Providing validated, templated and richer metadata using a bidirectional conversion between JSON and iRODS AVUs.

Paul van Schayck, Ton Smeele, Daniel Theunissen and Lazlo Westerhof

Maastricht UMC+ DataHub

Utrecht University

Metadata

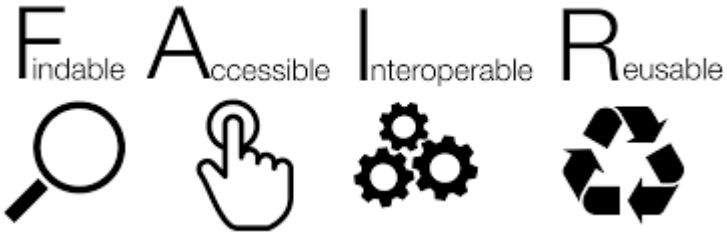
Maastricht UMC+ DataHub

Utrecht University

metaInx

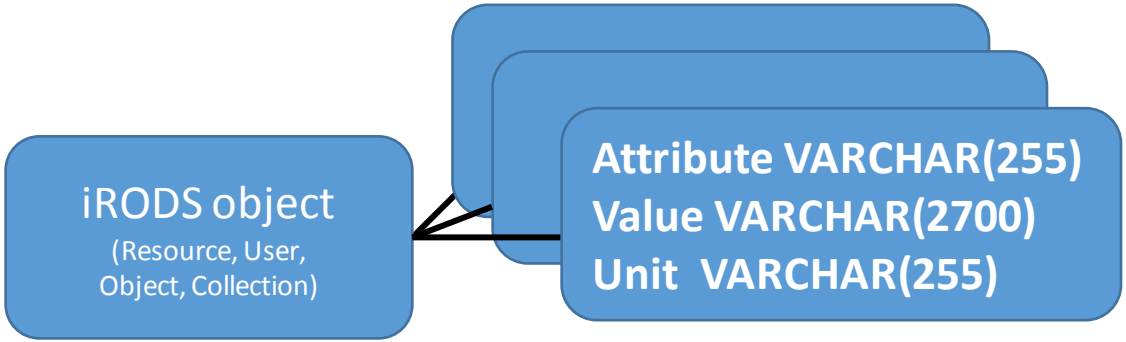
YODA

CYVERSE



Come up with a generic way within iRODS to provide metadata templates, validation and user interaction

Metadata in iRODS



AVU: Attribute Value Unit

Why JSON?

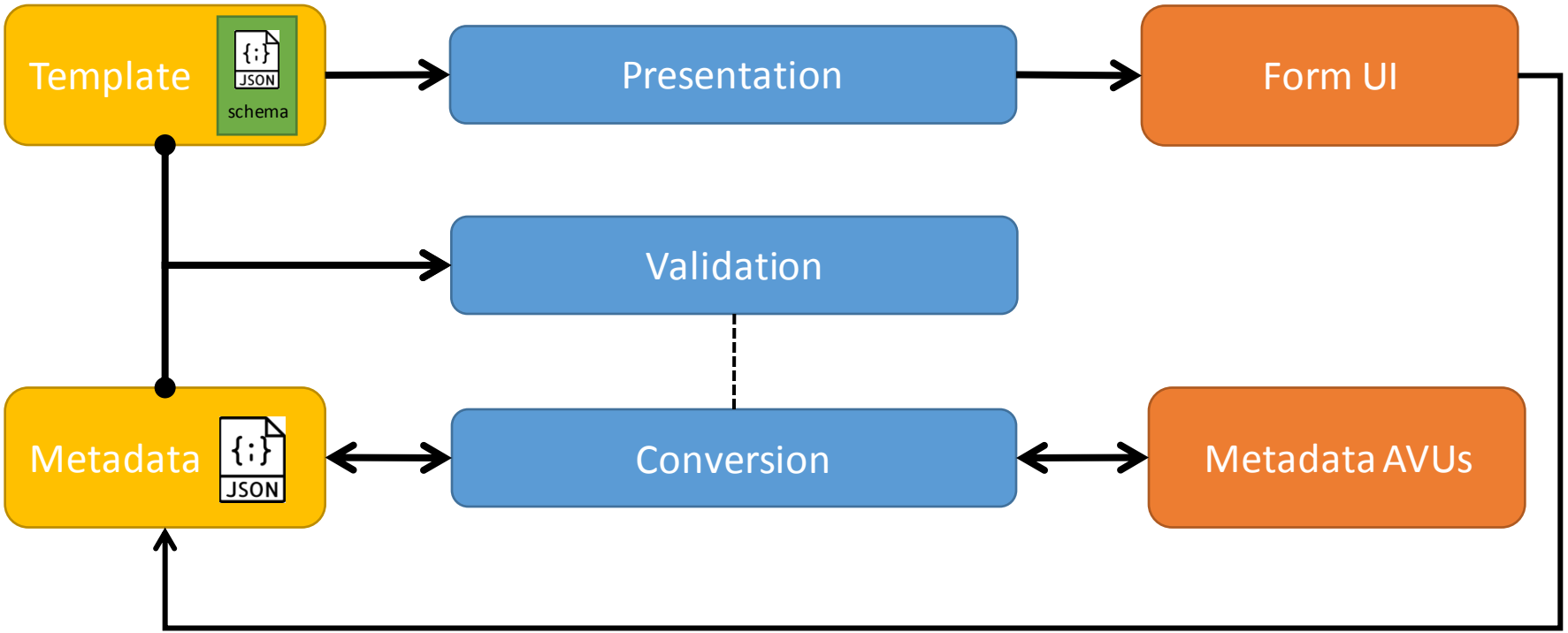
- Long dark flowing hair
- Knows C++
- Loves beer
- All around nice guy

