Developing a modern Kubernetes based Study Management Platform

Hugo Hiden

Stephen Dowsland

Paul Watson

Fundamentals of studies

- Data arriving from multiple sources
  - Medical devices
  - Clinical assessments
  - Environmental data
  - Derived from raw data processing
- Needs to be collated and organised
  - Associated with a Patient
  - Made available for analysis
  - Potentially edited after collection
  - Auditing and traceability critical
- Over time, things will change
  - Updated devices
  - New data sources
- Different every time

Platform design goals

- Studies, Participants, Devices, Forms
  - 1st class citizens
  - Efficient and scalable ingest from multiple sources
- Documented design
  - Design documentation
  - Unit testing
- Modern architecture
  - Container based
  - Cloud focussed
  - Clear separation of components
- Flexible
  - Customize behaviour
- External data access
  - API support for developers
  - Documentation and support
- Auditing
  - Capture entire patient journey and events

How to define / setup a study

- Data collection requirements
  - How much is there?
  - How often?
  - What form?
- Scale / performance needs
- External connectivity
  - Really important to get this right
- Custom behaviour
  - Study-specific actions
- Reporting requirements

Data collection requirements

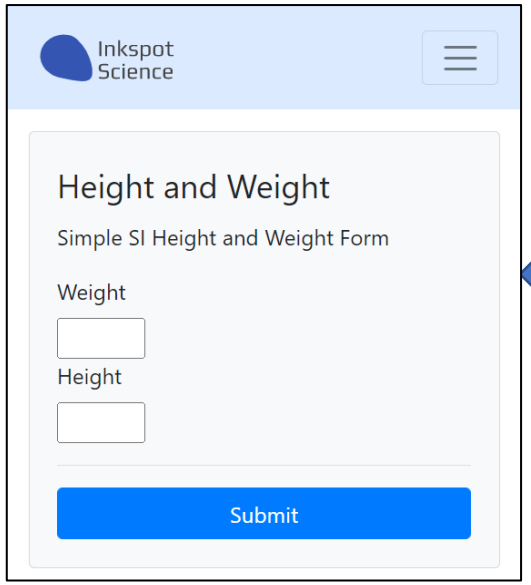
- Entire system oriented around Data
  - Form submissions
  - Streams of measurements
- Every piece of data is structured – a DataType
  - Defined system-wide
  - Shared between studies
- Use JSON data
  - Can represent anything
  - Widely supported
  - Can be validated

```
{
    "height": 1.76,
    "weight": 78.6
}
```

https://www.json.org/json-en.html

Data collection requirements

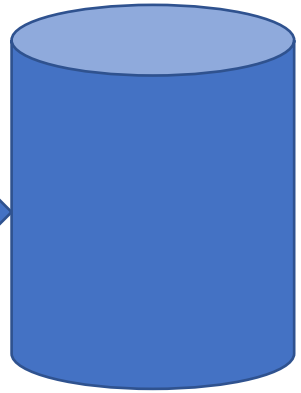
Form Generation



Common Schema

```
{
    "$id": “metrics.json",
    "$schema": "http://json-schema.org/draft-07/schema#",
    "title": "Height and Weight",
    "description": "Simple SI Height and Weight Form",
    "required": [ "height", "weight" ],
    "type": "object",
    "properties": {
      "height": {
        "title": "Height",
        "type": "number",
        "minimum": 1.0,
        "maximum": 2.4
      },
      "weight": {
        "title": "Weight",
        "type": "number",
        "minimum": 20,
        "maximum": 200
      }
    }
  }
```

