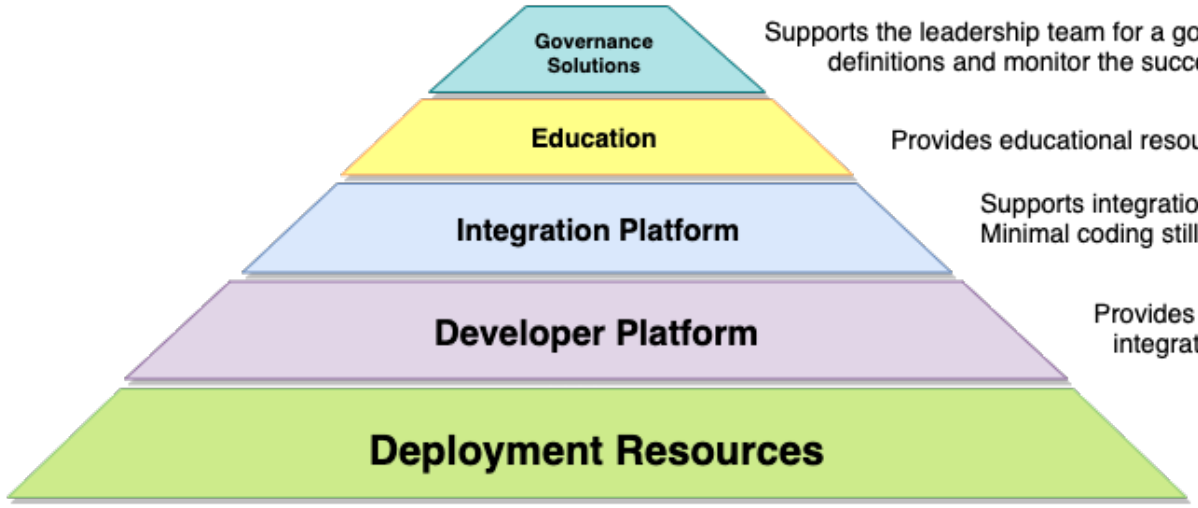
Egeria React UI and the architecture behind it

David Radley



Supports the leadership team for a governance program providing the ability to create common definitions and monitor the success of the governance efforts across the enterprise.

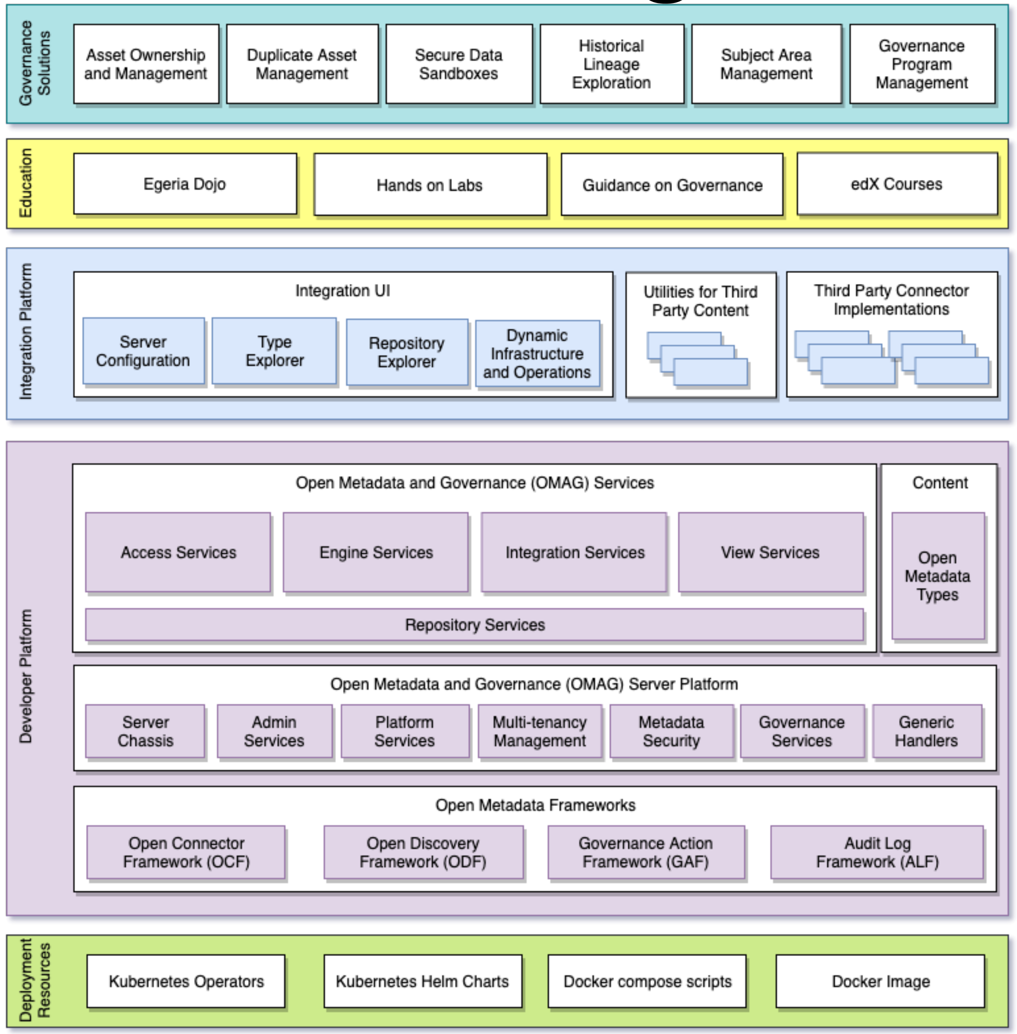
Provides educational resources for different personas and starting points.

Supports integration of popular technologies by installing and configuring Egeria. Minimal coding still required around unusual and home-grown tools and technologies.