Why JSON?

- JSON is flexible and easy
- Features: nesting and arrays
- Human and developer readable
- Validation: JSON-schema
- Linked Data: JSON-LD


```
{
    hey: "guy",
    anumber: 243,
  - anobject: {
        whoa: "nuts",
      - anarray: [
            1,
            2,
            "thr<h1>ee"
        ],
        more: "stuff"
    },
    awesome: true,
    bogus: false,
    meaning: null,
    japanese: "明日がある。",
    link: http://jsonview.com,
    notLink: "http://jsonview.com is great"
}
```

Overview - Layers



Design requirements

- Bijection between JSON <-> AVU
- Lean JSON -> AVU conversion.
- Keep Attribute->Value pairs the same
- Compatible with existing or additional AVUs
- Compatible/aware of JSON-LD

Conversion

```
{
  "title": "Hello World!",
  "parameters": {
    "size" : 42,
    "readOnly" : false
  },
  "authors" : ["Foo", "Bar"],
  "references": [
    {
      "title": "The Rule Engine",
      "doi": "1234.5678"
    }
  ]
}
```

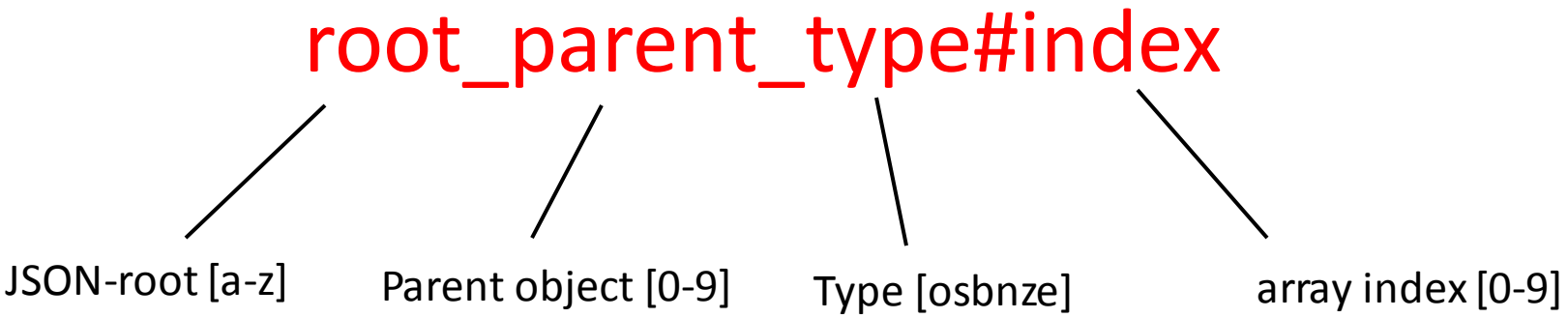
Attribute	Value	Unit
title	Hello World!	root_0_s
parameters	o1	root_0_o1
size	42	root_1_n
readOnly	False	root_1_b
authors	Foo	root_0_s#0
authors	Bar	root_0_s#1
references	o2	root_o_o2#0
title	The Rule Engine	root_2_s
doi	1234.5678	root_2_s

