Model Catalog Technologies

The MINT Model Services describe physical, environmental and social models

Ontologies

We have developed two main ontologies to help structure the metadata and contents of the model catalog:

Software Description Ontology (SD)

Ontology used to capture the overall metadata of scientific software, including its versions, functionality, inputs, outputs, etc. [Documentation]

Software Description Ontology for Models (SDM)

Extension of the Software Description Ontology to capture metadata specific to models (e.g., the region where they are valid, their spatial grid, their temporal restrictions, etc.). [Documentation]

Model Catalog API

Model Catalog API for adding/modifying/deleting model metadata. We provide several clients (available here) to improve the experience for developers when accessing the contents of our APIs.

[RESTAPI] [Documentation] [Code] [Releases]

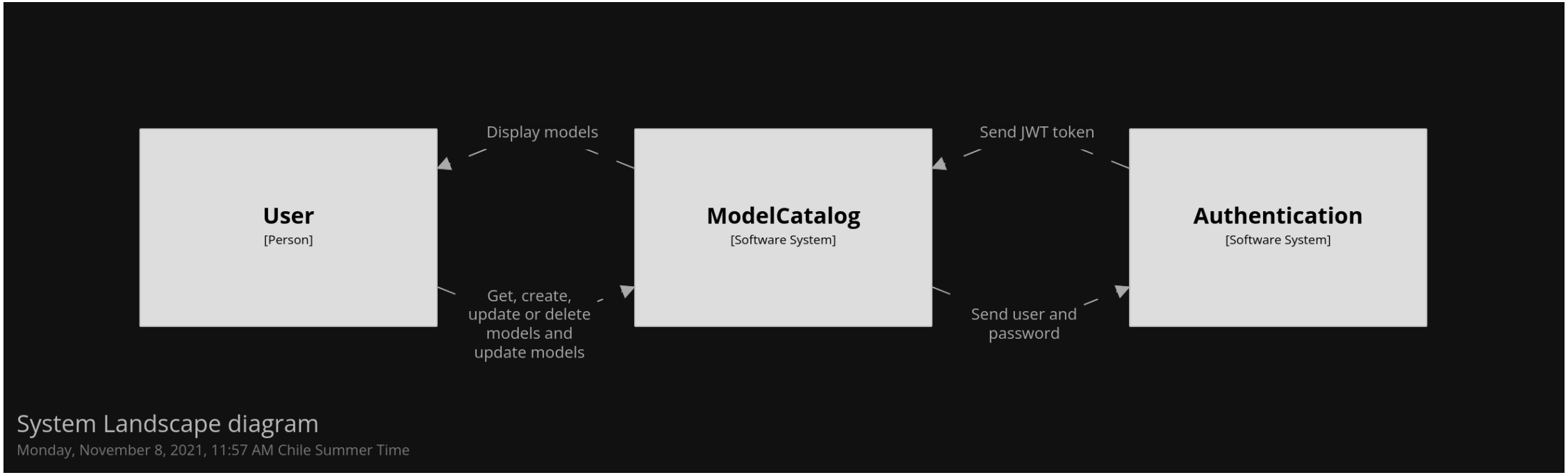
Model Catalog Explorer

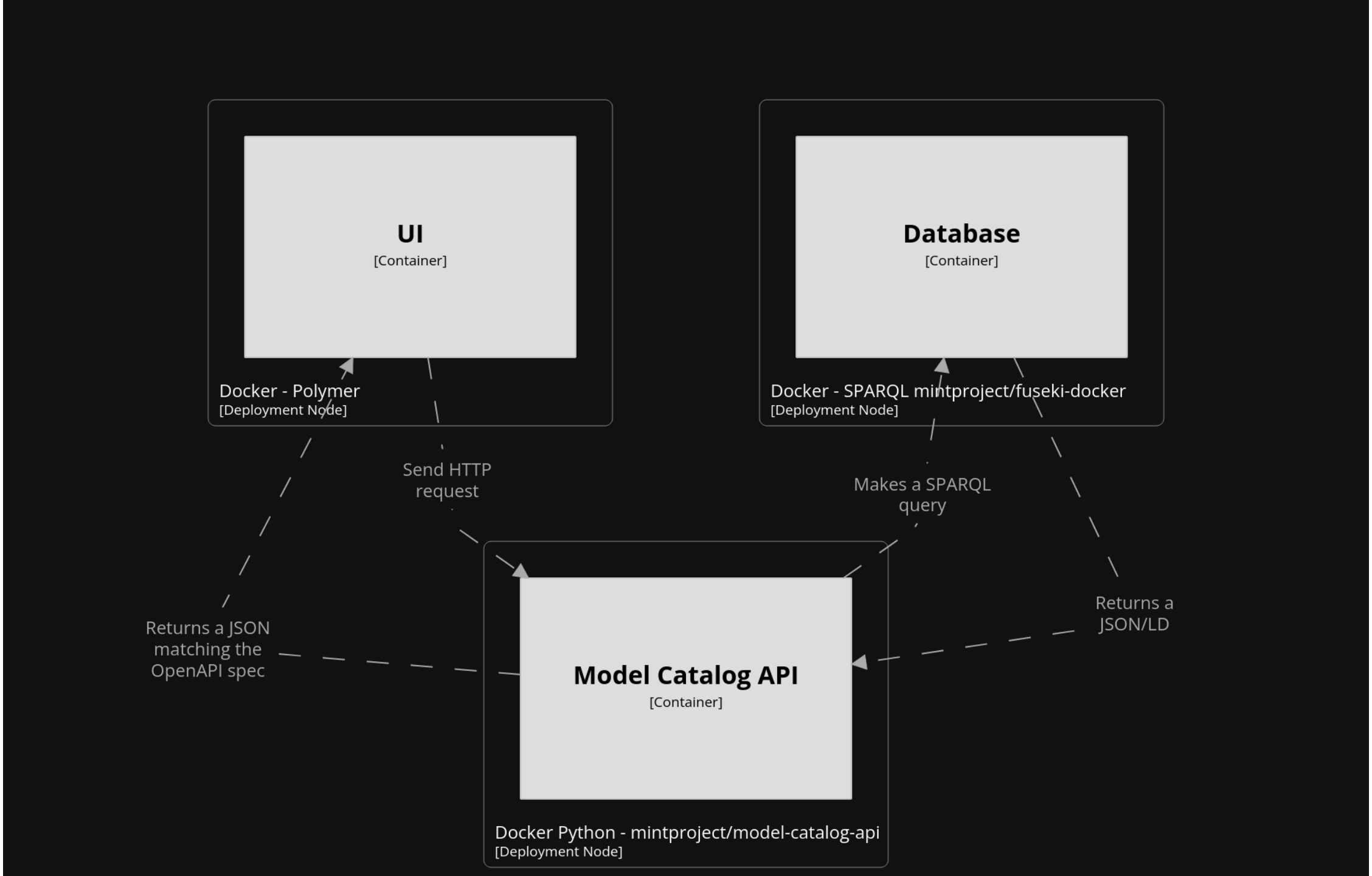
GUI for browsing the contents of the model catalog

[Website] [Demo]

