THE OPEN GROUP

OSDU™

FORUM

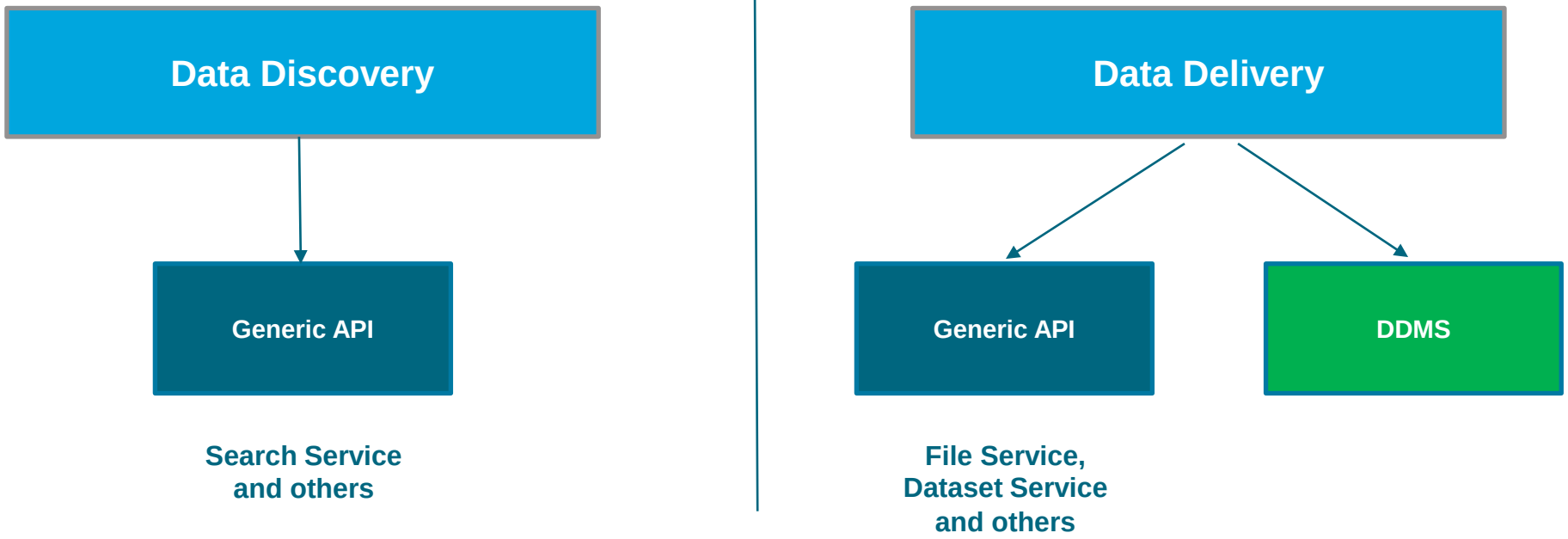
OSDU Data Platform Application Developer Training

Data Life Cycle Lab

epam

Data Life Cycle

Data Consumption



End-to-end Application

- These training materials are based on the Quickstart application
- You can explore the source code and run the application

- Backend in Python
- Frontend in React

Authorization: Configuration

- Register your application with the Identity Provider.
- Get **client_id**, **client_secret** and **auth_discovery_url** from your OSDU administrator or cloud provider.

```
OSDU_AUTH_CLIENT_ID="<your_client_id>"
OSDU_AUTH_CLIENT_SECRET="<well_hidden_secret>"
OSDU_AUTH_DISCOVERY_URL="<cloud_specific_auth_discovery_url_2.0>"
OSDU_AUTH_REDIRECT_URL="http://localhost:8080/auth/callback"
OSDU_AUTH_SCOPES="openid profile offline_access <your_client_id>/.default"
OSDU_DATA_PARTITION="opendes"
```

- Make sure **redirect_url** is whitelisted in cloud settings.

