FUTURE RUN REGISTRY

Fabio Espinosa

Contents

Users of Run Registry

- Online Shifters
- Offline Shifters
- DC Expert
- API users

Problems reported by users

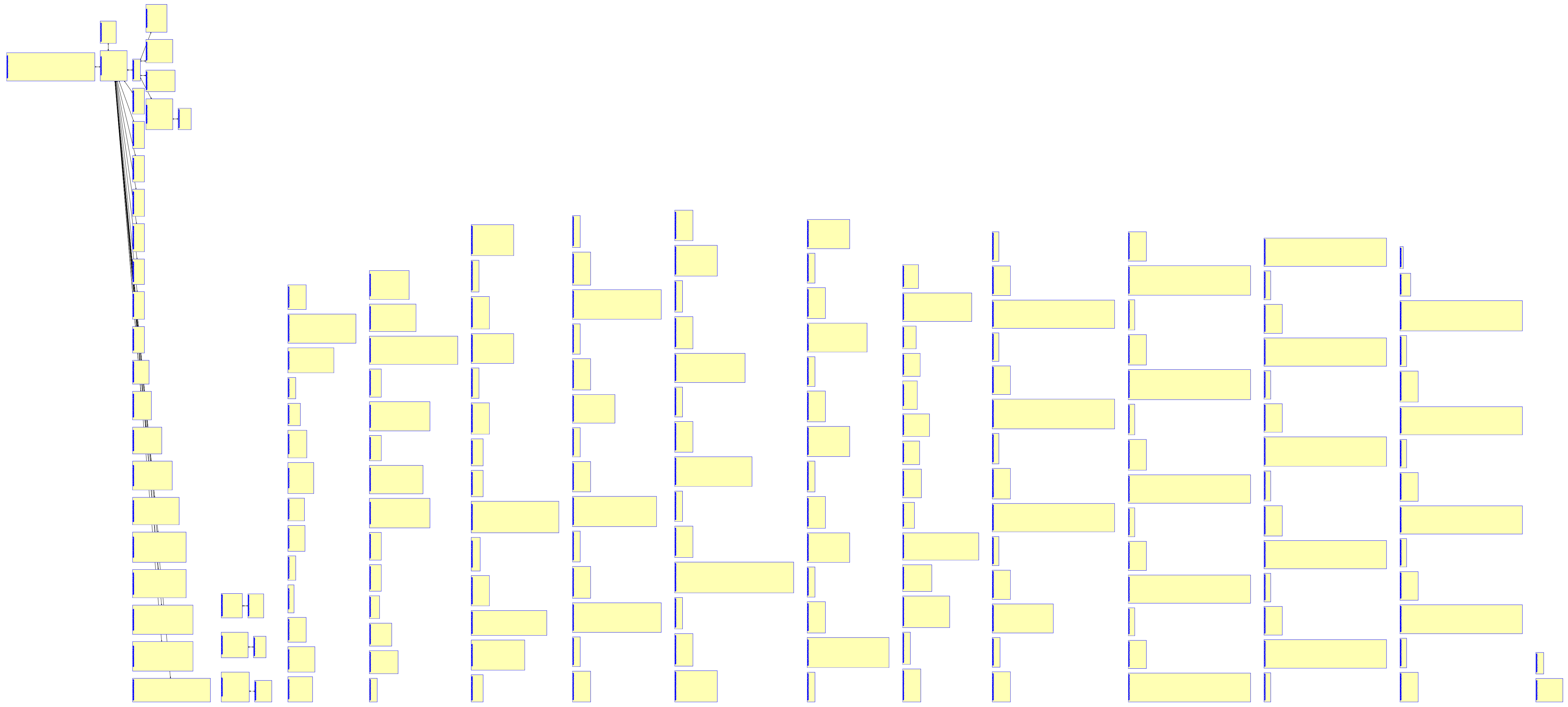
1. Delays of runs appearing in the system (sometimes in the size of hours). Cause: Synchronization is needed between two databases (online-offline).
2. Complicated GUI. Too much unused functionality.
3. Era is needed to be set manually. There can be more automation.
4. RR is slow and users sometimes don’t know if it is loading or stuck.
5. Slow fixes by maintainers.
6. Certificate issues and logins (authentication).