Population of Data Sets



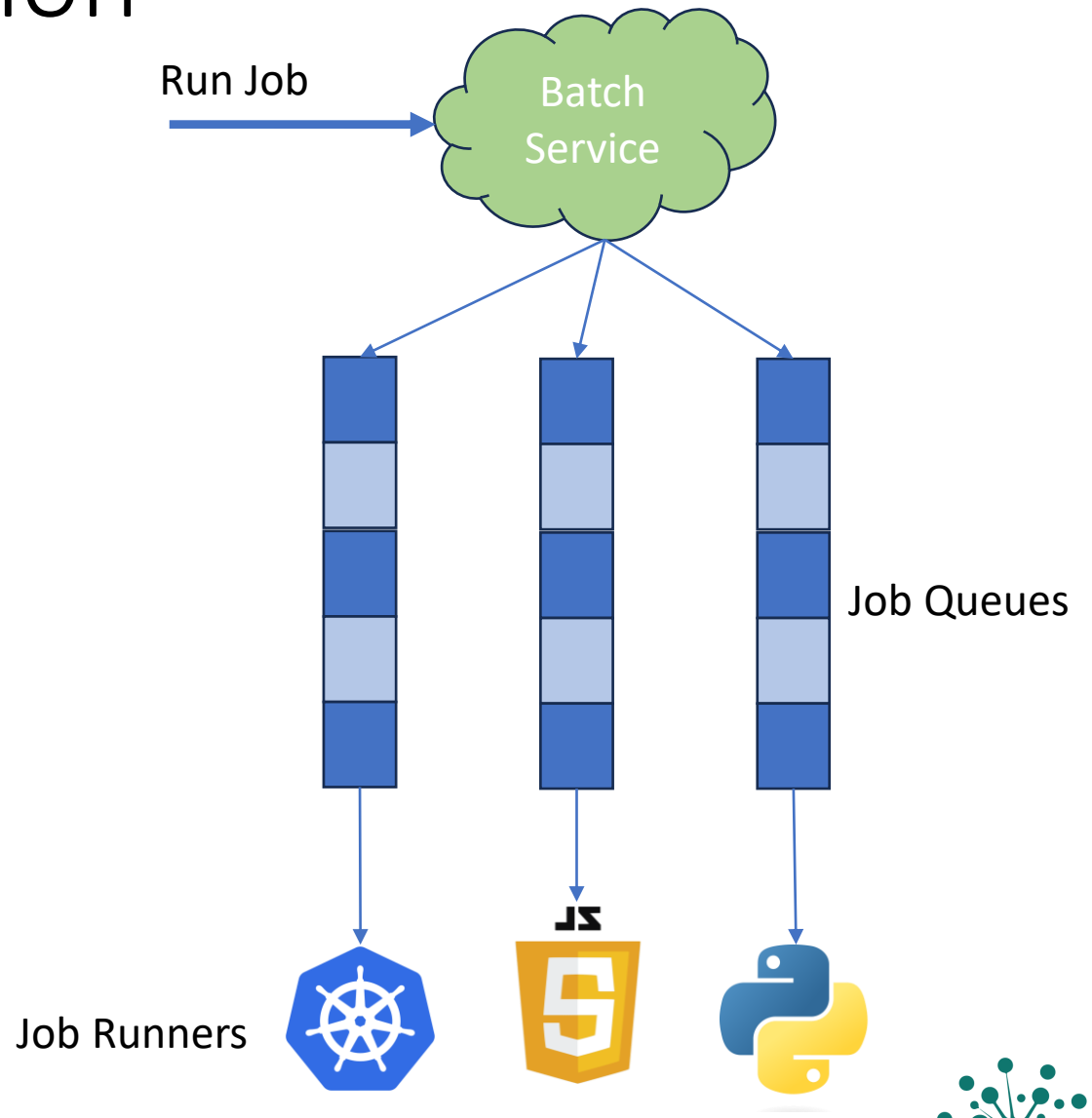
Form Validation

Performance and scalability

- API / Structure service
  - Study structure, folders and files
- Token Service
  - Login service or OAuth (keycloak)
- Form / Property / Variable service
  - Stores / queries participant forms and study data
- Audit Capture Service
  - Project level audit log
  - NOT middleware logging
- Batch Service
  - Run custom code to tailor behaviour / manage integrations
  - Kubernetes jobs, Javascript / Python actions
  - Timescale ms -> hours
- Trigger Service
  - Listens to audit events
  - Can run jobs
- Webhook Service
  - Provides integration with other systems