Conversion: step by step

```
{
  "title": "Hello World!",
  "parameters": {
    "size" : 42,
    "readOnly" : false
  },
  "authors" : ["Foo", "Bar"],
  "references": [
    {
      "title": "The Rule Engine",
      "doi": "1234.5678"
    }
  ]
}
```

Attribute	Value	Unit

Conversion - Usage of the unit field



Conversion - Implementation

Conversion
irods_avu_json

- Python
- pip package
- Standalone from iRODS

/MaastrichtUniversity/irods_avu_json

Ruleset
irods_avu_json-ruleset

- Python ruleset (core.py)
- Includes AVU microservices

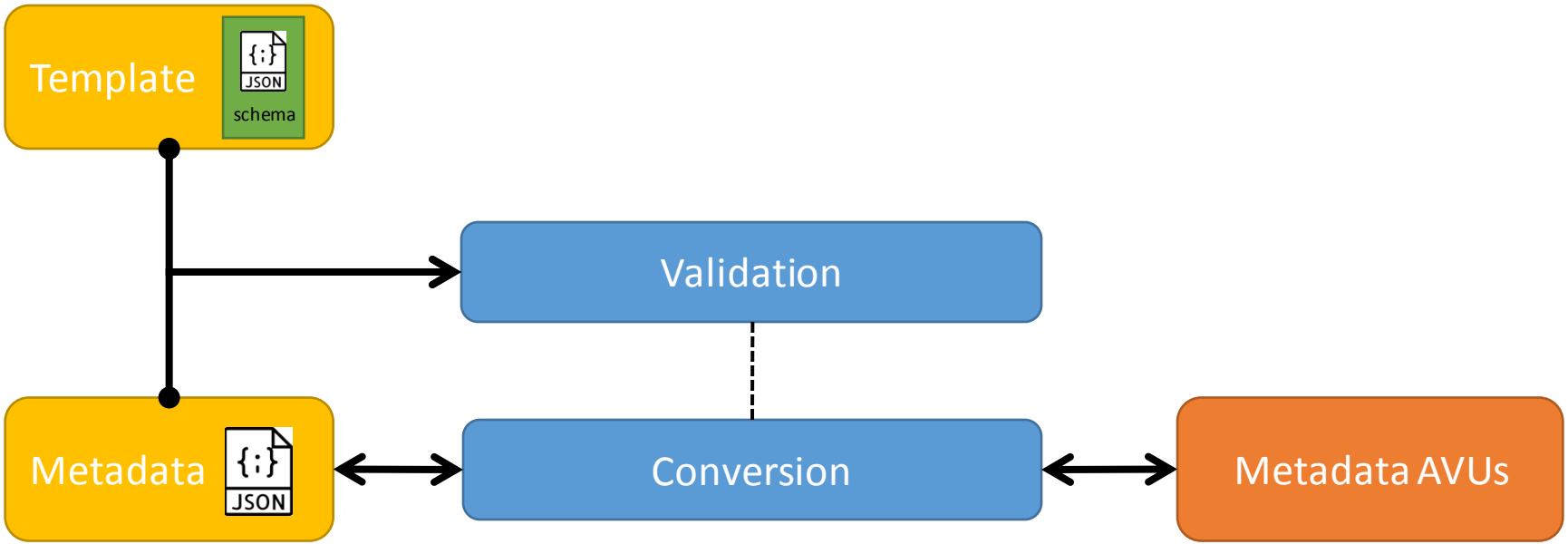
/MaastrichtUniversity/irods_avu_json-ruleset

setJsonToObj(*object, *objectType, *jsonRoot, *json)

getJsonFromObj(*object, *objectType, *jsonRoot)