Other Problems

7. Difficult maintainability. The db schema design is extremely complicated. There are over 260 tables between the two dbs. This makes a new developer to take ages to make any change in the current system.
8. Outdated technology. RR was built using the best web frameworks of 2012.
9. Three web apps that can fit into one. One that apparently nobody uses (User).

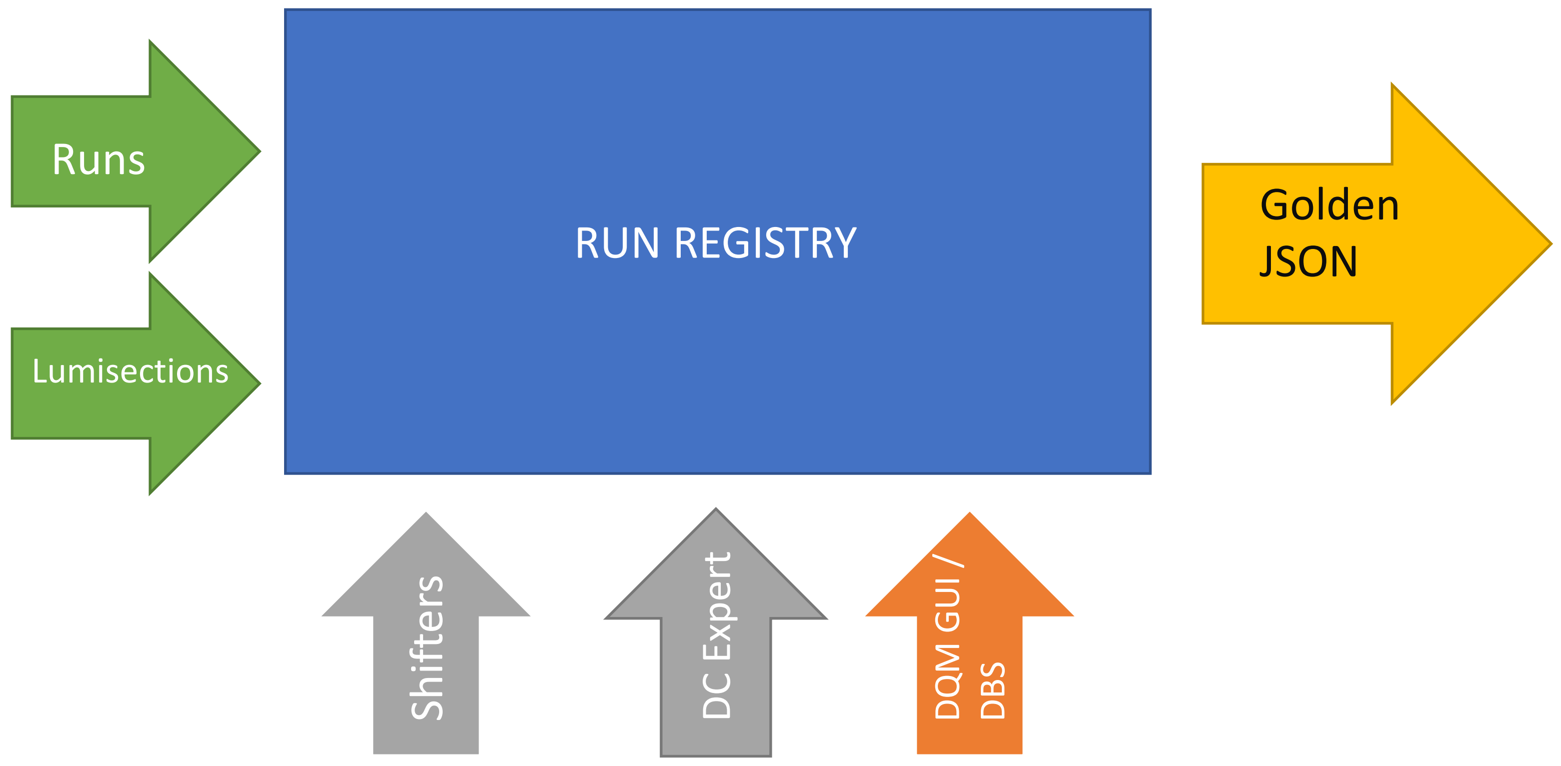
Actual Db Schema for Online RR

Actual Db Schema for Offline RR



PROPOSAL

RR Weekly Workflow



Legend:

🟩	Data input
⬜	Human input
🟧	Operation input
🟨	Data output

Overview for Future RR

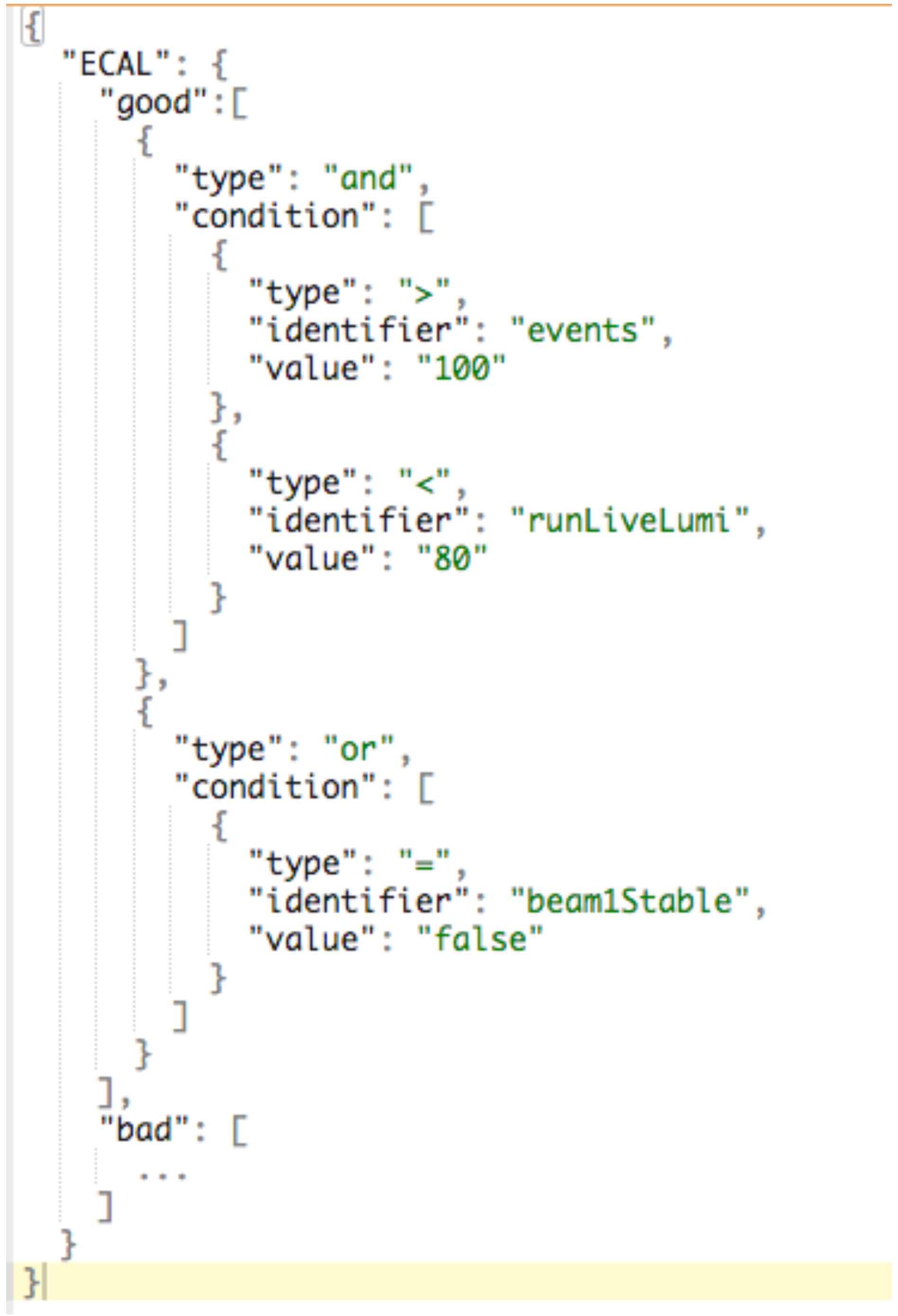
- Must be developed with state of the art technologies.
- Must be maintainable, well documented and design must be as simple as possible.
- Must allow lots of automation via API (API completely separated from front end).
- Must have great performance and be user-friendly.
- Possible human errors must be prevented by design.
- Information should be allowed for everyone to see, but only for authorized users to edit/delete.
- Must not be a copy of current RR, but a complete new solution.