External systems / customisation

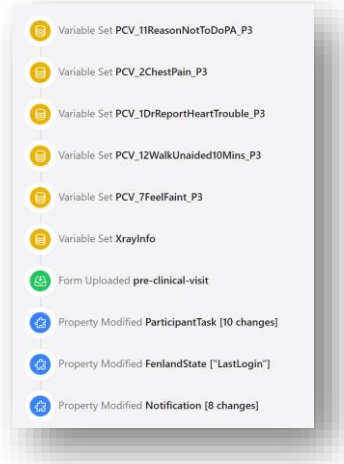
- Most studies need additional code
  - Data analysis
  - Data export
  - Study / participant setup
  - Co-ordination
- Developed a simple batch job framework
- Jobs defined by:
  - Job type
  - Reference to the code to run
  - Parameters / file references
- Messages put onto a queue
  - Jobs executed by queue listeners
- Allows flexibility
  - Quick scripts
  - Multi-hour jobs needing 16GB+ RAM
  - Manual, scheduled or triggered execution



Auditing and triggering

- Every action must be captured and logged
  - Important for medical regulators
  - Removal of participants - GDPR
  - Customisation

Audit service stores events and provides query interface



External Systems

capacitor

JavaScript Client Library

.NET Client Library

Python Client Library

jupyter

Java Client Library

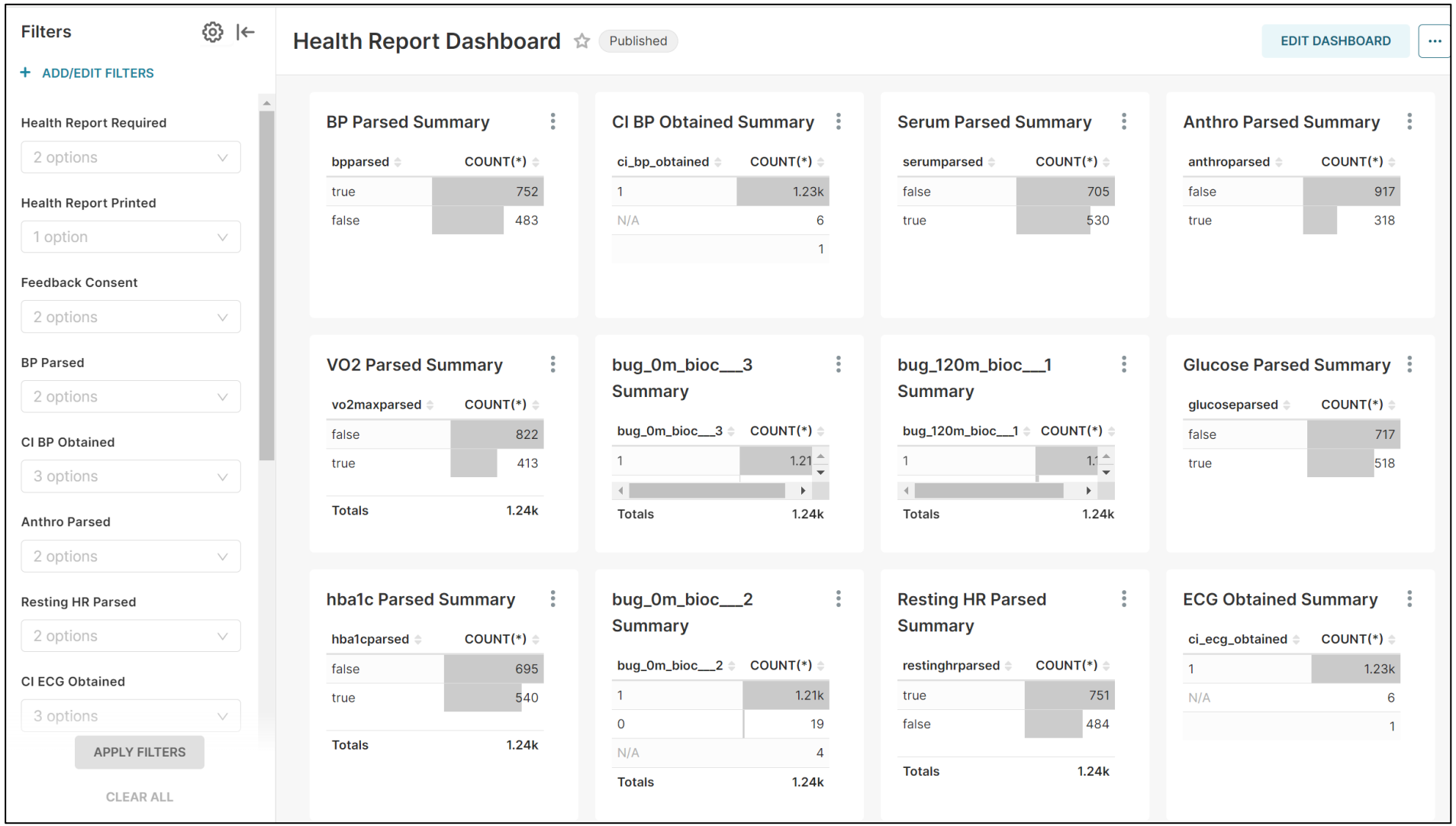
```
hugoh@LAPTOP-DRM0PKH5 MINGW64 /c/work/git/osm/osm-platform/clients/cli (hugo-dev)
$ ./osm2 project list
```

