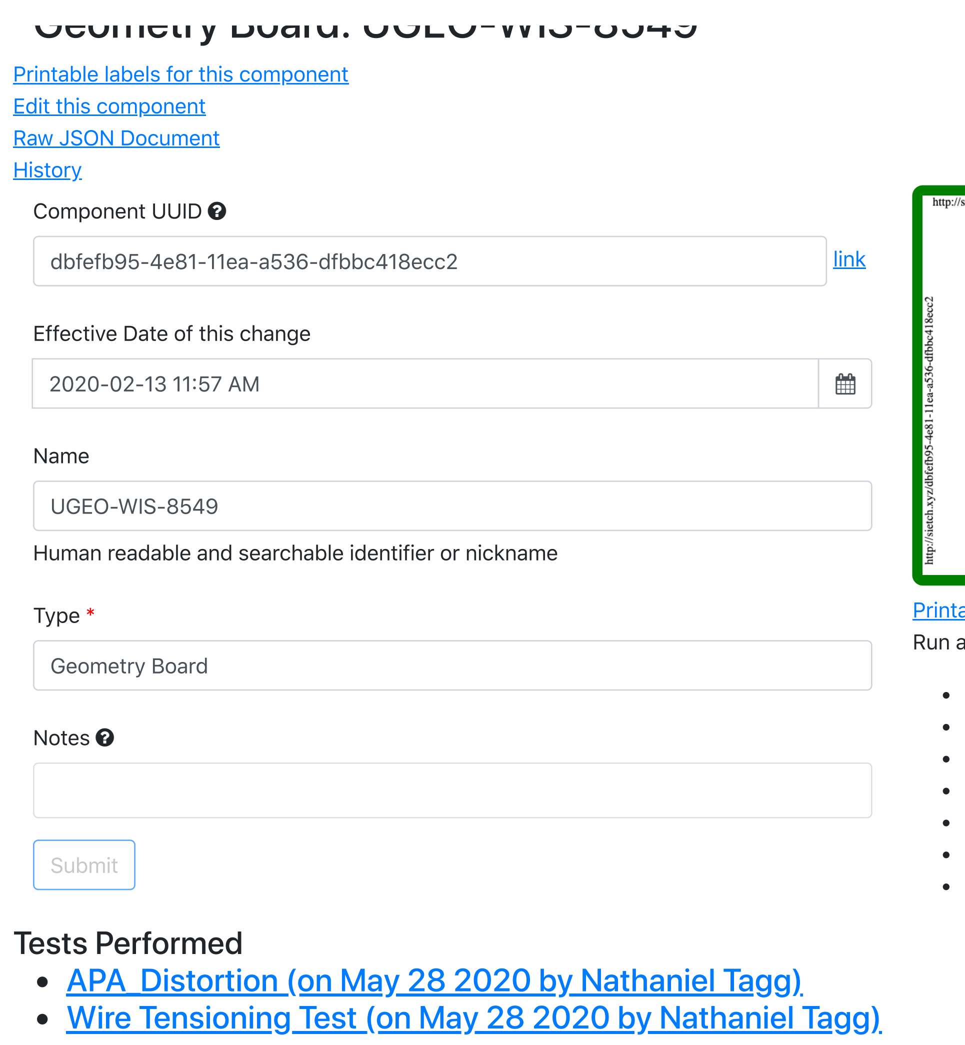
1. QA/QC data (e.g. wire tensions) that might be relevant to later physics analysis
2. Virtual “traveller” documents
3. Tracking component relationships (which cards built into which APA)
4. Inventory tracking
5. Checklists / work instructions -> Becoming more important


Array Data

118.65042745327493,77.57955345594007,87.93462380536687,70.74958276295098,126.60708746777244,94.45959641534608,

Show Info

Data length: 1000
Min: 40.32
Max: 203.67
Mean: 125.26
RMS: 25.90


Geometry Board: UGEO-WIS-8549

Printable labels for this component
Edit this component
Raw JSON Document
History

Component UUID

dbfefb95-4e81-11ea-a536-dfbbc418ecc2 link

Effective Date of this change

2020-02-13 11:57 AM

Name

UGEO-WIS-8549

Human readable and searchable identifier or nickname

Type *

Geometry Board

Notes

Submit

Tests Performed

- APA_Distortion (on May 28 2020 by Nathaniel Tagg)
- Wire Tensioning Test (on May 28 2020 by Nathaniel Tagg)

Printa
Run a

Uses

Mongo is a NoSQL database.

Instead of ‘rows’ there are JSON documents.
There is no (enforced) schema.

Instead of ‘tables’ there are ‘collections’.
Collections may be heterogenous.

This allows for a lot of flexibility: we can change the database schema on the fly!

This puts schema development in the hands of the experts instead of the database manager.

```
{
    "_id" : ObjectId("5ede8f3a87232201bb430e2d"),
    "APAID" : "US APA 004",
    "Frame_Serial_Number" : "004",
    "Head_SN" : "0005",
    "Center_SN" : "0006",
    "Side1_SN" : "0007",
    "Side2_SN" : "0007",
    "Foot_SN" : "0005",
    "componentUuid" : UUID("f54b4cc0-a9bc-11ea-bc7b-ff175b
    "type" : "Protodune APA",
    "name" : "Protodune APA US APA 004",
    "effectiveDate" : ISODate("2020-06-08T19:19:22.259Z"),
    "submit" : {
        "insertDate" : ISODate("2020-06-08T19:19:22.259Z")
        "user" : {
            "user_id" : "m2m9fe996973ff972f6",
            "displayName" : "autouser",
            "emails" : [
                "nathaniel.tagg@gmail.com"
            ],
        },
        "version" : 1,
        "diff_from" : null
    }
}
```