Authorization: Client initialization

```
from oic.oic import Client
from oic.utils.authn.client import CLIENT_AUTHN_METHOD

oidc_client = Client(client_authn_method=CLIENT_AUTHN_METHOD, verify_ssl=False)
oidc_client.provider_config(issuer=OSDU_AUTH_DISCOVERY_URL)

registration_info = RegistrationResponse(
    client_id=OSDU_AUTH_CLIENT_ID,
    client_secret=OSDU_AUTH_CLIENT_SECRET,
)
oidc_client.store_registration_info(registration_info)
```

Authorization: Redirect to authorization endpoint

```
# scopes depend on the cloud provider
scopes = []
for item in OSDU_AUTH_SCOPES.split(","):
    scopes.append(item)

# set parameters for authorization server request
args = {
    "response_type": "code",
    "client_id": oidc_client.client_id,
    "scope": scopes,
    "redirect_uri": OSDU_AUTH_REDIRECT_URL,
    "state": session.state,
}

# create authorization request
auth_request = oidc_client.construct_AuthorizationRequest(request_args=args)

# get login url from provider configuration
login_url = auth_request.request(oidc_client.authorization_endpoint)
raise HTTPFound(login_url)  # HTTP 302
```

Authorization: Get tokens

```
callback_response = oidc_client.parse_response(
    AuthorizationResponse,
    info=request.query_string,
    sformat="urlencoded")

# extract authorization code from auth server response
args = {
    "code": auth_response["code"],
}

# exchange authorization token for the access token
access_token_response = oidc_client.do_access_token_request(
    state=auth_response["state"],
    request_args=args
)

# extract tokens
access_token = access_token_response["access_token"]
refresh_token = access_token_response["refresh_token"]
```

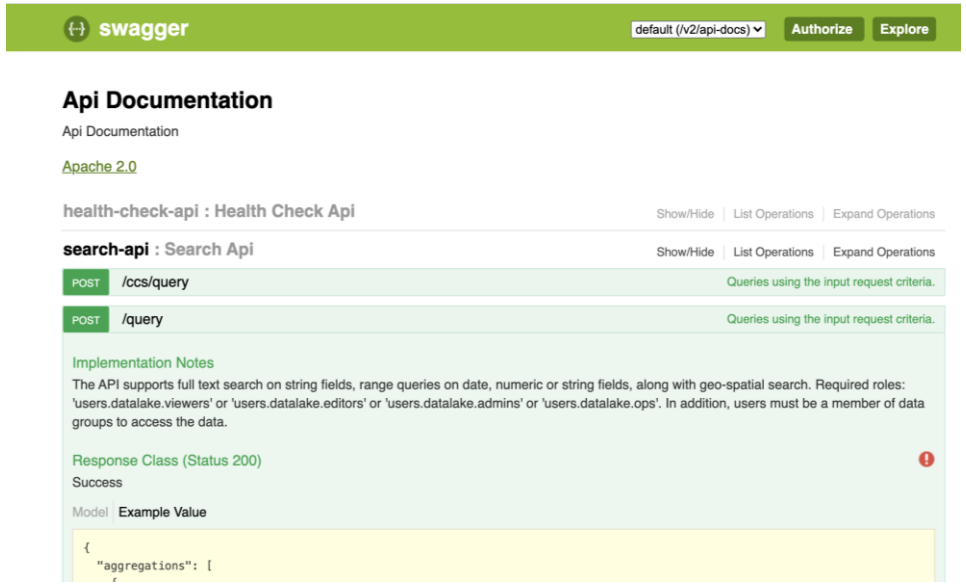
Authorization: Assemble headers

```
# Headers to be attached to every API call
headers = {
    "data-partition-id": OSDU_DATA_PARTITION,
    "Authorization": f"Bearer {self.access_token}",
}
```