Conversion - Demo

Validation - Overview

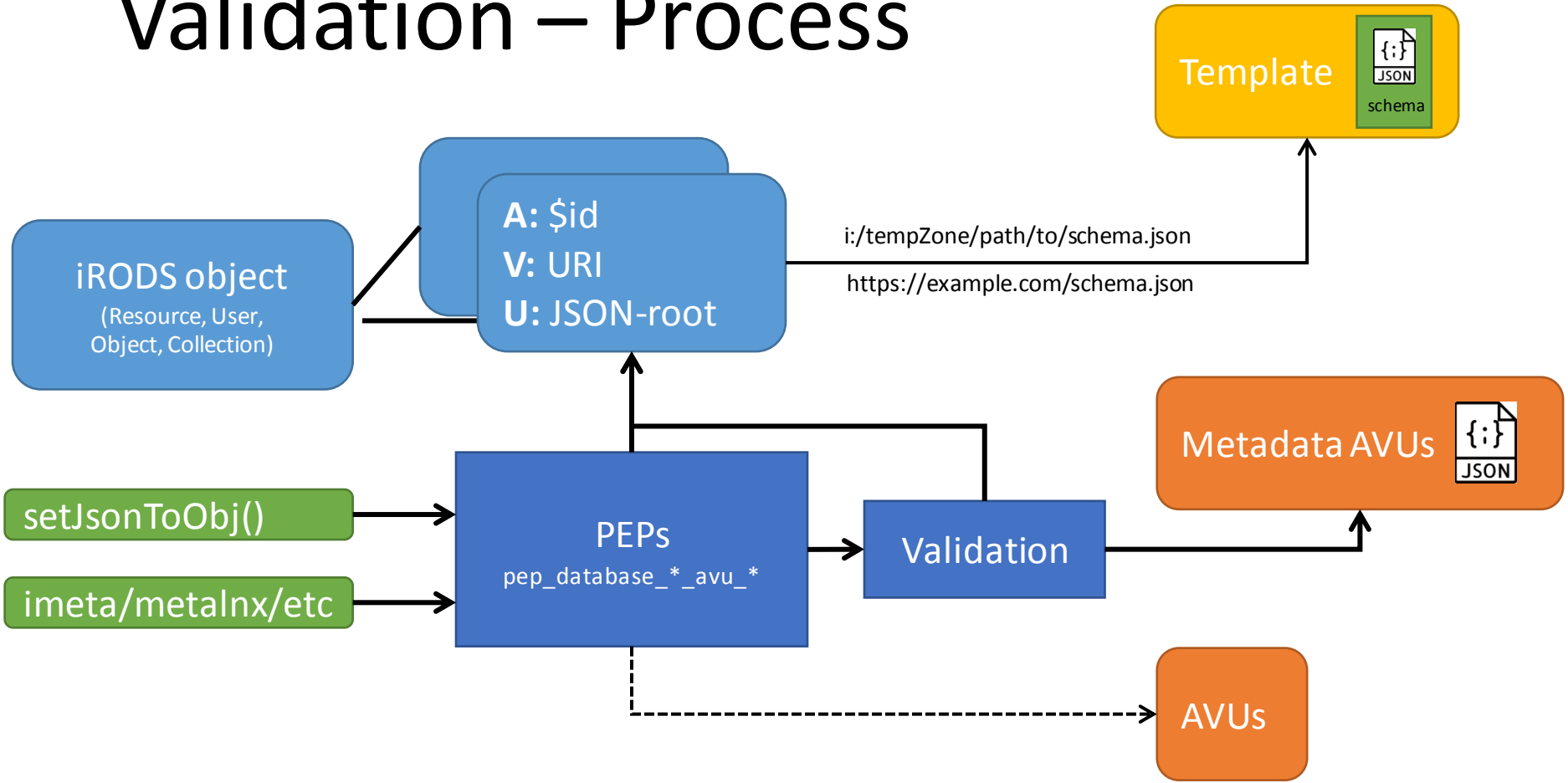


Validation – JSON-schema

JSON Schema is a vocabulary that allows you to **annotate** and **validate** JSON documents.

```
{
    "title": "Person",
    "type": "object",
    "properties": {
        "firstName": {
            "type": "string"
        },
        "lastName": {
            "type": "string"
        },
        "age": {
            "description": "Age in years",
            "type": "integer",
            "minimum": 0
        }
    },
    "required": ["firstName", "lastName"]
}
```