Every object has a unique "serial" number in the database (a UUID), represented by a QR code:

http://sietch.xyz/eb833bd0-a9c1-11ea-bbcc-8dda7beedcce

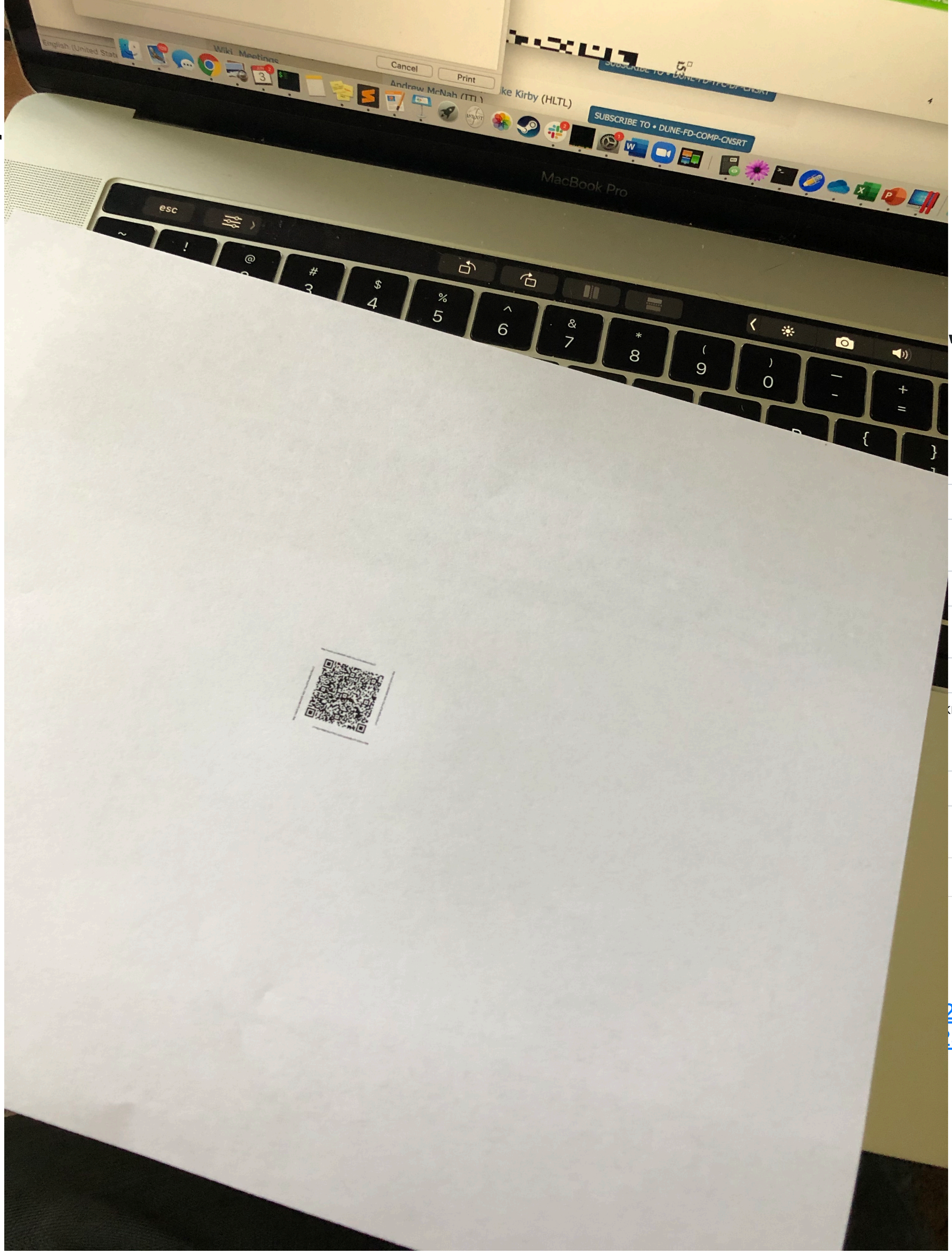
http://sietch.xyz/eb833bd0-a9c1-11ea-bbcc-8dda7beedcce

http://sietch.xyz/eb833bd0-a9c1-11ea-bbcc-8dda7beedcce

http://sietch.xyz/eb833bd0-a9c1-11ea-bbcc-8dda7beedcce

This is used as a indexed key into MongoDB

If



name

20 by Na
8 2020

Basic concepts

Basic philosophy based on Nick West's DatabaseInterface package for MINOS:

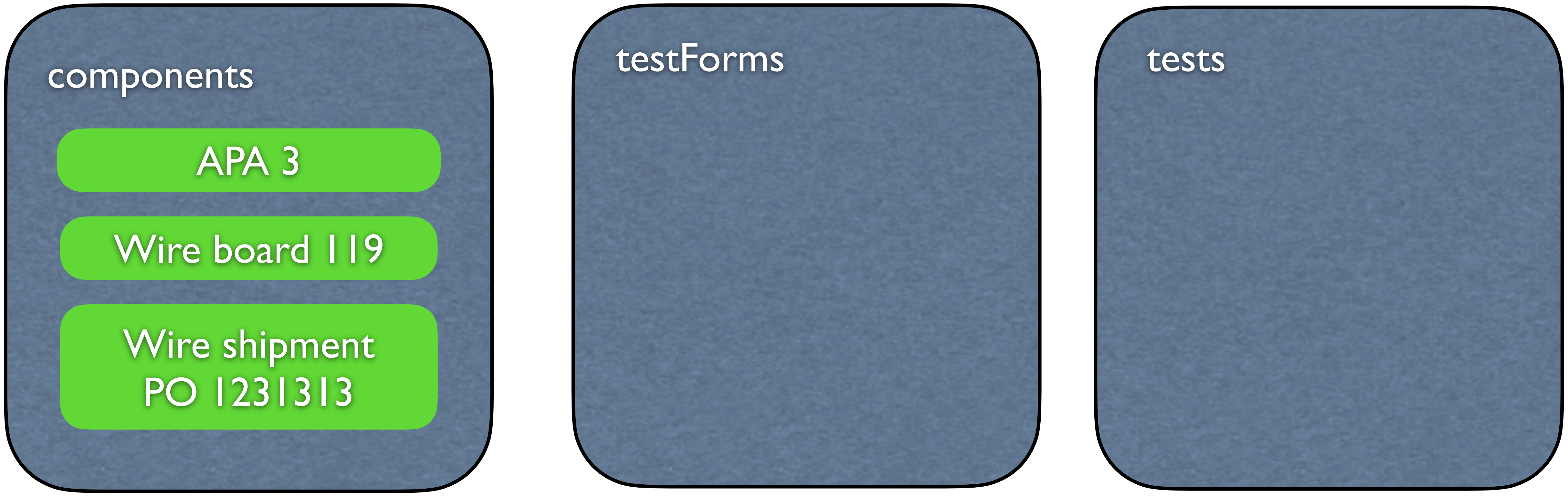
- Never delete things, only layer on top
- Use dates for both insertion AND validity range

Each record type has a rigorous schema for metadata,
but makes no restriction on the form of the data.