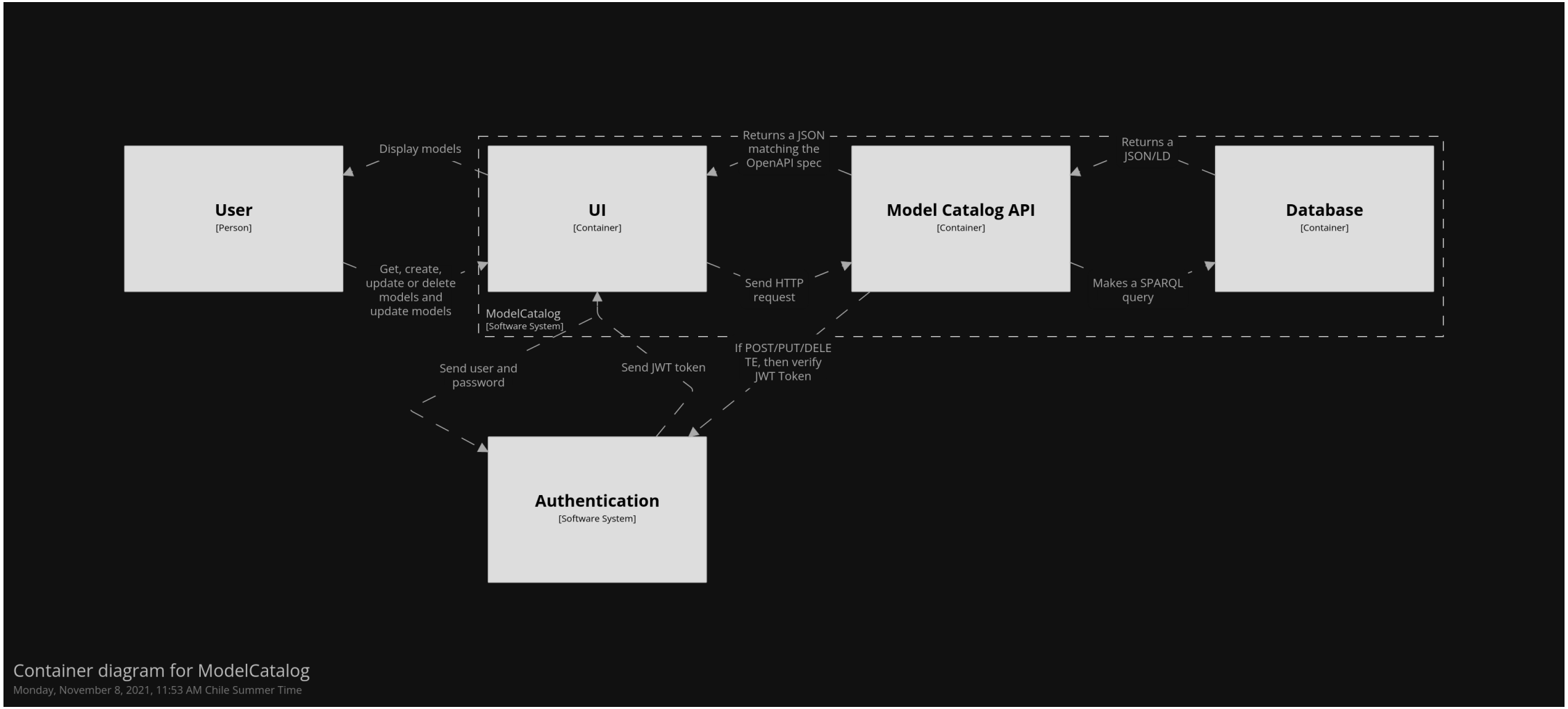
Registered users can add, edit, and remove metadata of their models, as well as creating new model setups, for example, adapting an existing model to a new region.

System


User
[Person]
Display models
Get, create, update or delete models and update models
ModelCatalog
[Software System]
Send JWT token
Send user and password
Authentication
[Software System]
System Landscape diagram
Monday, November 8, 2021, 11:57 AM Chile Summer Time


UI
[Container]
Database
[Container]
Docker - Polymer
[Deployment Node]
Docker - SPARQL mintproject/fuseki-docker
[Deployment Node]
Send HTTP request
Makes a SPARQL query
Returns a JSON matching the OpenAPI spec
Returns a JSON/LD
Model Catalog API
[Container]
Docker Python - mintproject/model-catalog-api
[Deployment Node]

More details



Repository

- API
- UI
- Database container

Issues

- Using urllib2 led to the code running sequentially and synchronously. Reported

Future work

Improve perfomance

- Move API from Python to Nodejs. The tools are better and they run async.

SPARQLjs

Future work

Use Linked Data Plataform (Trellis)

1. Resources are managed by means of a RESTful HTTP API,
2. An HTTP client is also able to retrieve the historical state Resource versioning and provenance(audit) feature
3. Authentication and Authorization can be easier Docs