Explore service API

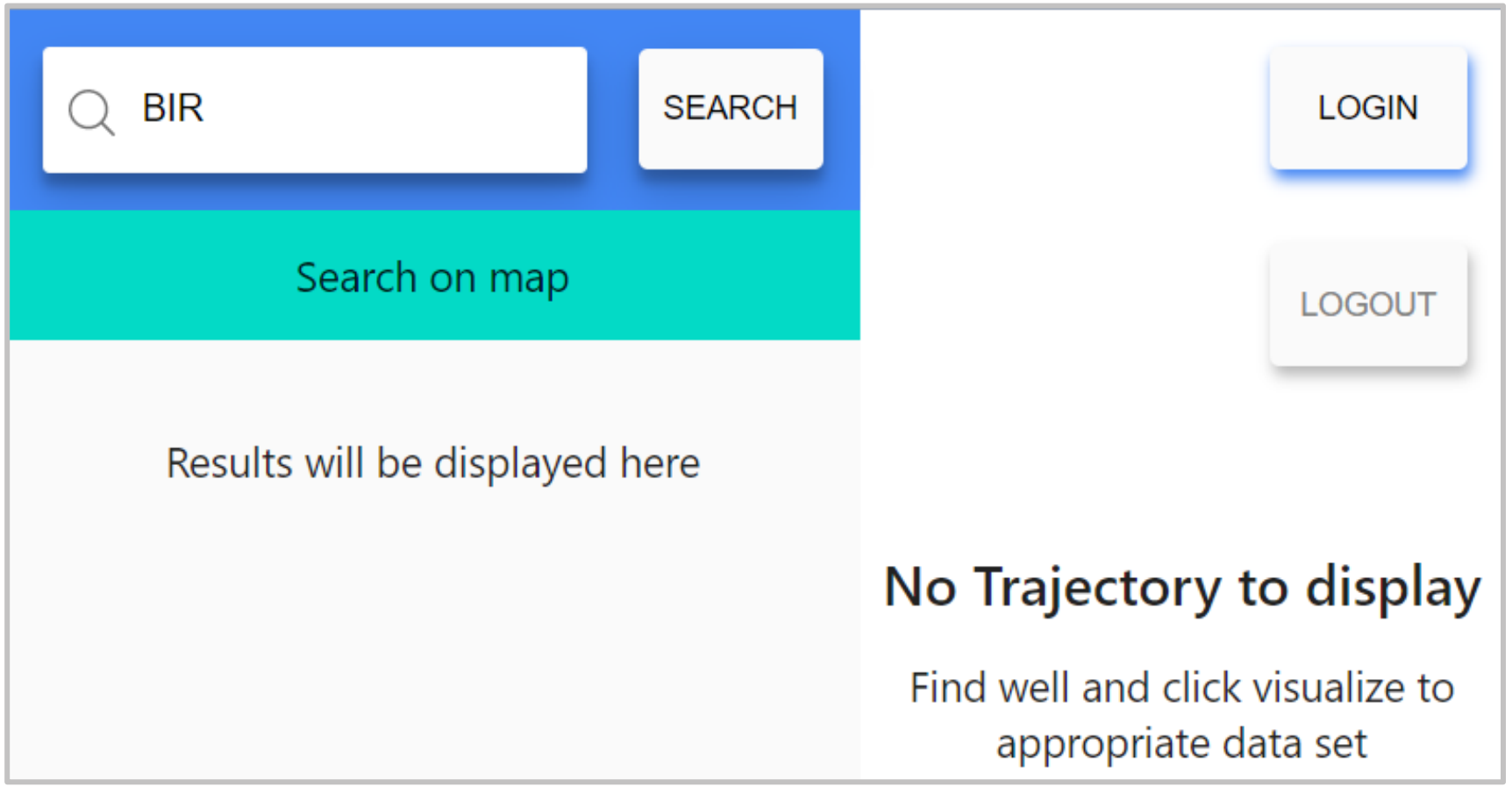
- General list of services is well documented at https://community.opengroup.org/osdu/documentation/-/wikis/Core-Services-Overview

- Every service has its own swagger page
  - <osdu_endpoint>/api/search/v2/swagger-ui.html
  - <osdu_endpoint>/api/file/swagger-ui.html