Basic concepts

components
testForms
tests

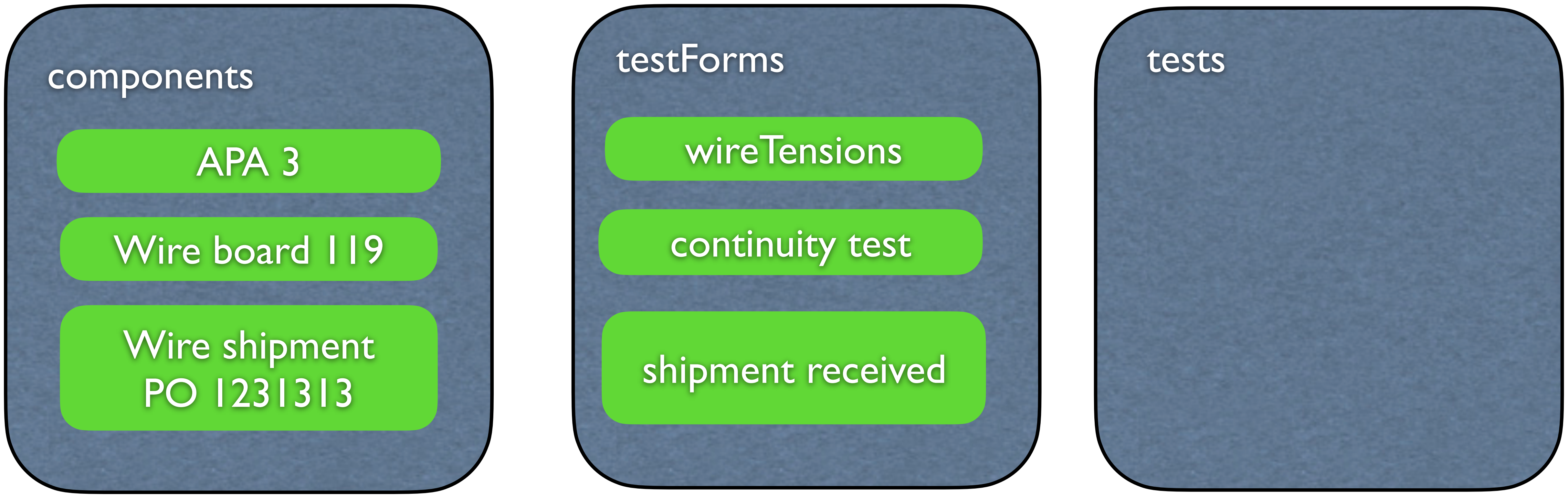
Important Sietch Collections



Each "component" is a physical object

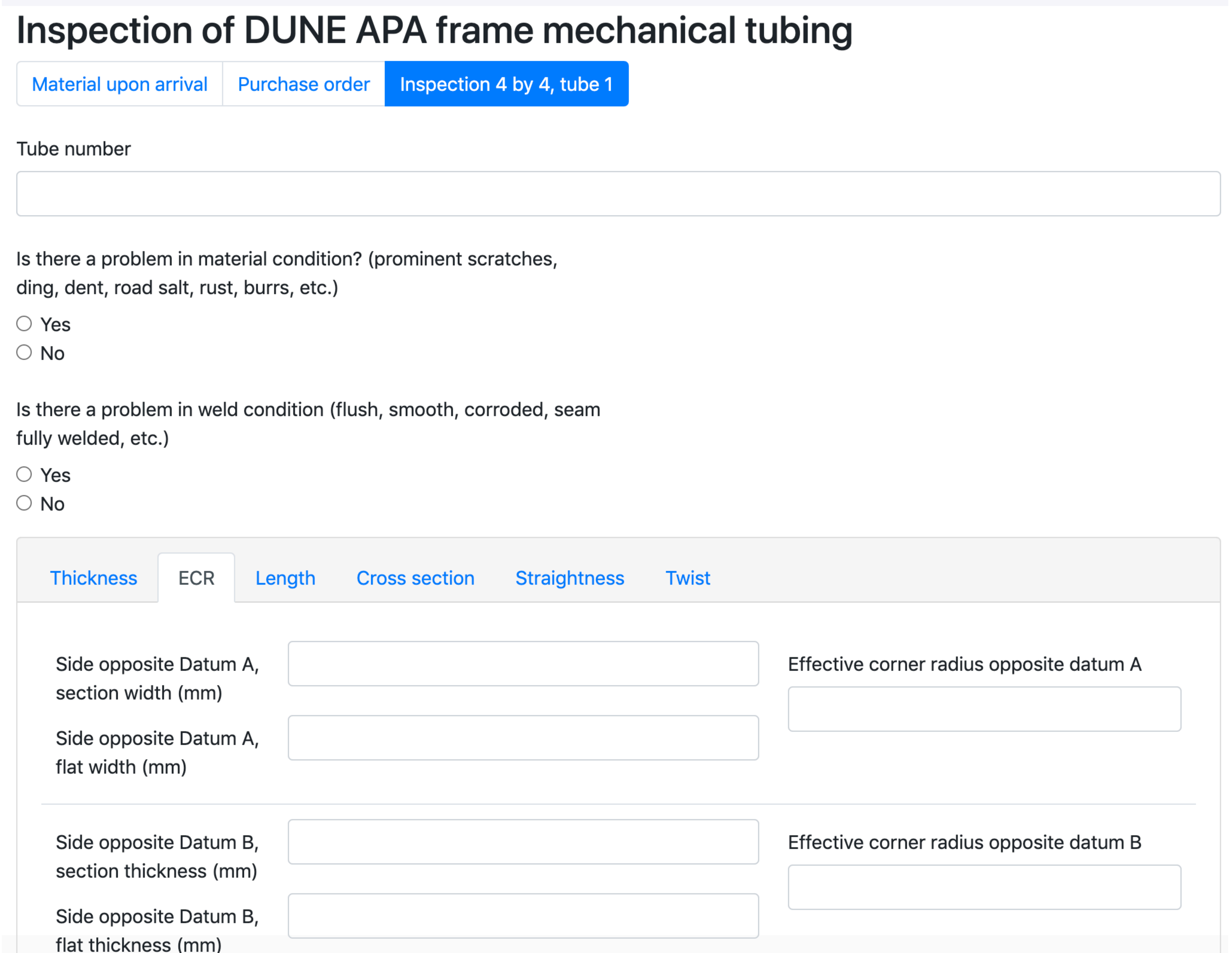
Each has a record keyed by a UUIDv1, which can be put on a QR code on the object or package

Important Sietch Collections



These are ‘forms’.
They describe the UI that allows user data input.

Important Sietch Collections



Inspection of DUNE APA frame mechanical tubing

Material upon arrival | Purchase order | Inspection 4 by 4, tube 1

Tube number

Is there a problem in material condition? (prominent scratches, ding, dent, road salt, rust, burrs, etc.)

- Yes
- No

Is there a problem in weld condition (flush, smooth, corroded, seam fully welded, etc.)