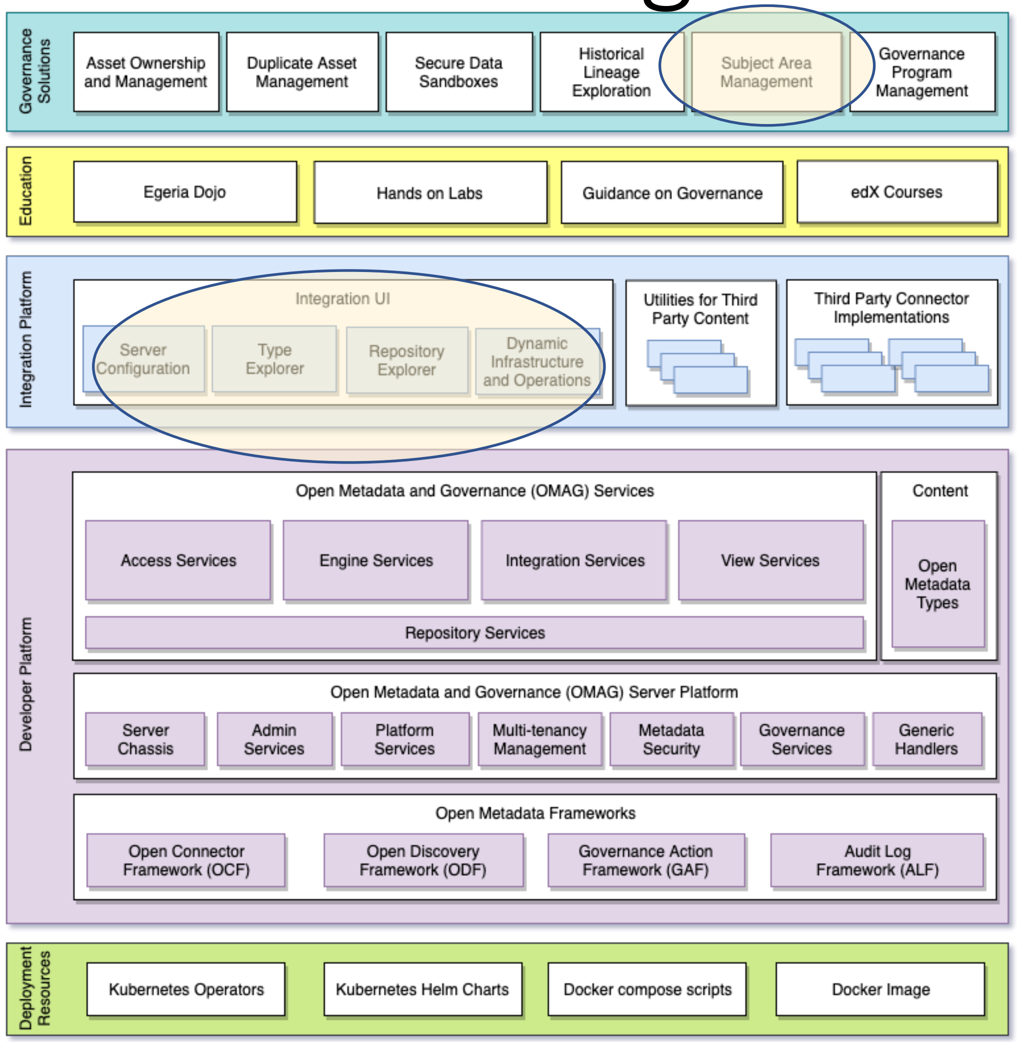
Provides frameworks, APIs, hosting platforms for building an integrated metadata and governance distributed solution.

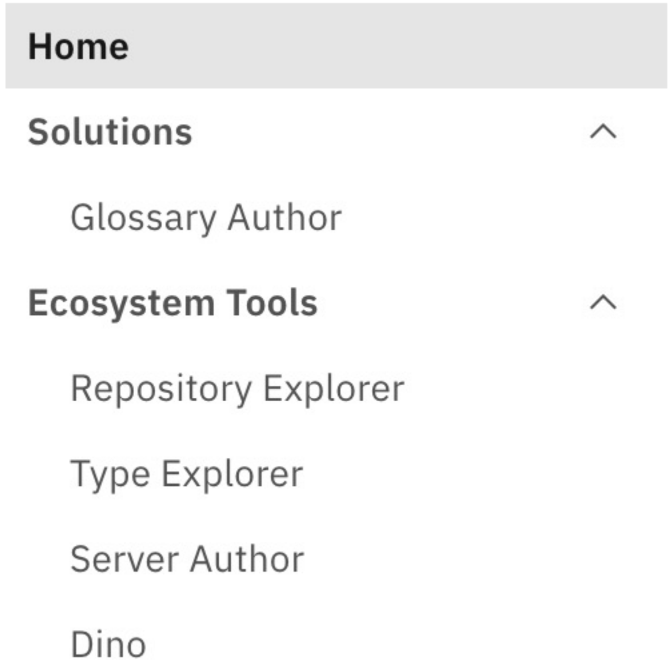
Tools and components to ease the process of deploying Egeria into production.