Two search options

- Full-text
- Geo-spatial



Search Service: Make a request

- get **osdu_instance_url** from your OSDU administrator or cloud provider

```
79  OSDU_API_SEARCH_URL="https://<osdu_instance_url>/api/search/v2/query"
80
81  search_response = requests.post(
82          OSDU_API_SEARCH_URL,
83          headers=headers,
84          json=search_query)
85
86  search_response_json = search_response.json()
```

Search Request: Construct a query

```
66  # get well name from request, such as "A05-01"
67  well_name = request.params.get("well_name")
68
69  search_query = {
70      "kind": "opendes:wks:master-data--Well:1.0.0",
71      "query": f"data.FacilityID: \"{well_name}\"",
72      "returnedFields": [
73          "data.FacilityID",
74          "id",
75          "data.SpatialLocation.Wgs84Coordinates.geometries"
76      ]
77  }
```

Search Request: View response data


```
{
    "results": [
        {
            "data": {
                "FacilityID": "A05-01",
                "SpatialLocation.Wgs84Coordinates": {
                    "geometries": [
                        {
                            "coordinates": [
                                3.51906683,
                                55.68101428
                            ],
                            "type": "point"
                        }
                    ]
                }
            },
            "id": "opendes:master-data--Well:8438"
        }
    ],
    "totalCount": 1
}
```

Spatial Search: Construct a query

```
# create Search API request payload:
search_query = {
    "kind": "opendes:wks:master-data--Well:1.0.0",
    "spatialFilter": {
        "field": "data.SpatialLocation.Wgs84Coordinates",
        "byGeoPolygon": {
            "points": [
                {
                    "longitude": 5.1580810546875,
                    "latitude": 52.859180945520826
                },
                ...
            ]
        },
    },
    "returnedFields": [
        "data.FacilityID",
        "id",
        "data.SpatialLocation.Wgs84Coordinates.geometries"
    ],
}
```

Search Request: Query related records

```
97   well_id = request.params.get("well_id")
98
99   search_query = {
100      "kind": "opendes:wks:master-data--Wellbore:1.0.0",
101      "query": f"data.WellID: \"{well_id}\"",
102      "returnedFields": [
103          "id",
104          "data.NameAliases.AliasName"
105      ]
106  }
107
108  search_response = requests.post(
109      OSDU_API_SEARCH_URL,
110      headers=headers,
111      json=search_query)
```

Fetch data: Signed URL and file download

```python
file_id = ids_list[0]

file_response = requests.get(
    f"{OSDU_API_FILE_URL}/{file_id}/downloadURL",
    headers=headers
)

signed_url = file_response.json()["SignedUrl"]

blob_storage_response = requests.get(signed_url)
file_content = blob_storage_response.content
```


```
{
    "SignedUrl": "https://[redacted]/file-persistent-area/
        osdu-user%2Fopen-test-data%2FR3%2Fsource_files%2FTNO_Source_Files%2Ftrajectories%2F5496.csv?sv
        sig=F7P%2F6XcYZdr%2Bcg11JESc0yMCkRLX36HgmeGE%2BgH%2BWms%3D"
}
```

Fetch data: Obtain dataset ids

```python
search_query = {
    "kind": "opendes:wks:work-product-component--WellboreTrajectory:1.0.0",
    "query": f"data.WellboreID: \"{wellbore_id}\"",
    "returnedFields": [
        "data.Datasets"
    ]
}

search_results = call_search_api(request, search_query)

ids_list = [record["data"]["Datasets"] for record in search_results["results"]]
```

```json
{
  "results": [
    {
      "data": {
        "Datasets": [
          "opendes:dataset--File.Generic:e1a9aaf71a2442e59db0dada8be7d86d:"
```

Application Development Guide

- https://community.opengroup.org/osdu/tutorials/guides/-/blob/master/OSDU_Release2_Application_Development_Guide.pdf
- Comprehensive and structured information about application development for OSDU platform
- Written for R2 but most of the information is relevant
- This document is available to everyone as our contribution to OSDU