- Yes
- No

Thickness | ECR | Length | Cross section | Straightness | Twist

Side opposite Datum A, section width (mm)

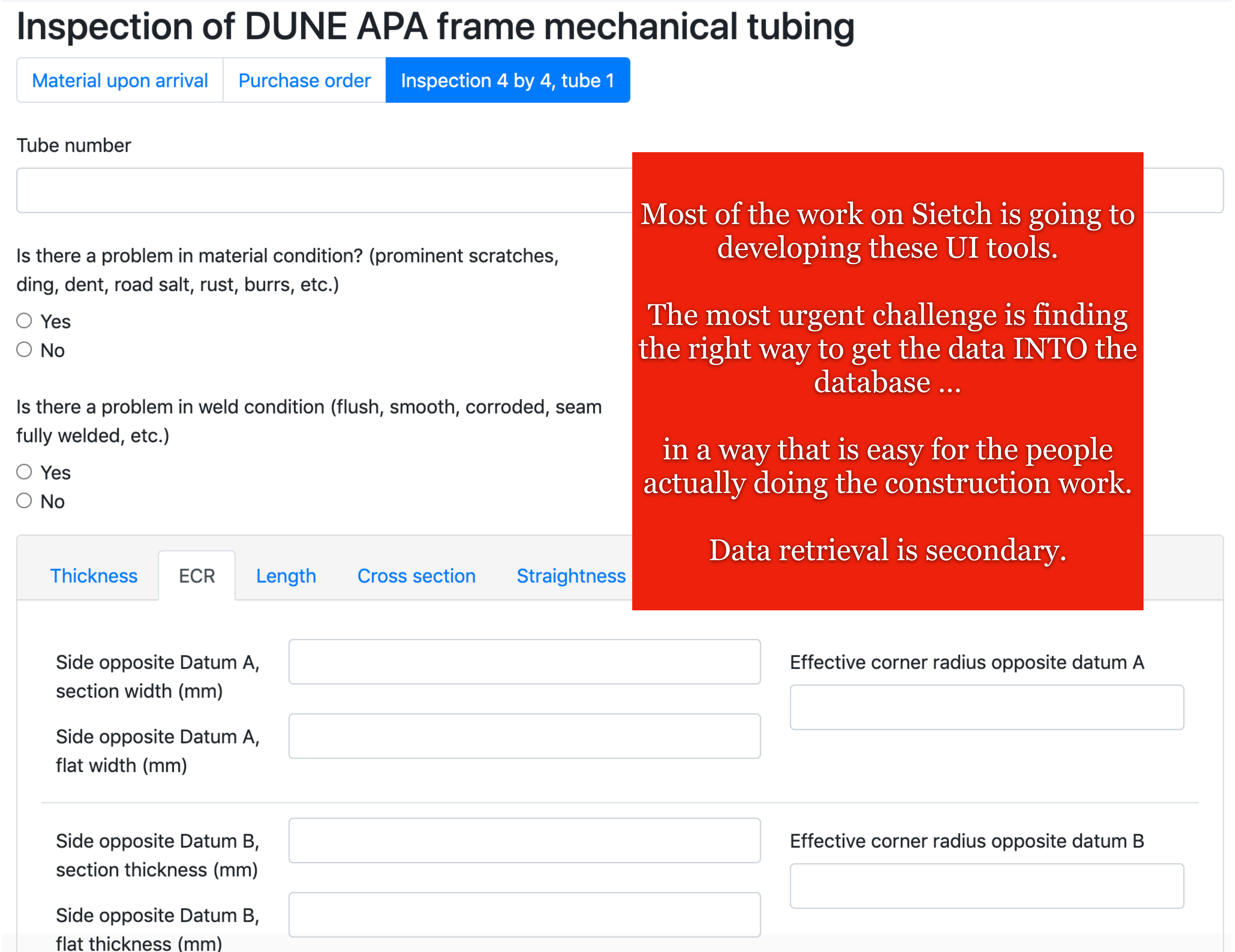
Side opposite Datum A, flat width (mm)

Effective corner radius opposite datum A

Side opposite Datum B, section thickness (mm)

Side opposite Datum B, flat thickness (mm)

Effective corner radius opposite datum B



Inspection of DUNE APA frame mechanical tubing

Material upon arrival | Purchase order | Inspection 4 by 4, tube 1

Tube number

Is there a problem in material condition? (prominent scratches, ding, dent, road salt, rust, burrs, etc.)

- ○ Yes
- ○ No

Is there a problem in weld condition (flush, smooth, corroded, seam fully welded, etc.)

- ○ Yes
- ○ No

Thickness | ECR | Length | Cross section | Straightness

Side opposite Datum A, section width (mm)

Side opposite Datum A, flat width (mm)

Effective corner radius opposite datum A

Side opposite Datum B, section thickness (mm)

Side opposite Datum B, flat thickness (mm)

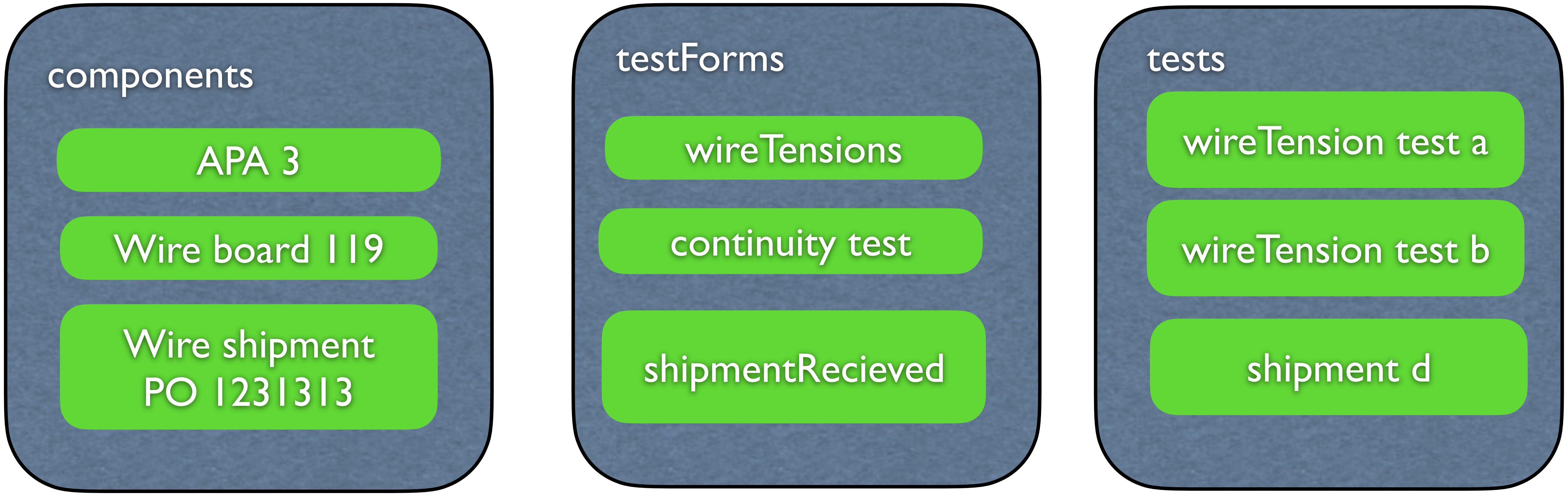
Effective corner radius opposite datum B

Most of the work on Sietch is going to developing these UI tools.