Validation – Process



Validation - Implementation

Ruleset
irods_avu_json-ruleset

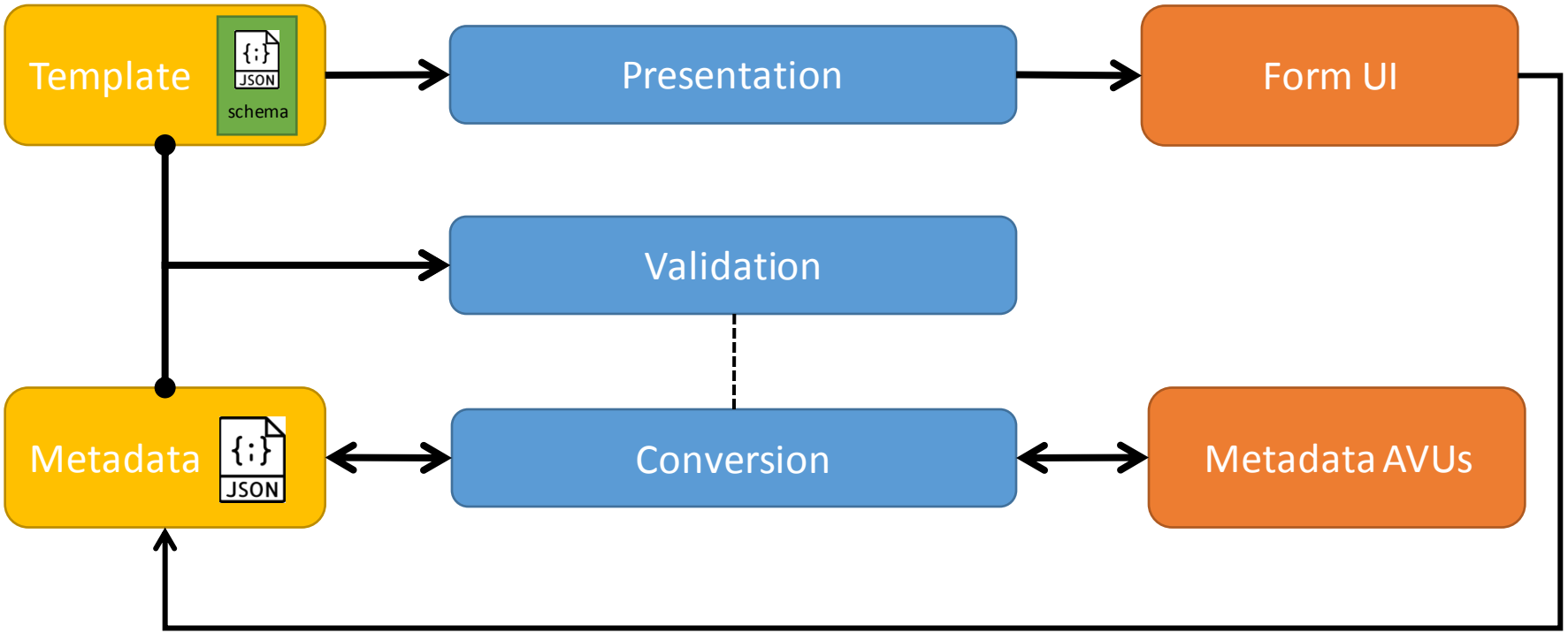
- Python ruleset (core.py)
- Includes AVU microservices
- PEPs

/MaastrichtUniversity/irods_avu_json-ruleset

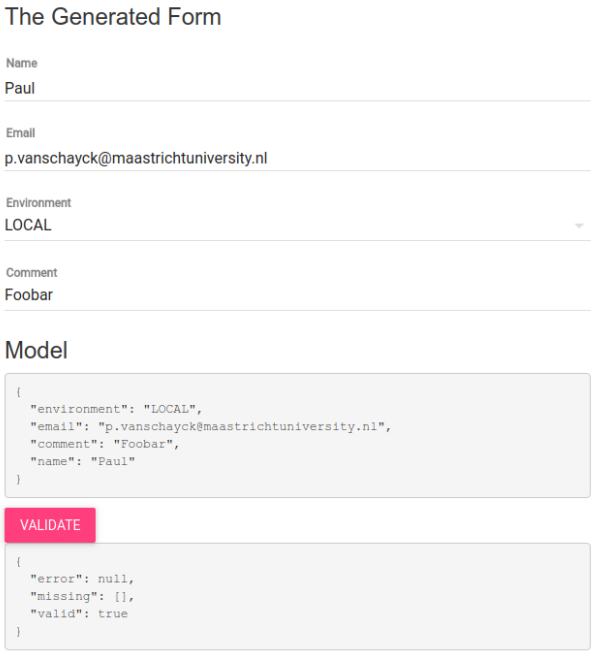
- setJsonSchemaToObj(*object, *objectType, *jsonSchema, *jsonRoot)
- getJsonSchemaFromObj(*object, *objectType, *jsonRoot)
- pep_database_*_avu_*(*)