New Design Major Challenges

1. Handling triggers, make them work as soon as they should, store them and make them user-friendly to write.
2. User authentication working with e-groups.
3. Make existing data in current RR fit new RR. This includes 69 tables for Online db and 191 tables for Offline db. Have to go through each table importing data and exporting to new schema. (Do we actually need old data).
4. Handling possible immutable database schema design.

Handling Triggers viable solution

- Can be easily stored
- Are not hard to understand
- Can be easily validated
- Requires a learning curve
- Are easily evaluated at runtime
- Must have consistency
- A JSON editor is not the friendliest thing, but is not changed frequently.



```
{
  "ECAL": {
    "good":[
      {
        "type": "and",
        "condition": [
          {
            "type": ">",
            "identifier": "events",
            "value": "100"
          },
          {
            "type": "<",
            "identifier": "runLiveLumi",
            "value": "80"
          }
        ]
      },
      {
        "type": "or",
        "condition": [
          {
            "type": "=",
            "identifier": "beam1Stable",
            "value": "false"
          }
        ]
      }
    ],
    "bad": [
      ...
    ]
  }
}
```

Database PostgreSQL vs Oracle

PostgreSQL

Pros:

- Open source
- Vastly more used in real world
- Has support in managed DBS at CERN
- Very popular in developer community
- Notify() function built in
- Built in support for JSON storage

Cons:

- Will not be able to query WBM/OMS data source directly.

Oracle

Pros:

- Has support as managed DBS at CERN
- Current WBM uses Oracle
- More developers in CERN

Cons:

- Closed source
- Fewer developers worldwide know it.

DB Schema

Event sourcing schema

Pros:

- It will have a built in log “for free”
- Database can be viewed at any point in time

Cons:

- It will be my first time using such schema.
- Unknown time horizon for learning curve.

Normal “relational” schema

Pros:

- It is the most common way to use a relational db.

Cons:

- It is problematic for creating snapshots, or restoring to any point in time
- Creating a log, and user actions is not trivial.

Back End (REST API)

Python: current language and probably the chosen language (with Flask framework):

Pros:

- Is used throughout CERN.
- Acquiring talent of Python developers is easy.

Cons:

- The new maintainers will also need to know JavaScript to modify front end.

JavaScript (Node.js)

Pros:

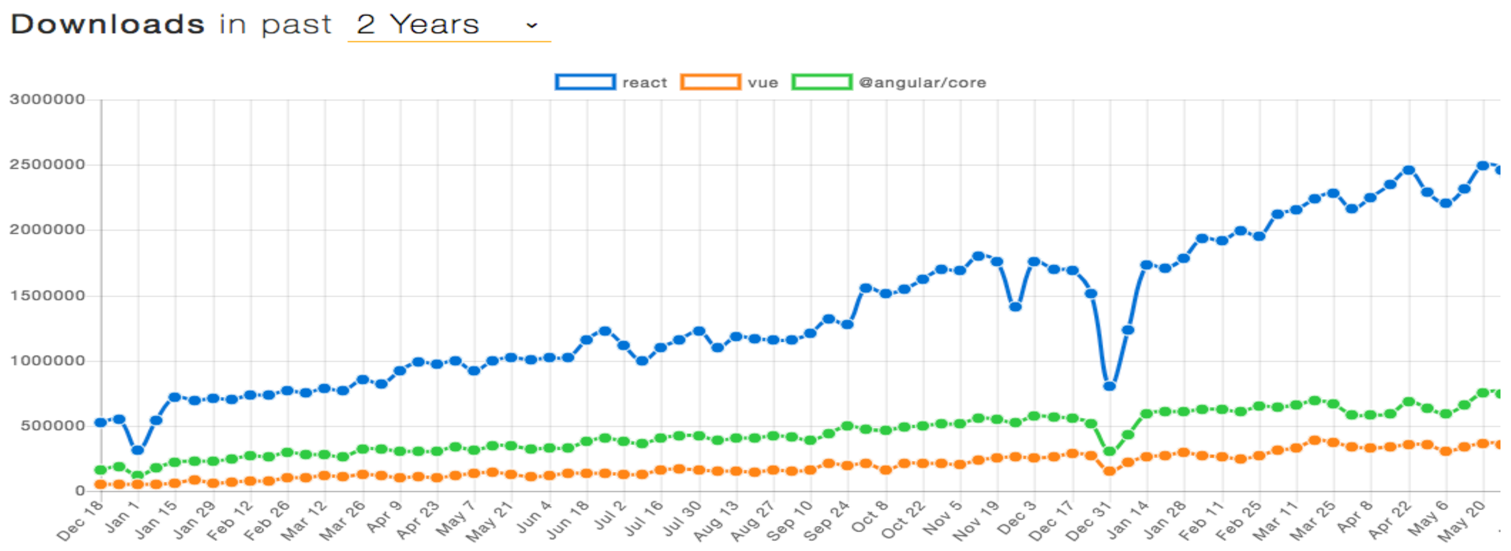
- Has the advantage that the whole stack of Run Registry will be in only one prog. language.