The most urgent challenge is finding the right way to get the data INTO the database …

in a way that is easy for the people actually doing the construction work.

Data retrieval is secondary.



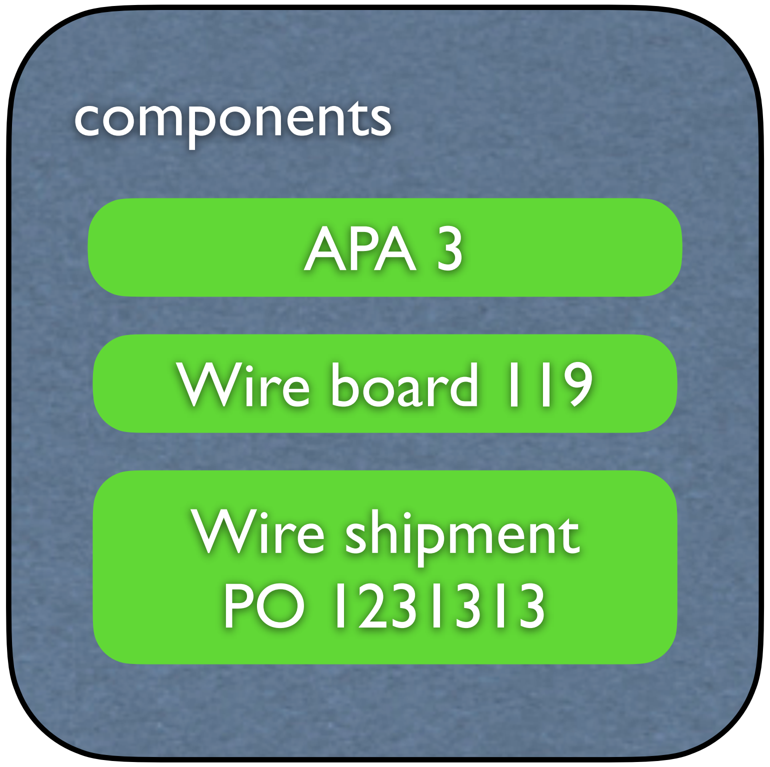
These are actual instances of filled out forms.

Each represents a procedure or test that was performed on a specific Component,

Must be keyed to a specific ComponentUUID

i.e. measured the wire tensions.

Important Sietch Collections



Component object schema:
{
  _id: (mongo primary key, autogenerated)
  ComponentUUID: “1f234e…” <binary UUID>,
  … any fields relevant…,
  type: “APA” <String>,
  name: “APA 3” <String, human readable>,
  effectiveDate: <date that this record takes effect>
  submit: {
    insertDate: <date this record went in>,
    ip: <ip address submitting this record>,
    user: <identifying info for user that submitted>,
    version: 2 <version of the form used>
    diff_from: <ObjectID> (key tolast version of this record),
  }
  version: 2, (Number of times this record has been changed)
}

Component Schema

Component Schema

actual data about the object

- serial number
- lot #
- notes
- what other components are attached or related

  e.g. chipInSlot1: <uuid>

Component object schema:

```
{
  _id: (mongo primary key, autogenerated)
  ComponentUUID: “1f234e…” <binary UUID>,
  … any fields relevant…,
  type: “APA” <String>,
  name: “APA 3” <String, human readable>,
  effectiveDate: <date that this record takes effect>
  submit: {
      insertDate: <date this record went in>,
      ip: <ip address submitting this record>,
      user: <identifying info for user that submitted>,
      version: 2 <version of the form used>
      diff_from: <ObjectID> (key tolast version of this
      record),
   }
   version: 2, (Number of times this record has been
   changed)
}
```

Component Schema

“Row” ID

Unique identifier, indexed.

Type of object

Date this data become true

insert date - for rollback

```
Component object schema:
{
  _id: (mongo primary key, autogenerated)
  ComponentUUID: “1f234e…” <binary UUID>,
  … any fields relevant…,
  type: “APA” <String>,
  name: “APA 3” <String, human readable>,
  effectiveDate: <date that this record takes effect>
  submit: {
     insertDate: <date this record went in>,
     ip: <ip address submitting this record>,
     user: <identifying info for user that submitted>,
     version: 2 <version of the form used>
     diff_from: <ObjectID> (key tolast version of this
     record),
   }
   version: 2, (Number of times this record has been
   changed)
}
```

Component Schema

Two boards are mounted on this APA on Monday. This record is the valid one on Monday or Tuesday

APA 3

```
Wire Board 1: 1111-…
Wire Board 2: 2222-…
effectiveDate: Monday
insertDate: Monday
version: 2
```

We change boards on Thursday, but there's a typo.

APA 3

```
Wire Board 1: aaaf-…
Wire Board 2: bbbb-…
effectiveDate: Wednesday
insertDate: Wednesday
version: 3
```

An error is discovered; later someone fixes the ID, given the same effectiveDate and bigger version, so this one is the one retrieved

APA 3

```
Wire Board 1: aaaa-…
Wire Board 2: bbbb-…
effectiveDate: Wednesday
insertDate: Friday
version: 2
```