Functional organisation

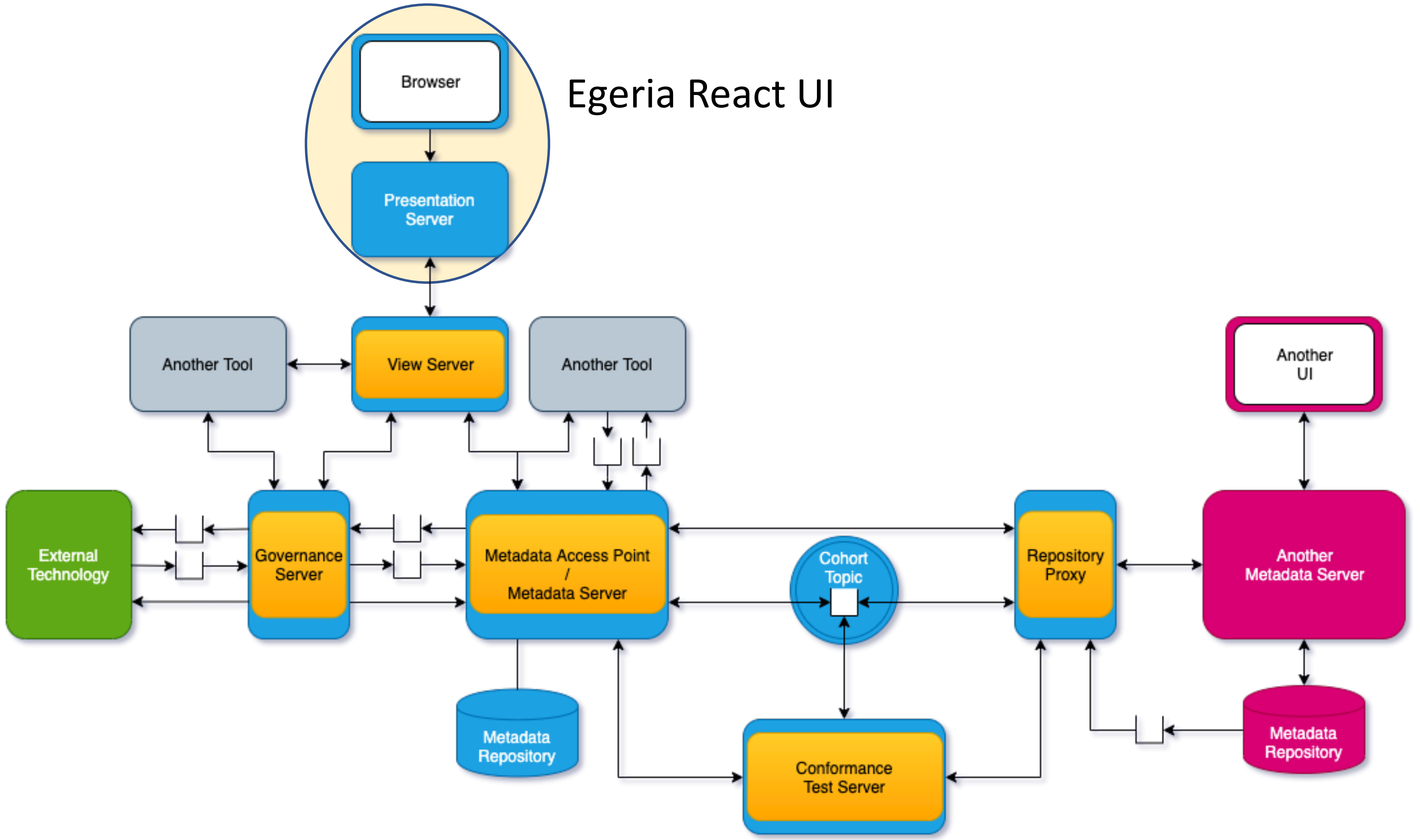


Functional organisation





The high level architecture



https://egeria.odpi.org/open-metadata-publication/website/planning-guide/

Github

https://github.com/odpi/egeria-react-ui

Please star us if you haven't

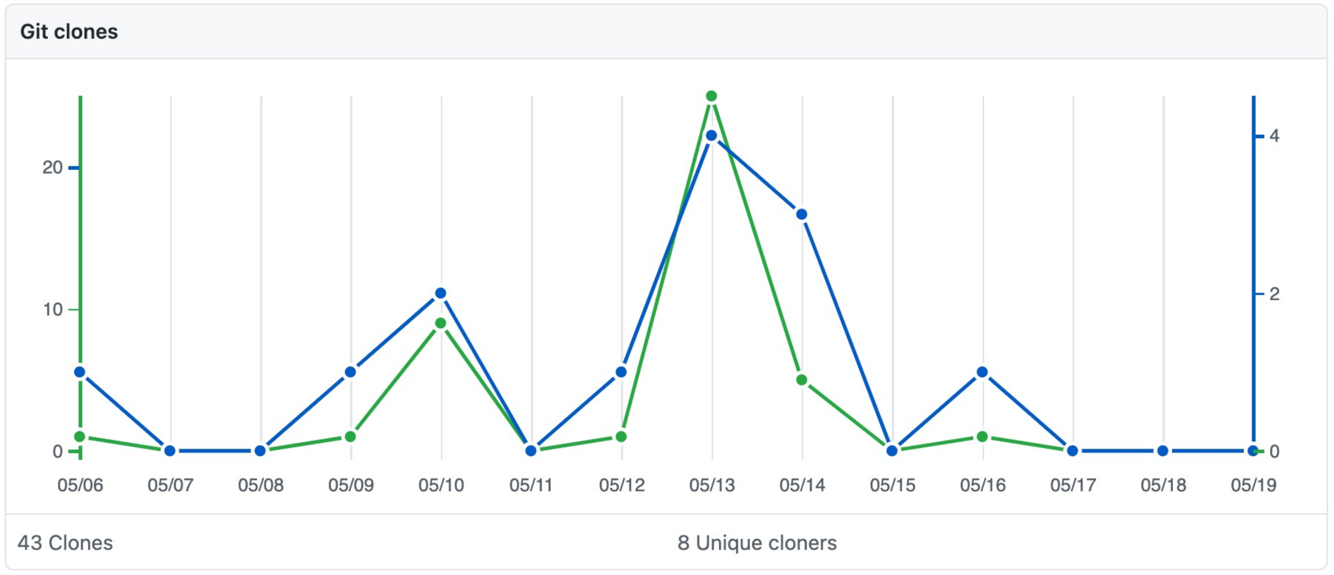
FE Developers with these skills, can become a valued contributor quickly:

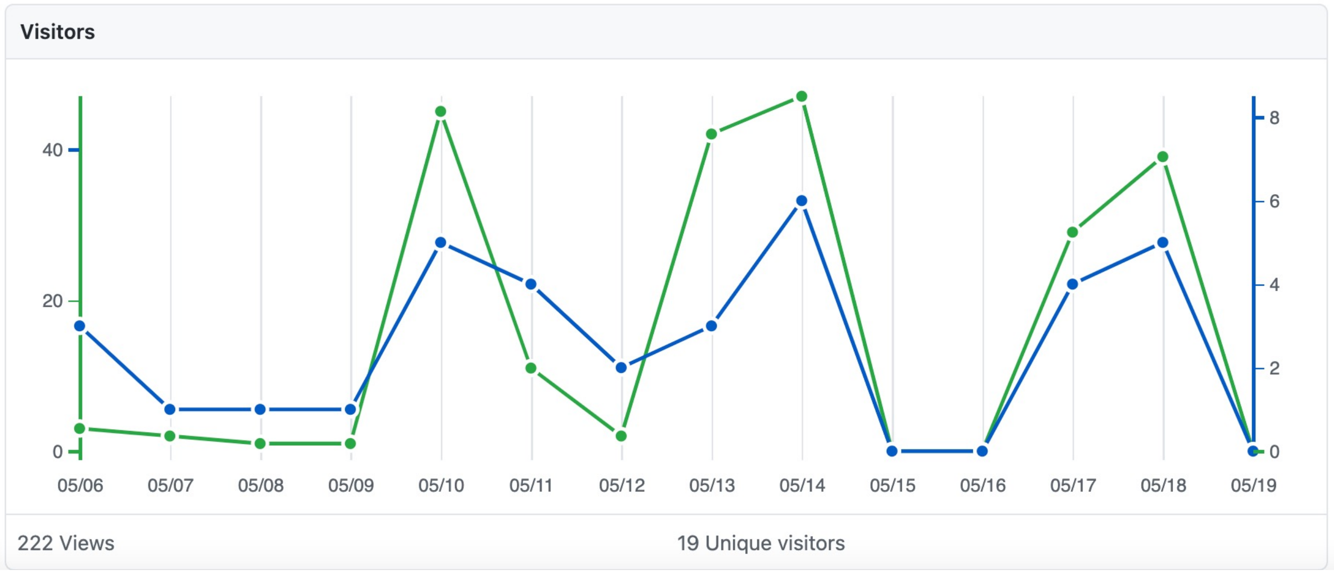
Javascript

Node

React

Web pack





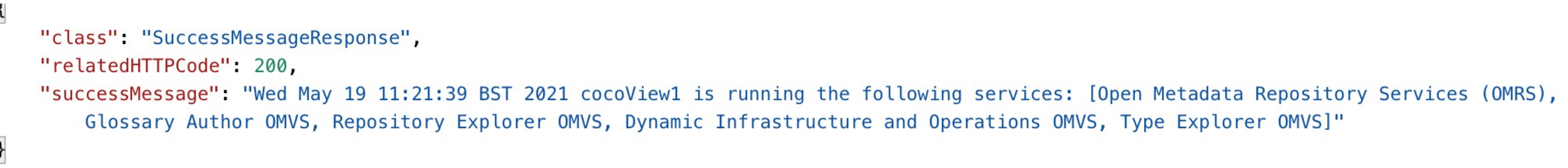
Checklist - before you run the UI

- Have you got Egeria platform running , in one of the following ways:
  - Locally
  - Kubenetes
  - Docker Compose
- Work out which servers you need. I minimal scenario would be a metadata server and a view server
- Work out which UI capability you want to use and configure the appropriate View services and Access Services.

UI and view services

UI Capability	View Service	View service target
Tex	Tex	OMRS
Rex	Rex	OMRS
Dino	Dino	OMRS
Glossary Author	Glossary author	Subject Area OMAS
Server Author	Server Author (not their yet)	Admin server and platform services

Expect the following OMVS's to be configured and started for a fully functioning UI.



```
{
  "class": "SuccessMessageResponse",
  "relatedHTTPCode": 200,
  "successMessage": "Wed May 19 11:21:39 BST 2021 cocoView1 is running the following services: [Open Metadata Repository Services (OMRS),
      Glossary Author OMVS, Repository Explorer OMVS, Dynamic Infrastructure and Operations OMVS, Type Explorer OMVS]"
}
```

Configuration and starting the presentation server

Assuming you have a running Egeria including view services. Clone the Git repo
In cra-server folder. Create a .env file containing (you can use environment variables – remember to escape /):

```
EGERIA_PRESENTATIONSERVER_SERVER_coco={"remoteServerName":"cocoView1","remoteURL":"https://localhost:9443"}
```