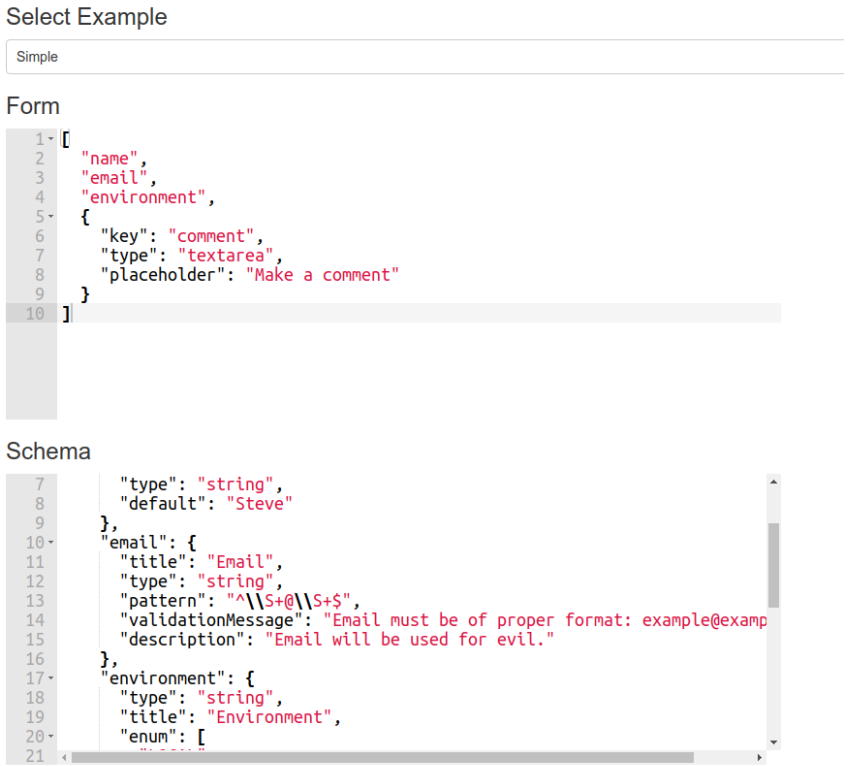
Validation - Demo

Overview - Presentation



Presentation – JSON-schema -> Form





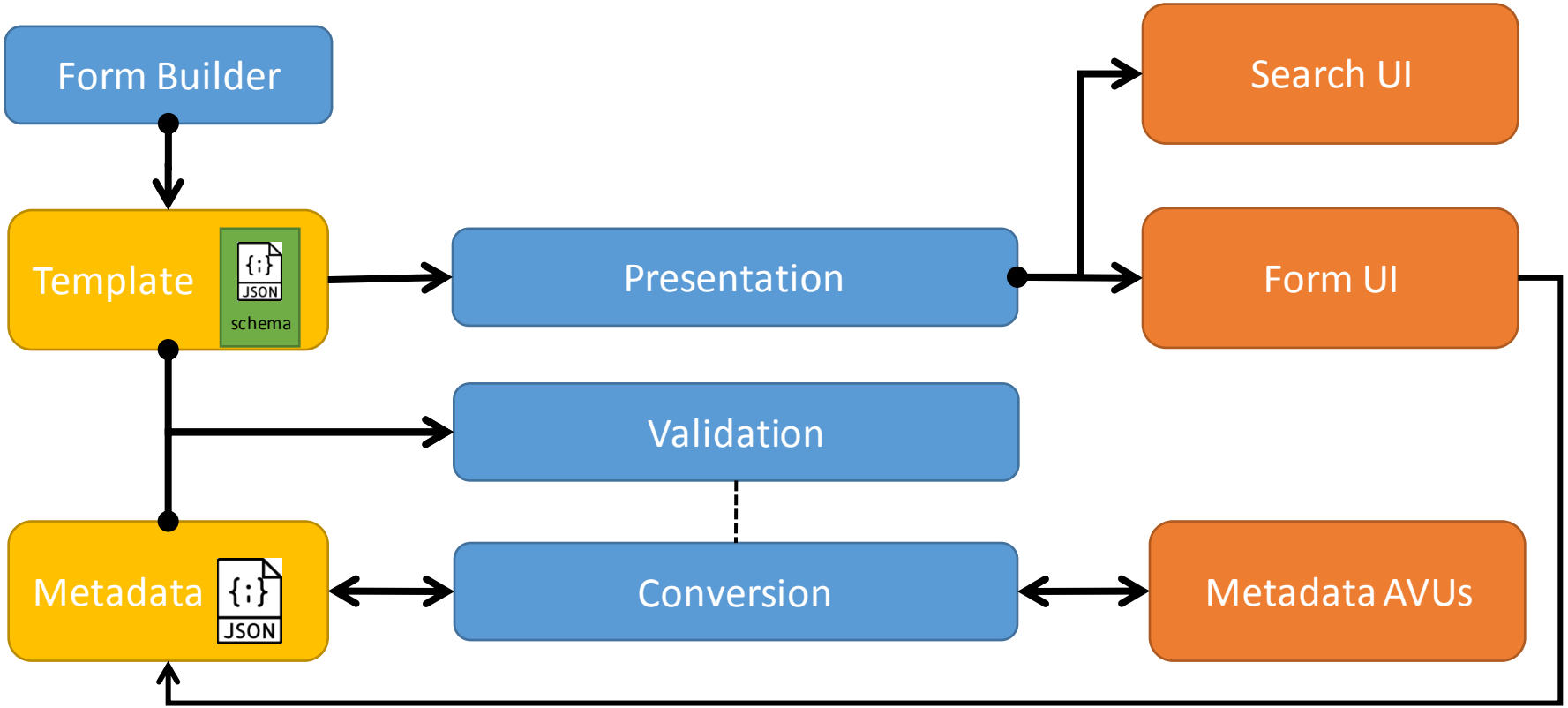
From https://github.com/networknt/react-schema-form