Component Schema

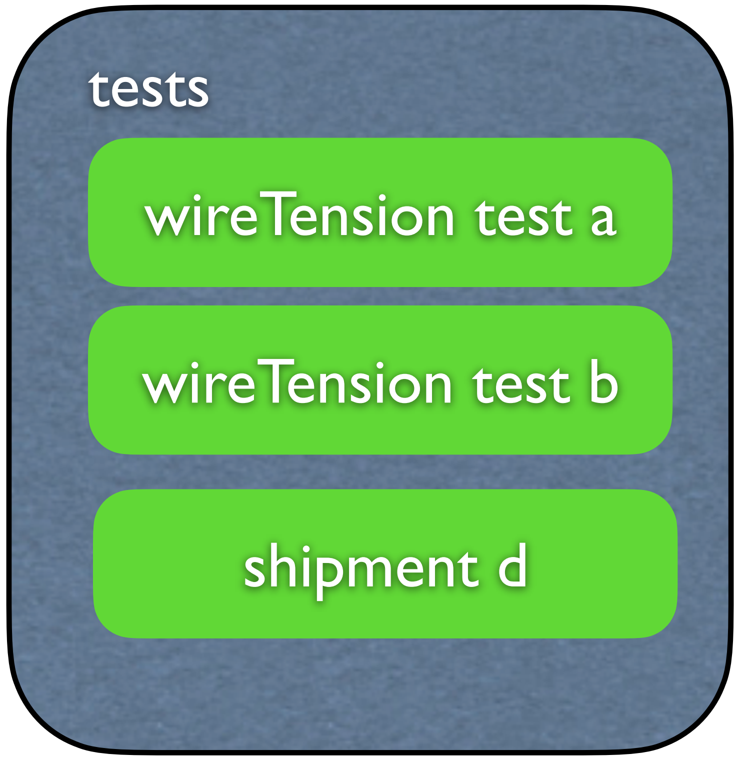
If we rollback the database to Thursday, we get this version as the latest, which represents the DB in that state.

APA 3
```
Wire Board 1: 1111-…
Wire Board 2: 2222-…
effectiveDate: Monday
insertDate: Monday
version: 2
```

APA 3
```
Wire Board 1: aaaf-…
Wire Board 2: bbbb-…
effectiveDate: Wednesday
insertDate: Wednesday
version: 3
```

APA 3
```
Wire Board 1: aaaa-…
Wire Board 2: bbbb-…
effectiveDate: Wednesday
insertDate: Friday
version: 2
```

```
Test Schema
{
    _id: (mongo primary key, autogenerated)
    ComponentUUID: "1f234e…" <binary UUID>,
    form_id: <String> which_test_was_done,
    state: "submitted" or "draft"
    insertDate: <Date> data was submitted
    ip: <ip address submitting this record>,
    user: <identifying info for user that submitted>,
    data: {
        … any JSON object, defined by the form
        … can contain urls to saved files …
    }
}
```



Test Schema

UUID of APA 3

```
insertDate: Monday
data: {width:100}
```

UUID of APA 3

```
insertDate: Tuesday
data: {width:101}
```

UUID of APA 3

```
insertDate: Wednesday
data: {width:103}
```

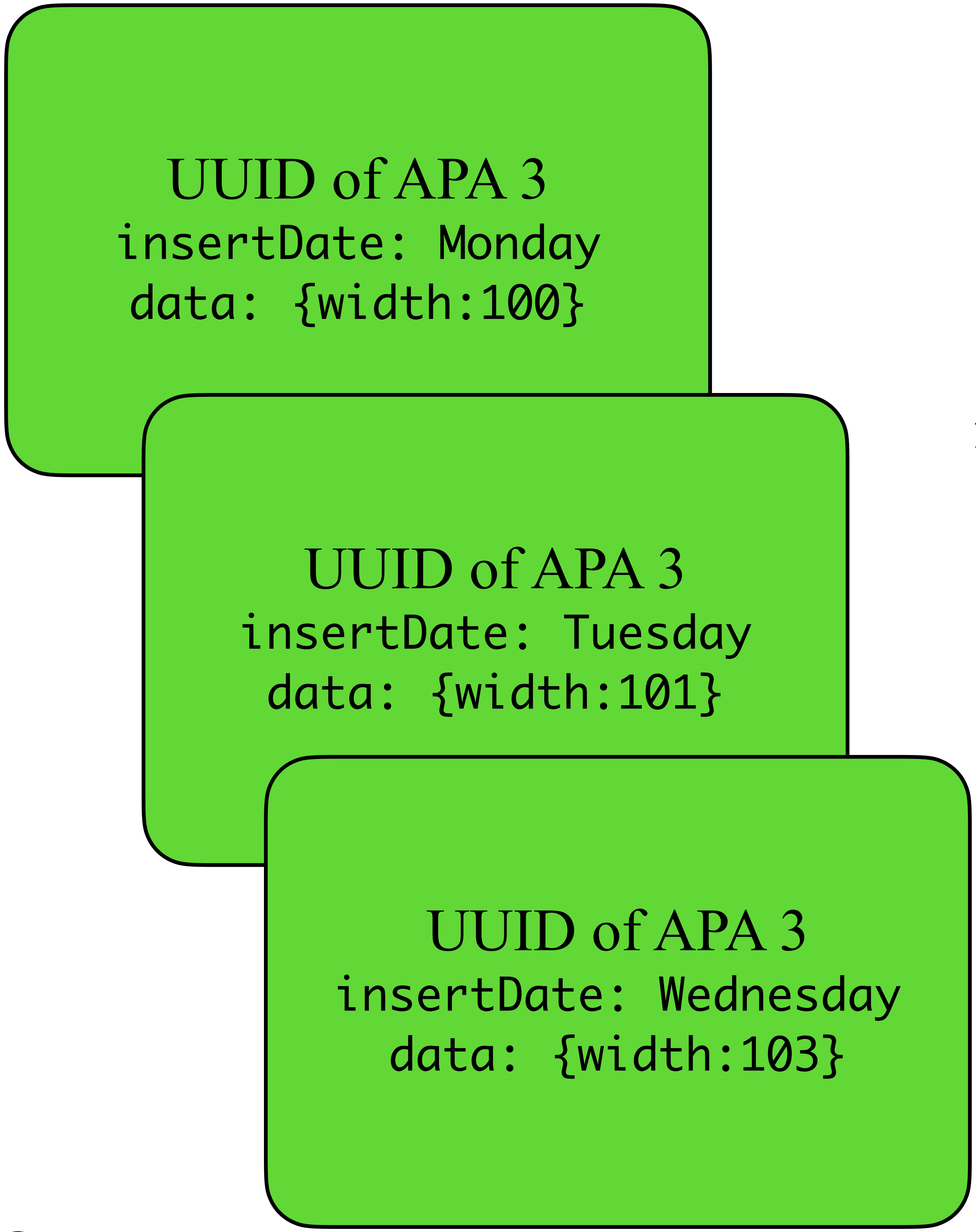
Test Schema

Must reference a specific Component.

“draft” versions are not considered real, but are there to allow interrupted workflow

Once submitted, record is never changed.

(Could add exception to this: e.g. allow flags to mark test as ‘bad’ so it doesn’t show up in searches)



UUID of APA 3
```
insertDate: Monday
data: {width:100}
```

UUID of APA 3
```
insertDate: Tuesday
data: {width:101}
```

UUID of APA 3
```
insertDate: Wednesday
data: {width:103}
```

The width of this frame was measured three times.