tenant

Tenant endpoint

Production

- In cra-client folder

```
npm install
```

```
npm run build
```

- In cra-server folder

```
npm install
```

```
npm run prod
```

Development

- In cra-client folder

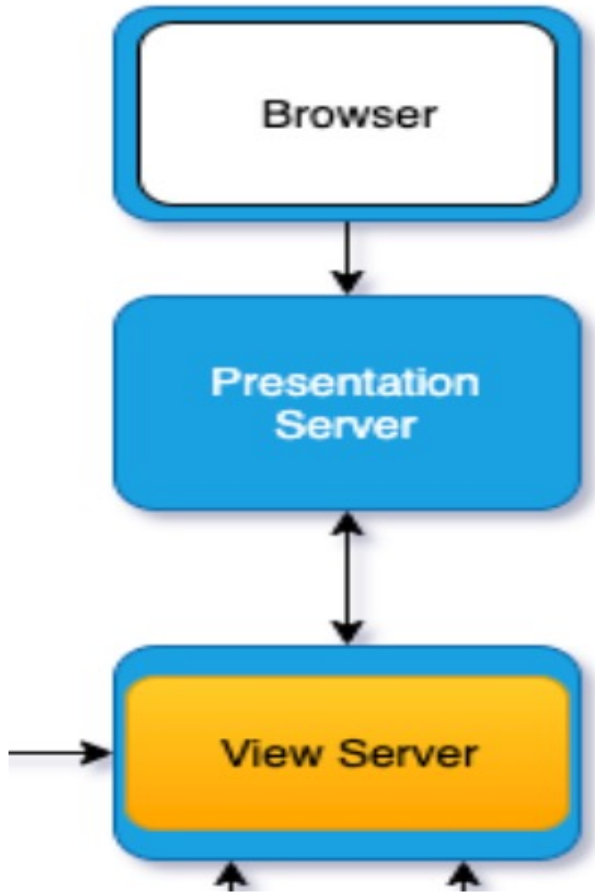
```
npm install
```

- In cra-server folder

```
npm install
```

```
npm start
```

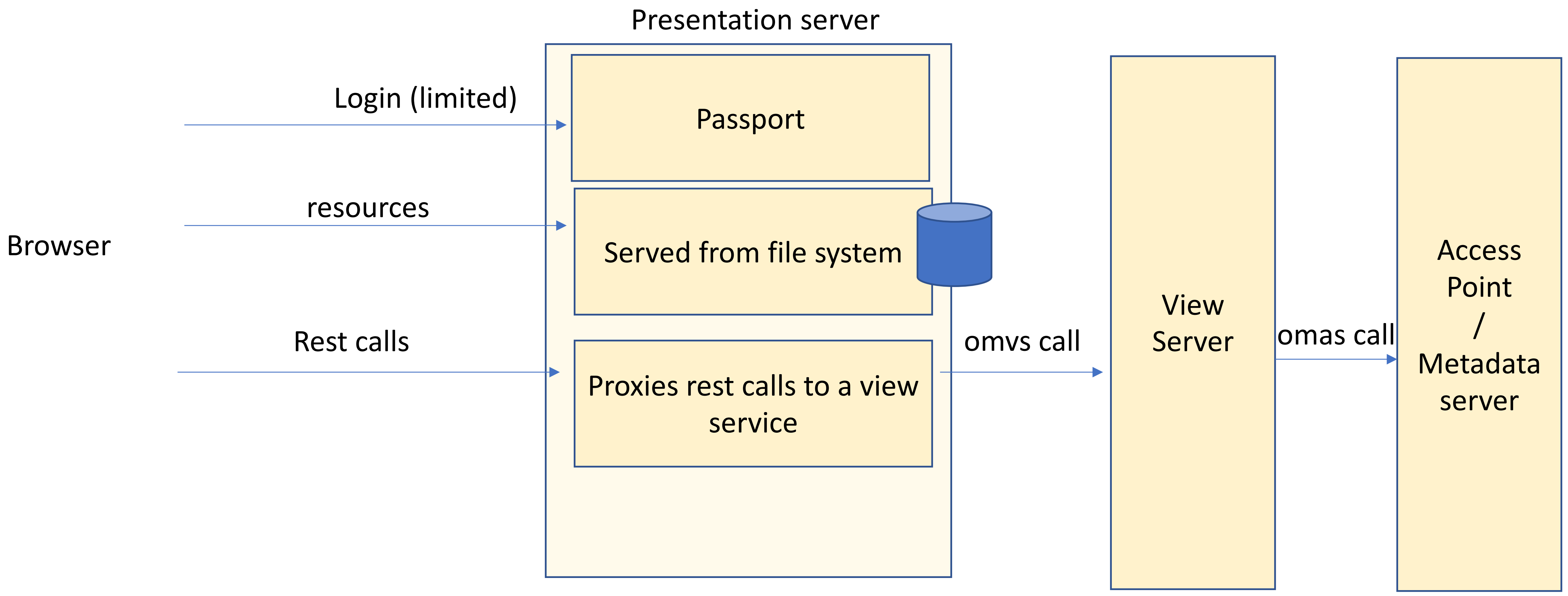
The UI part



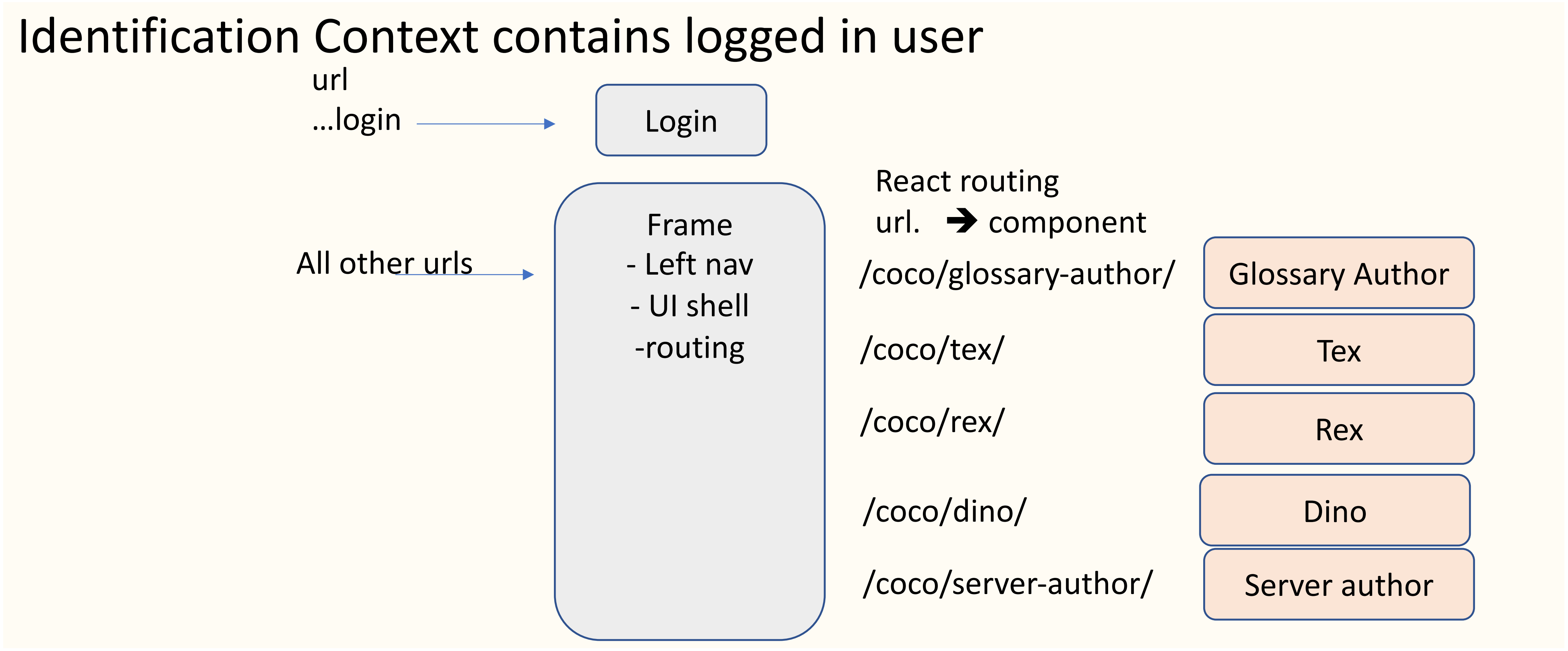
Express based server, uses passport for authentication, webpack for build and client side functional React components with hooks, uses Carbon components. It is tenant aware; tenant information is in environment variables to indicate the View Service endpoint details.