Cons:

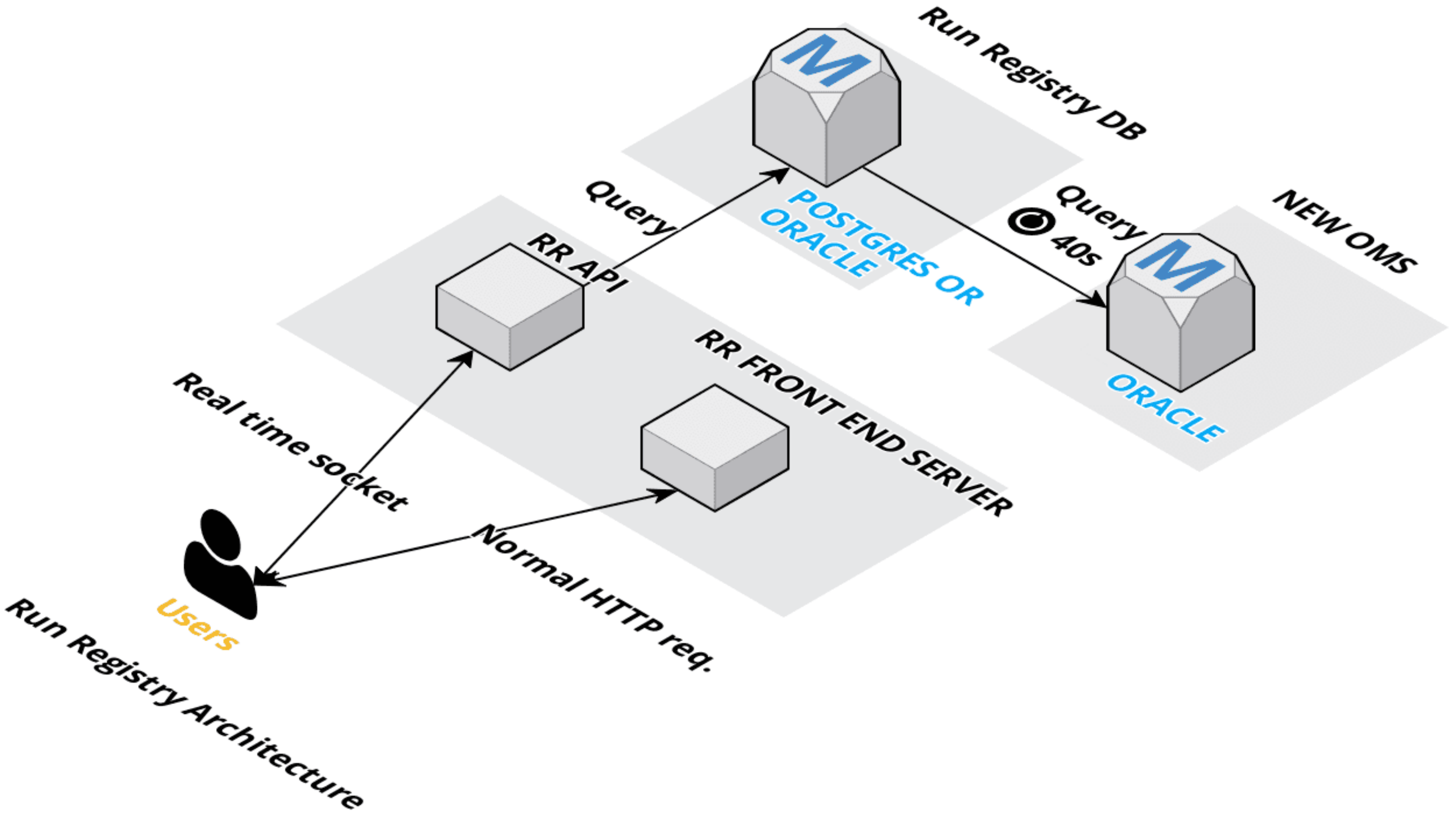
- It is not as maintainable as Python

Front End

- React.js for interactivity. Hands down the most popular js library/framework for front-end in the world right now. (Already used in other CMS projects like OMS
- Easy integration of real-time architecture.
- Other options: Vue.js, Angular or Backbone.



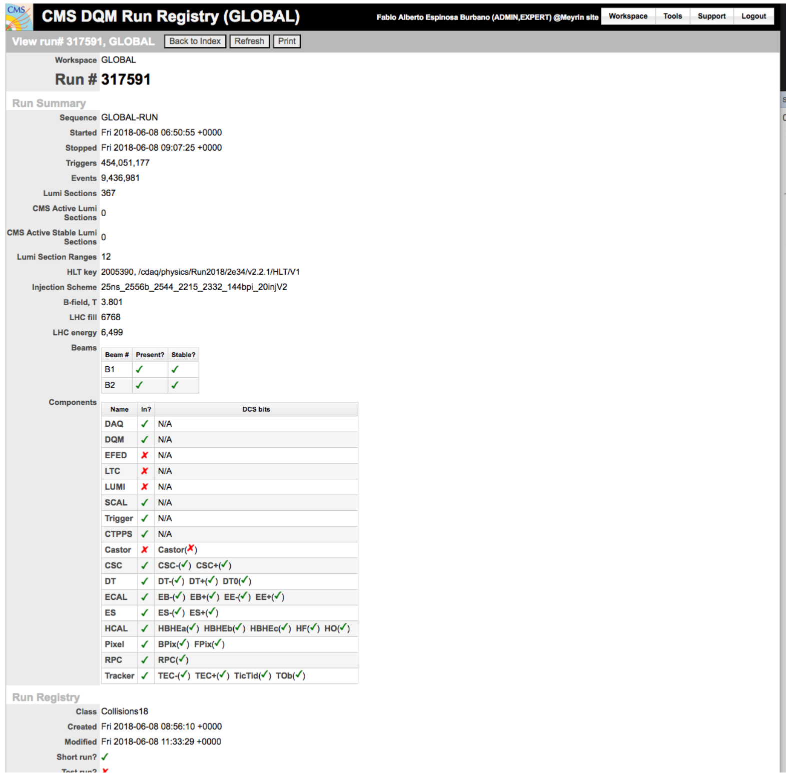
PROPOSED ARCHITECTURE



Additional Users requests

- Link to elogs for every run, the elogs must be filtered by run and by timestamp.
- Link to DQM GUI for every run.
- Option to upload text file that contemplates changes between component status (Change in batch) or WYSIWYG editor.
- Bookmarkability. Make URL contain filtered data table.
- Preserve functionality of data filtering in current tables. Mostly ”like” functionality.