The last measurement is assumedly the most relevant, but more selection criteria could be assigned.

All test results are shown for this APA’s history

Test Schema

Even grittier nitty-gritty

components

componentForm

Form describing component data

testForms

tests

workflowForms

jobs

The same as 'tests' but does not require unique Component UUID.

Used like VKS: define a series of tasks that can be done, which may result in creation of new components and tests

gridFS
bit-bucket to allow upload of binary files

m2mUsers
permissions granted to for machine-to-machine access

Database is only accessed through website or through HTTP API

Website: activates require user permissions.
Users are maintained by the auth0.com service
Allows authentication against google Fermilab SERVICES account.
→ Users create their own accounts, and can either have permissions granted by an admin, or we can use a generic password for account upgrading.
→ All transactions are logged with user and ip, so we can weed out any bad actors.

API allows direct submission of data from a computer program or script

- Authenticates with secret id, checked against m2mUsers
- Accesses with JWT token. Data is tagged with this 'machine' userId and email
- Easy to use from any high-level language
  - Or from low-level language with http library

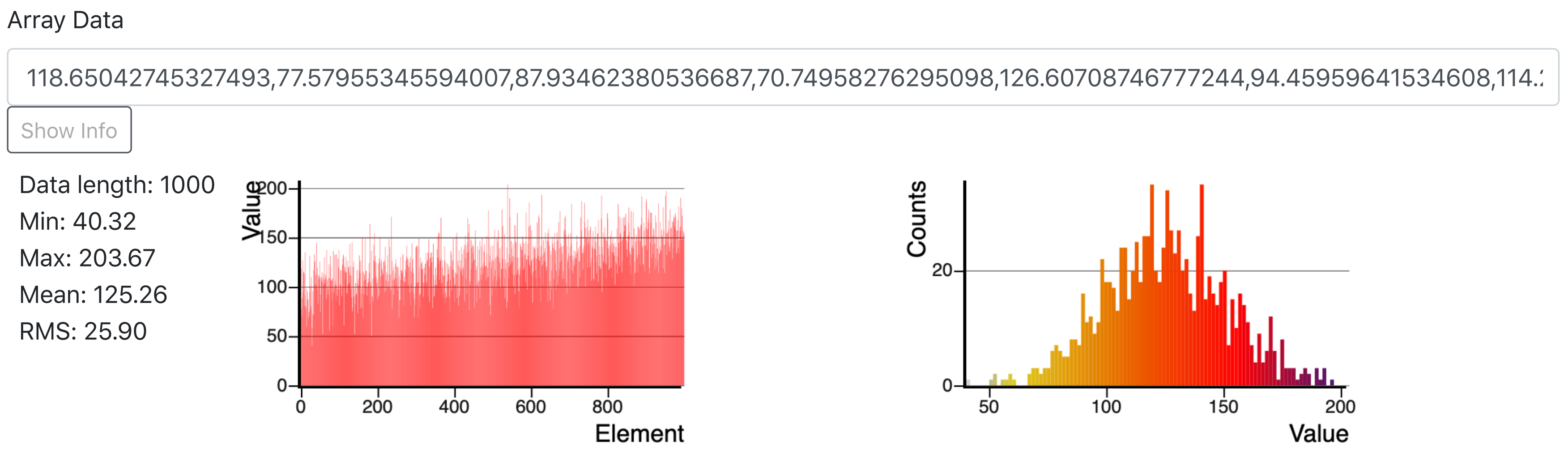
m2mUsers
permissions granted to for machine-to-machine access

Authorization/Authentication

- A lot of work is going into the UI / Web interface

Outstanding design priorities

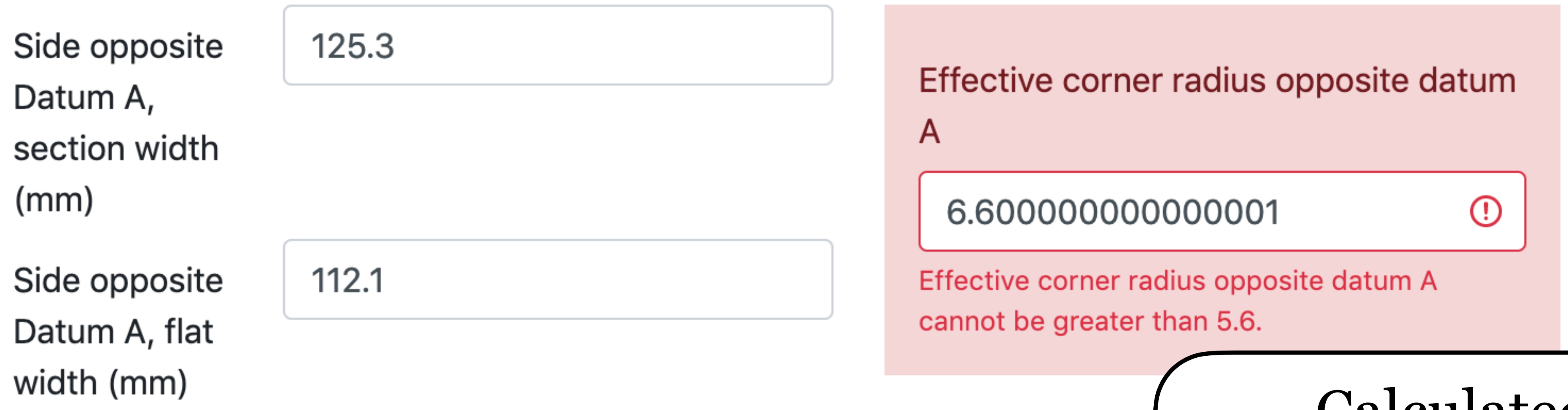
Useful views of array data (i.e. tensions)



Array Data

118.65042745327493,77.57955345594007,87.93462380536687,70.74958276295098,126.60708746777244,94.45959641534608,114.