View server is a tenanted view of metadata for UI consumption. It is a type of OMAG Server (so is configured in the usual way)

React UI architecture – presentation server



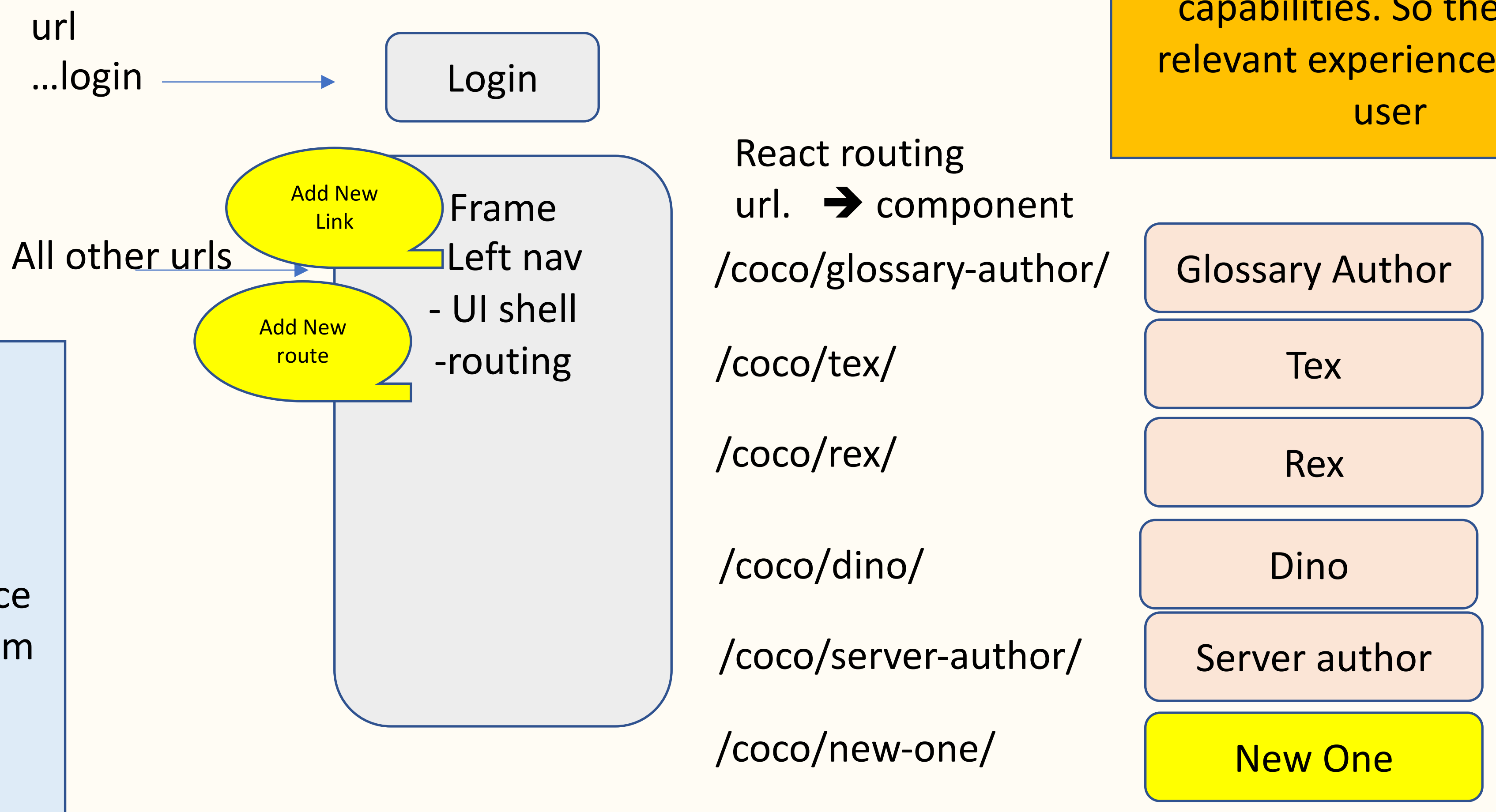
React UI architecture – React part



Adding a new component

Future
In time we would like the logged in user to have an associated profile, that would determine the user interfaces home screen, preferences and capabilities. So there is a relevant experience for the user

Identification Context contains user



React routing

url. ➔	component
/coco/glossary-author/	Glossary Author
/coco/tex/	Tex
/coco/rex/	Rex
/coco/dino/	Dino
/coco/server-author/	Server author
/coco/new-one/	New One

A new component
- Particular persona
- Pointing to a view service
- Can pick up the user from the context.

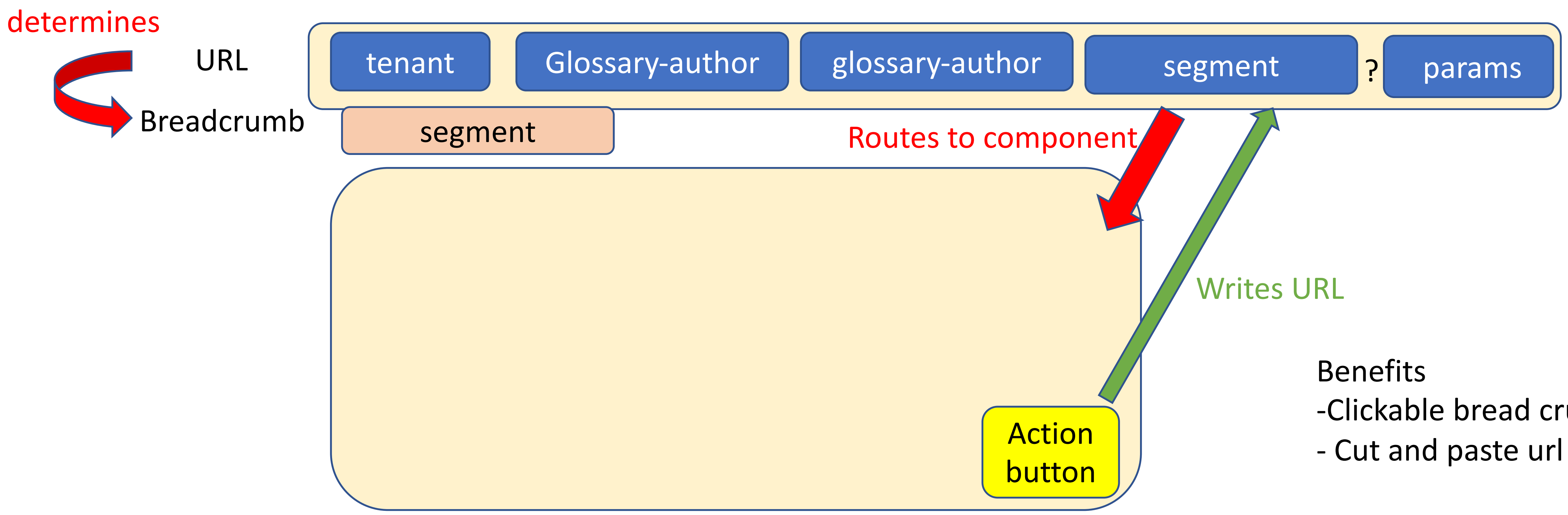
Tex Rex Dino nested contexts

```
<InteractionContextProvider>
   <RepositoryServerContextProvider>
     <TypesContextProvider>
       <InstancesContextProvider>
             <GraphControls />
             <DetailsPanel />
             <DiagramManager />
...
```