What a current shifter screen looks like



CMS DQM Run Registry (GLOBAL)

Fabio Alberto Espinosa Burbano (ADMIN,EXPERT) @Meyrin site

View run# 317591, GLOBAL

Workspace	GLOBAL
Run #	317591

Run Summary

Sequence	GLOBAL-RUN
Started	Fri 2018-06-08 06:50:55 +0000
Stopped	Fri 2018-06-08 09:07:25 +0000
Triggers	454,051,177
Events	9,436,981
Lumi Sections	367
CMS Active Lumi Sections	0
CMS Active Stable Lumi Sections	0
Lumi Section Ranges	12
HLT key	2005390, /cdaq/physics/Run2018/2e34/v2.2.1/HLT/V1
Injection Scheme	25ns_2556b_2544_2215_2332_144bpi_20injV2
B-field, T	3.801
LHC fill	6768
LHC energy	6,499

Beams

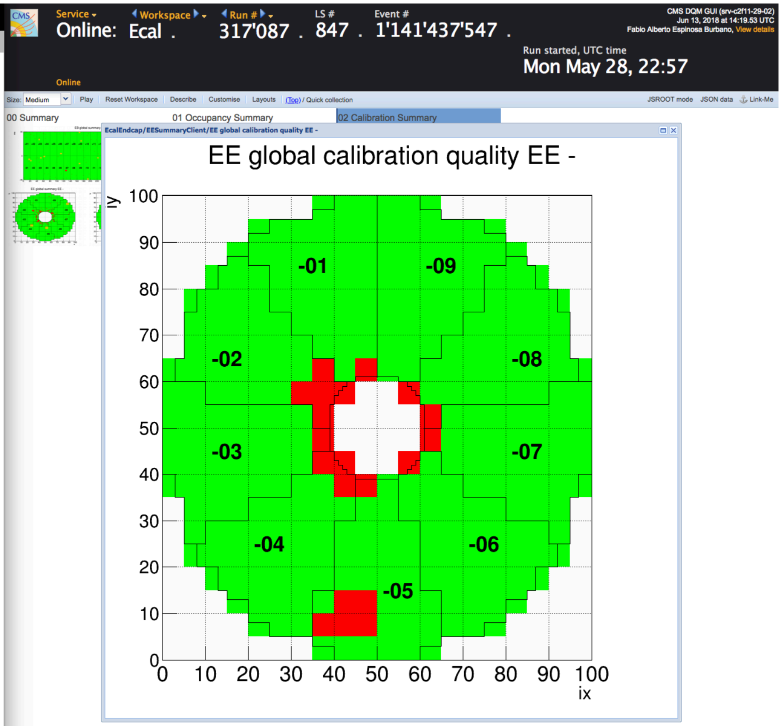
Beam #	Present?	Stable?
B1	✓	✓
B2	✓	✓

Components

Name	In?	DCS bits
DAQ	✓	N/A
DQM	✓	N/A
EFED	✗	N/A
LTC	✗	N/A
LUMI	✗	N/A
SCAL	✓	N/A
Trigger	✓	N/A
CTPPS	✓	N/A
Castor	✗	Castor(✗)
CSC	✓	CSC-(✓) CSC+(✓)
DT	✓	DT-(✓) DT+(✓) DT0(✓)
ECAL	✓	EB-(✓) EB+(✓) EE-(✓) EE+(✓)
ES	✓	ES-(✓) ES+(✓)
HCAL	✓	HBHEa(✓) HBHEb(✓) HBHEc(✓) HF(✓) HO(✓)
Pixel	✓	BPix(✓) FPix(✓)
RPC	✓	RPC(✓)
Tracker	✓	TEC-(✓) TEC+(✓) TicTid(✓) TOb(✓)

Run Registry

Class	Collisions18
Created	Fri 2018-06-08 08:56:10 +0000
Modified	Fri 2018-06-08 11:33:29 +0000
Short run?	✓
Test run?	✗



Service: Online: Workspace: Ecal . Run # 317'087 . LS # 847 . Event # 1'141'437'547 .

CMS DQM GUI (srv-c2f11-29-02)
Jun 13, 2018 at 14:19.53 UTC
Fabio Alberto Espinosa Burbano, View details

Run started, UTC time
Mon May 28, 22:57

Size: Medium

EcalEndcap/EESummaryClient/EE global calibration quality EE -

Two tabs that can be integrated into one -> Make DQM GUI available in RR. There will be no way that "human mistake" occurs in selecting same run in both applications.

Other ideas

- Implement a real time chat inside RR to facilitate collaboration between offline physics teams.
- Enforce deadline of physics groups to set components good/bad Monday midnight for weekly runs. Controversial.
- Integrate DQM GUI into RR. Make it easy so that in the future, machine learning can fit in RR.