id	name	code	description
4	IngestTest	IT001	
437	ParticipantIngestProject	PINGEST1	A PRoject

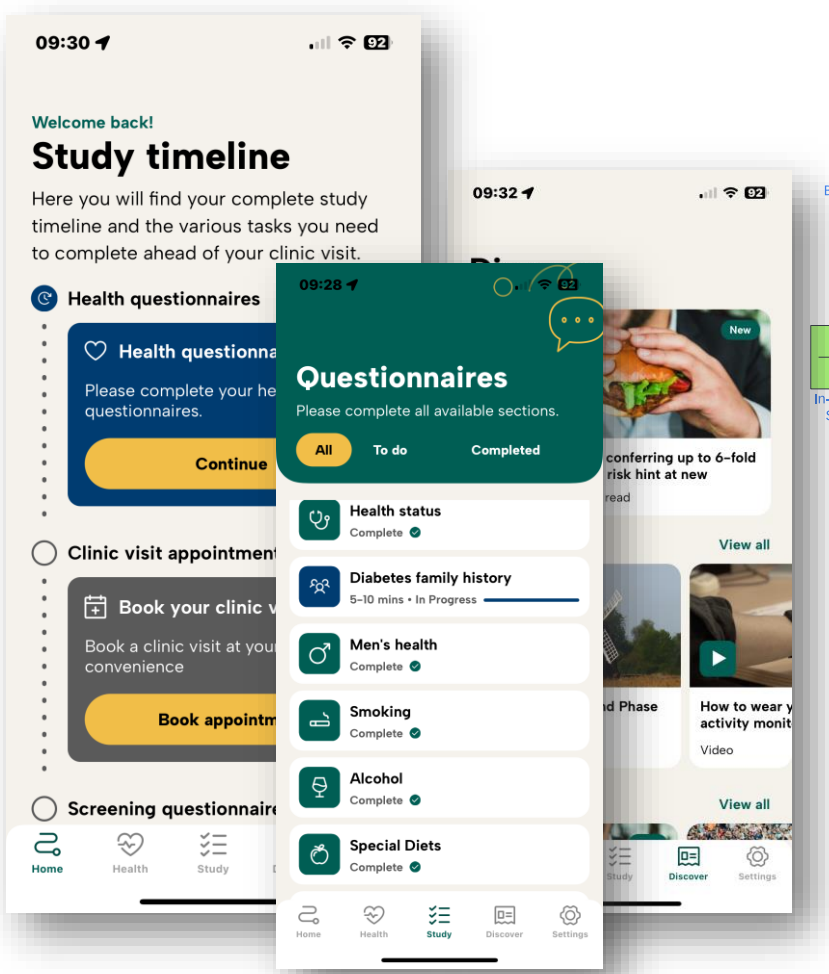
```
hugoh@LAPTOP-DRM0PKH5 MINGW64 /c/work/git/osm/osm-platform/clients/cli (hugo-dev)
$ ./osm2 project ep list -i 437
```