Our use cases

1. YoDa (Utrecht University)
   - Uses forms defined in JSON-schema (build in ReactJS), outputting JSON
2. DataHub (Maastricht University)
   - Looks at CEDAR, an Angular JSON-schema form generator, outputting JSON-LD

Presenation - Demo

Future


Form Builder
Template
JSON
schema
Presentation
Search UI
Form UI
Validation
Metadata
JSON
Conversion
Metadata AVUs

Final thoughts

- Devils advocate:
  - Why not store entire JSON in single AVU?
- JSON-schema:
  - No real standard for UI presentation (yet)
- Implementation:
  - Devs: Microservices for setting AVUs! (iRODS-4185)
  - Devs: PEPs for AVU control are difficult to use
  - Possible race conditions during set (locking?)

Acknowledgements

Maastricht UMC+ DataHub

Utrecht University

iRODS

Daniel Theunissen + rest of team

Lazlo Westerhof

Ton Smeele

Metadata Templates Working Group

Hackaton

This Friday!

Conversion
irods_avu_json

- Python
- pip package
- Standalone from iRODS

MaastrichtUniversity/irods_avu_json

Ruleset
irods_avu_json-ruleset

- Python ruleset (core.py)
- Includes AVU microservices
- PEPs

/MaastrichtUniversity/irods_avu_json-ruleset

Docker
irods_avu_json-docker

- Single iCAT instance
- Install microservices and ruleset
- Runs tests

/MaastrichtUniversity/irods_avu_json-docker

Ideas:

- Add AVU-unit microservices to core?
- C++ microservice implementation?
- Better PEPs?
- Your applications?

A quick introduction to JSON-LD

JSON input

```
{
 "id": "http://hdl.handle.net/21.12109/P000000009C000000008",
 "creator": "https://orcid.org/0000-0001-6591-4637",
 "description": "Lorem Ipsum",
 "title": "Foobar"
}
```

@context (also JSON)

```
{
  "id": "@id"
  "creator": "http://purl.org/dc/terms/creator"
  "description": "http://purl.org/dc/terms/description",
  "title": "http://purl.org/dc/terms/title",
}
```

Result: Linked data (RDF)

```
<http://hdl.handle.net/21.12109/P000000009C000000008> <http://purl.org/dc/terms/creator> "https://orcid.org/0000-0001-6591-4637" .
<http://hdl.handle.net/21.12109/P000000009C000000008> <http://purl.org/dc/terms/description> "Lorem Ipsum" .
<http://hdl.handle.net/21.12109/P000000009C000000008> <http://purl.org/dc/terms/title> "Foobar" .
```

Convert any JSON into linked data by providing the @context

Result: Human and developer readable linked data!