Data length: 1000
Min: 40.32
Max: 203.67
Mean: 125.26
RMS: 25.90



Side opposite Datum A, section width (mm)	125.3
Side opposite Datum A, flat width (mm)	112.1

Effective corner radius opposite datum A

6.6000000000000001

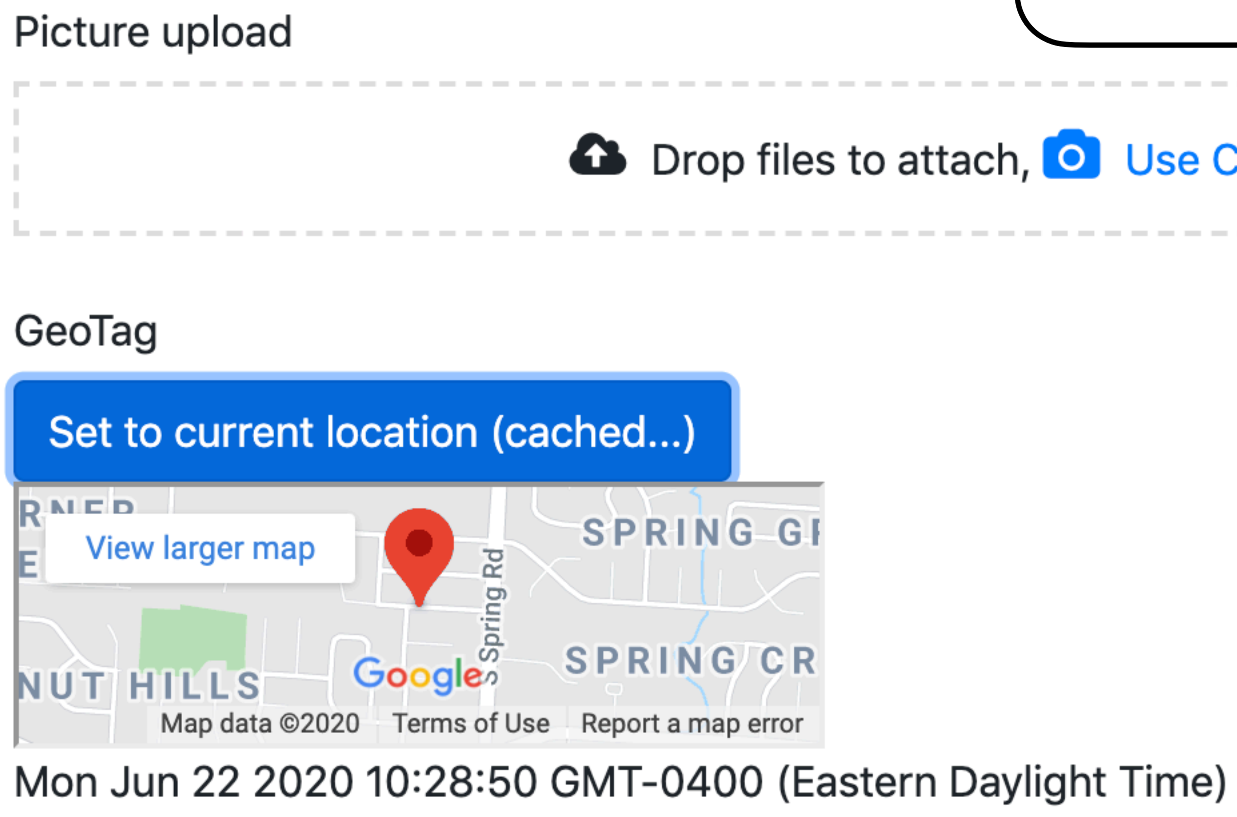
Effective corner radius opposite datum A cannot be greater than 5.6.

Calculated values.

Automated warnings for values-out-of-spec to make pass/fail tests easier.

User Interface (Web)

Uploads or pictures taken


Picture upload

Drop files to attach, Use Camera, or browse

GeoTag

Set to current location (cached...)

Mon Jun 22 2020 10:28:50 GMT-0400 (Eastern Daylight Time)

Location tagging

("Which side of the warehouse did we last see this board?")

User Interface (Web)

http://sietch.xyz/eb833bd0-a9c1-11ea-bbcc-8dda7beedcce

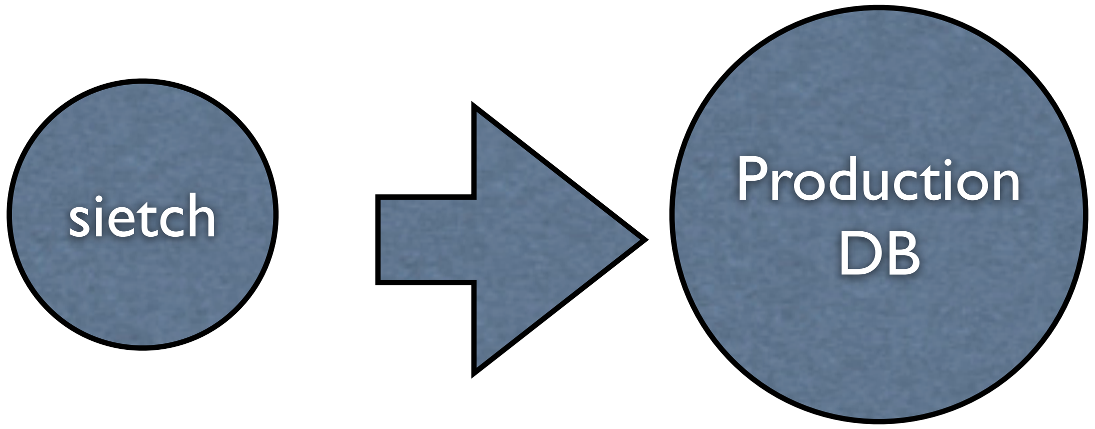
http://sietch.xyz/eb833bd0-a9c1-11ea-bbcc-8dda7beedcce

http://sietch.xyz/eb833bd0-a9c1-11ea-bbcc-8dda7beedcce

http://sietch.xyz/eb833bd0-a9c1-11ea-bbcc-8dda7beedcce

Virtual “traveller”
for every object in the database

User Interface (Web)



We write a script to export relevant data to the analysis databases

- e.g. for each APA, find the most recent valid wireTension measurements
- concatenate wireTension measurement into whole-APA
- write to official DB

Do this once, maybe ~few times.

Most data in Sietch will not be relevant for analysis.

Some data in Sietch MAY be relevant for geo_id info in the HW database.

What happens to the data?

- A lot of work is going into the UI / Web interface
  - Data entry(e.g. How does a tech enter the 80 different measurements of the procured steel pipes?)
  - Replace VKS for serving work instructions and checklists.
- "Travellers" and/or "Virtual Travelers"
- We need data **entry** to be fully-functional, fully-tested by October.
- Want to do a beta test soon, use it to enter steel procurement (which is low priority for long-term data storage, but a highly effective use case to get designs working)

Outstanding design priorities

- Are the metadata schemas sufficient?
  Is this design good?

- What is the scope? I have big ambitions.
  - Would like to see this used for all construction, not just APA
  - I think a core strategy like this makes a lot of sense going past construction

- This overlaps with a lot of other people's work, may incorporate them (via m2m API)

- Very little search/retrieval built yet.

Outstanding design issues