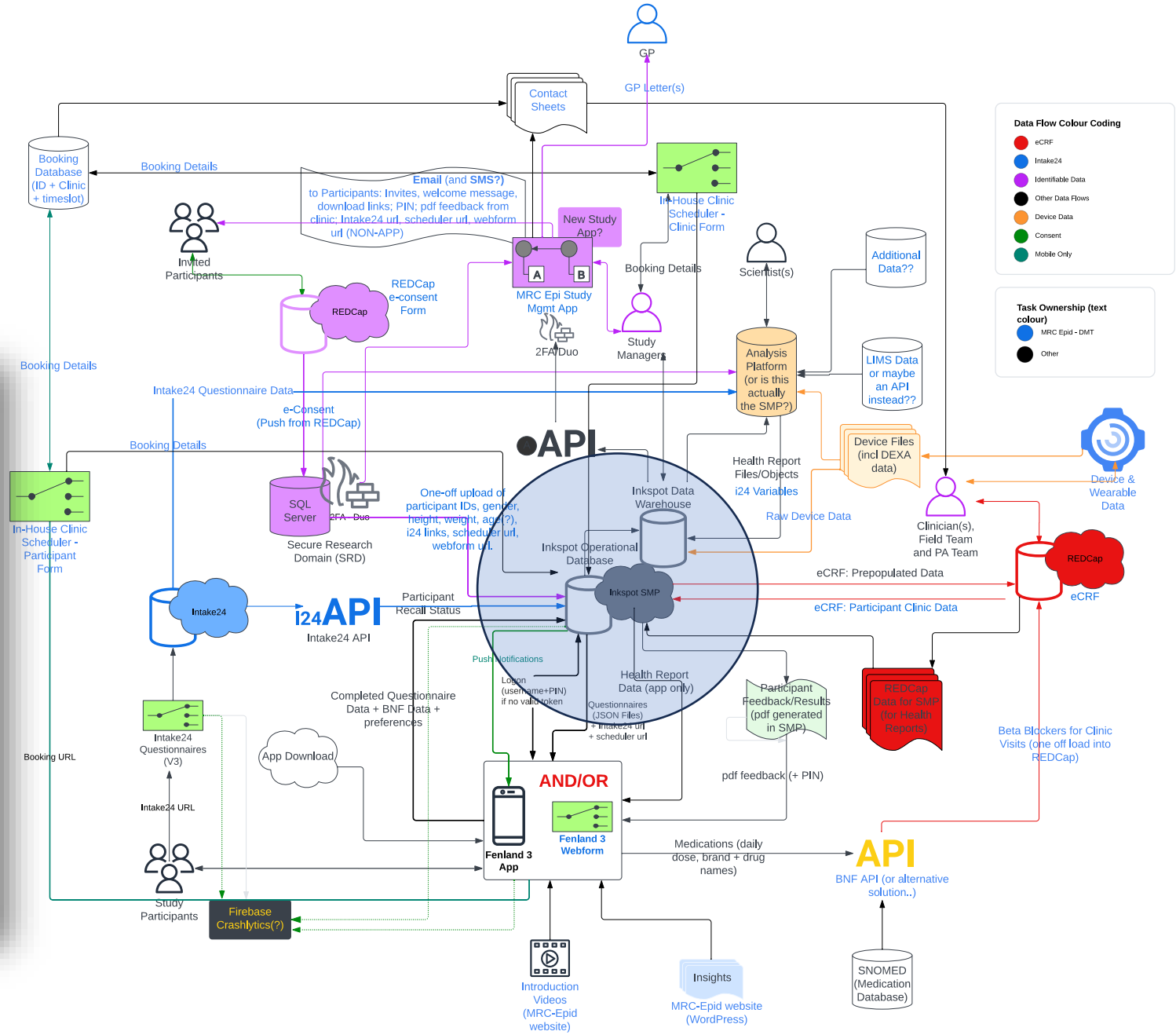
id	endpointTypeId	description
443	mongo	Mongo Database containing partipant data