Nested contexts works well when there is one page with complex interactions between the components

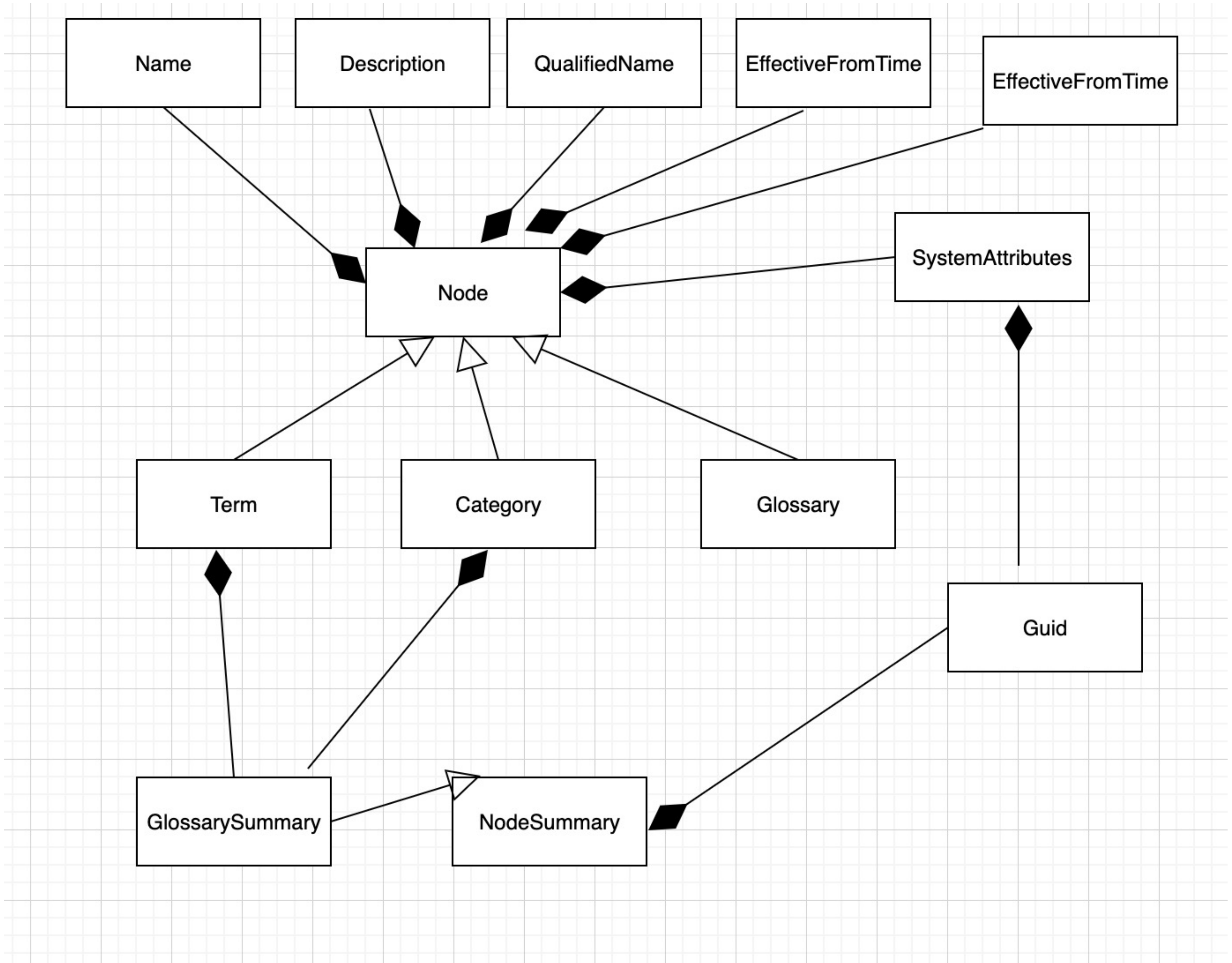
Wraps D3 graph

Glossary author url driven

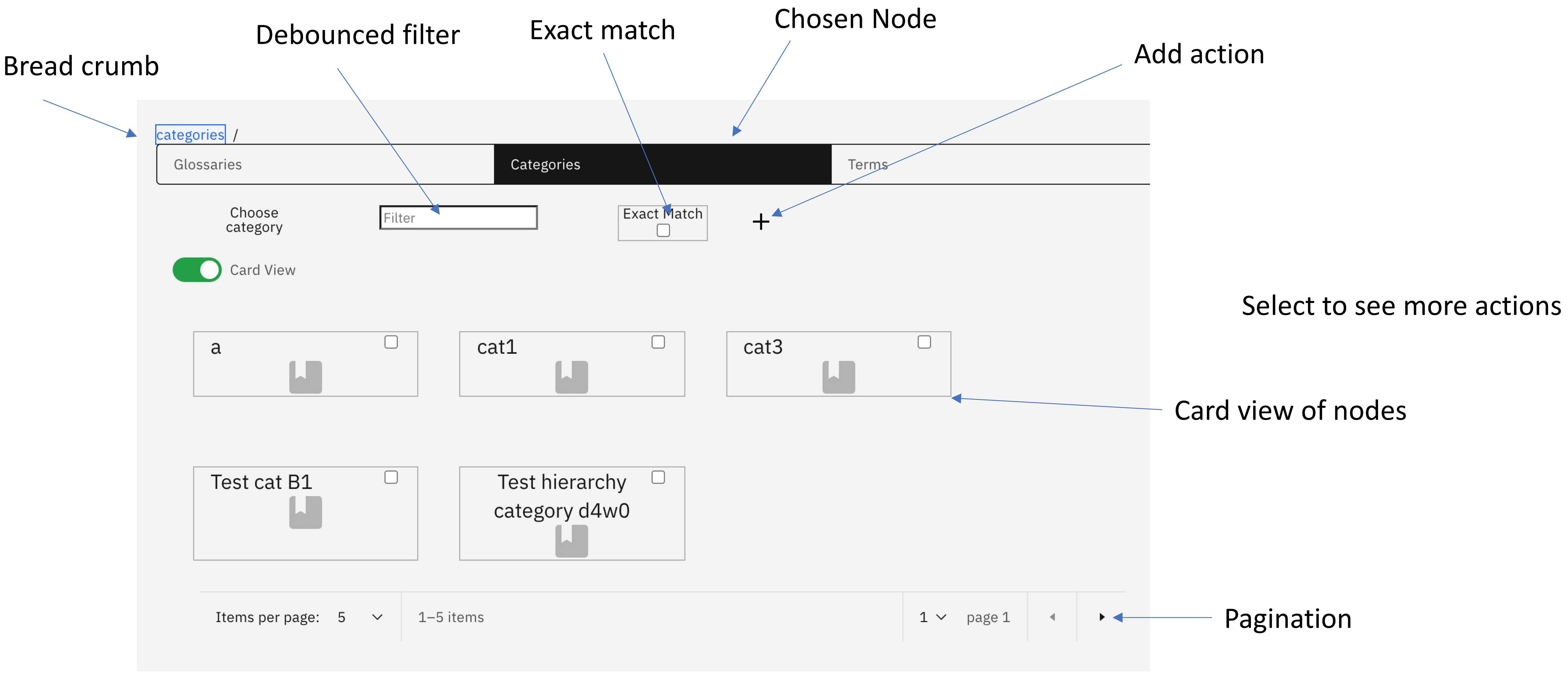


Benefits
-Clickable bread crumb
- Cut and paste url works

Node

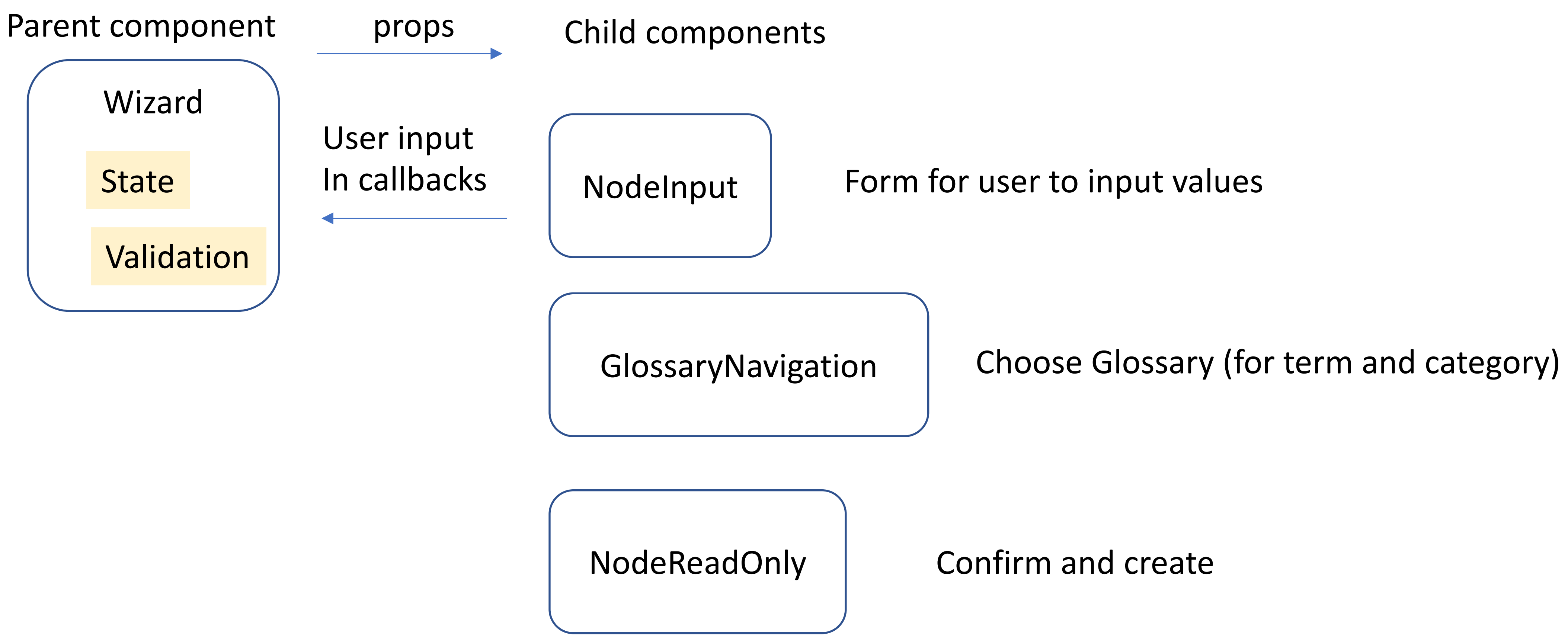

Name
Description
QualifiedName
EffectiveFromTime
EffectiveFromTime
SystemAttributes
Node
Term
Category
Glossary
Guid
GlossarySummary
NodeSummary

Glossary author search screen anatomy



Create Wizard

Controlled Component



Intention is to move update and delete
To this model

Selected node actions

- Quick terms – quickly author Term names for innovation session
- Update – update the selected Node
- Delete – delete the deleted node
- Glove – visualize the selected node
- See the children of the selected node

Glossary author Glove

- video

Server Author

- Context driven wizard to author servers.
- Interesting to see how this positions with the operator. Likely this will be used only in development to create configurations.
- Needs to move to use the view service
- Needs to pick up platform values from metadata

Where next?

- Core
  - Bring in line with the latest core Egeria security artifacts, which ill be a pattern on how to customize security for an organization
  - May require server author view to be enabled.
- Glossary author
  - Finish the glossary author create and update wizards
  - Create and update on relationships.
  - Standardise icons, using the new repo.
- Server author
  - How to position with operator
  - View service
  - Picking up platform values from metadata server e.g. security connector class name
- Community profile and the governance program
  - Glossary author to pick up custom confidentiality etc classifications as defined by the governance program
  - Enable the community profile to get a more personized UI.
  - Reference data & Valid values
- Consider UI capabilities as required to grow the community: asset search, visualization, semantic assignment, collaboration.
- One experience across Polymer and React?