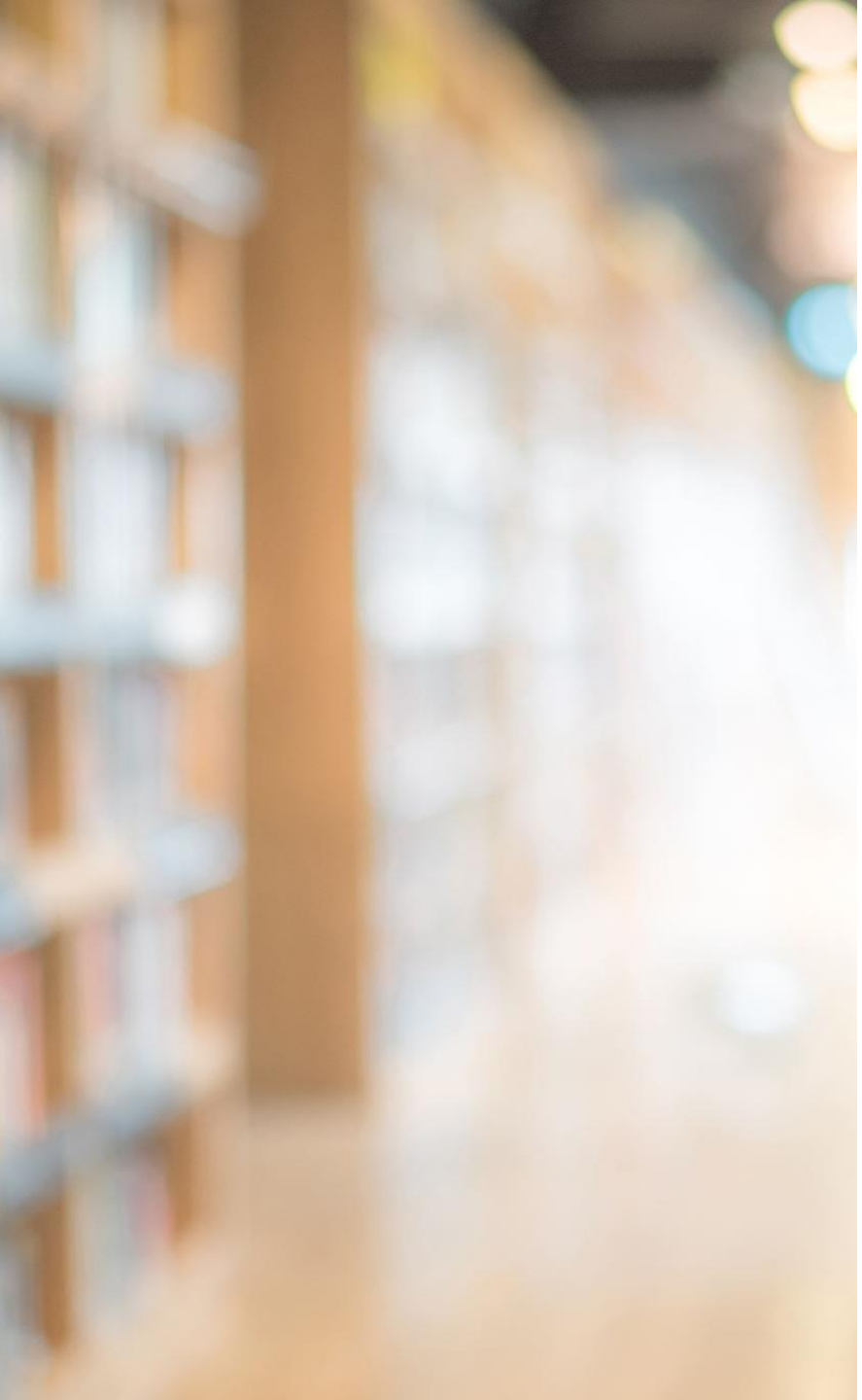
Reporting - Apache superset



Latest application







Lessons Learned

- Kubernetes solves some scalability challenges...
  - Database scalability
  - Monitoring: memory, CPU, errors
  - A more complex environment
  - Why doesn't it work?
- Make sure you understand interactions
  - How many external systems
  - Can you connect to them
  - Client libraries etc etc
- Be ready for a disaster