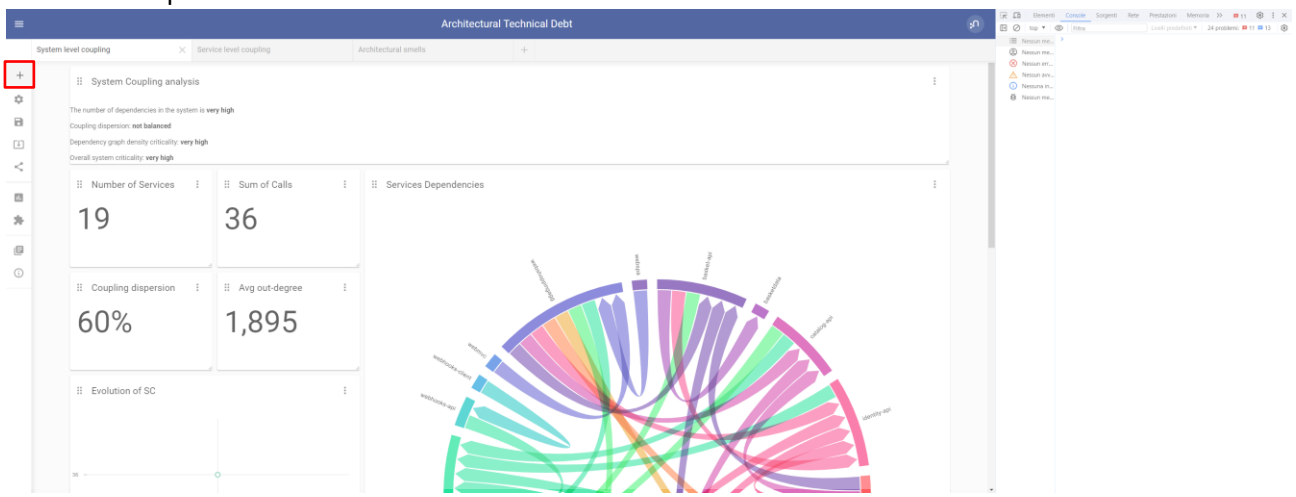
DTOps tool configuration

Table of contents

Generate the dependency graph

1. Open the code located at 'server\coupling-metrics\DepGrapGenerator.tsx.' Then, assign the raw GitHub URL of the Docker Compose file for the project you want to analyse to the variable 'dockerComposeFileUrl'.
2. Open the DTOps tool in your browser and the developer tools.
3. Click on the plus icon.



4. Wait until you see “Dependency graph generated” in the console.

Automatic tool configuration

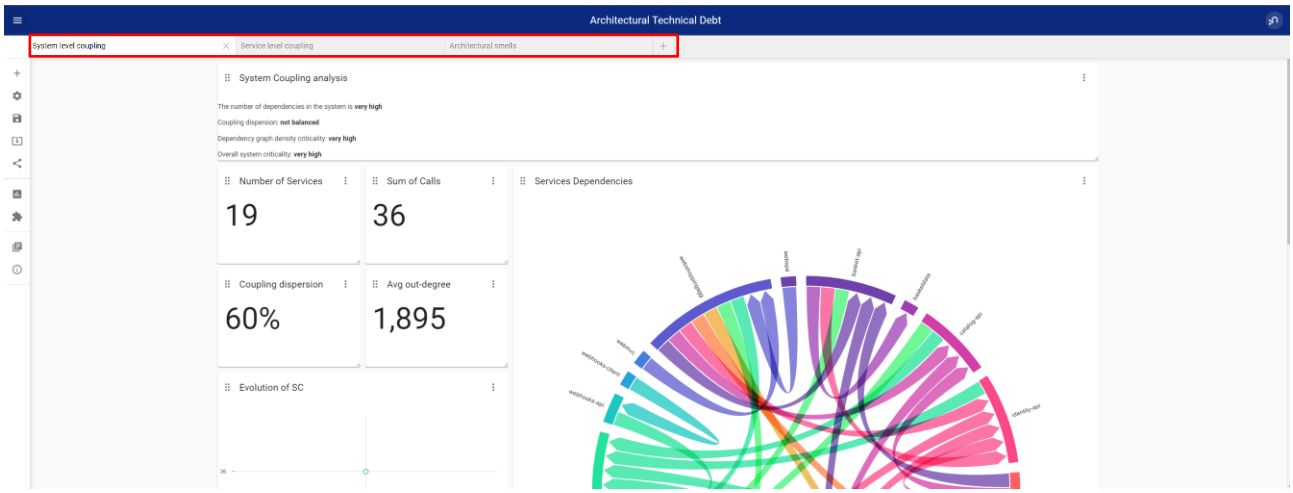
If you have not done it already, open the DTOps tool in your browser, click on the “Load Dashboard” button, chose the “select from file” option and select the file “Architectural_TD_dashboard.json” from the project’s directory.

Manual tool configuration

Step 1: create different pages

Create three different pages with the following titles:

1. System level coupling
2. Service level coupling
3. Architectural smells



Step 2: create the views for the System level coupling page

Views creation

1. System coupling analysis
    - Configuration:
        - Title: System coupling analysis
        - Type: Coupling
        - Cypher query:

```
MATCH (s:System)
RETURN s
```

2. Number of Services
    - Configuration:
        - Title: Number of Services
        - Type: Single Value
        - Cypher query:

```
MATCH (Sm:System)
RETURN Sm.N
LIMIT 1
```

3. Sum of calls
    - Configuration:
        - Title: Sum of Calls
        - Type: Single Value
        - Cypher query:

```
MATCH (Sm:System)
RETURN Sm.SC
LIMIT 1
```

4. Coupling dispersion

- Configuration:
  - Title: Coupling dispersion
  - Type: Single Value
  - Cypher query:

```
MATCH (Sm:System)
RETURN Sm.giniADS
LIMIT 1
```

5. Average out-degree

- Configuration:
  - Title: Avg out-degree
  - Type: Single Value
  - Cypher query:

```
MATCH (Sm:System)
RETURN Sm.ADSA
LIMIT 1
```

6. Services Dependencies

- Configuration:
  - Title: Services Dependencies
  - Type: Chord Diagram
  - Cypher query:

```
MATCH (n:Service)
OPTIONAL MATCH (n)-[r]->(m)
RETURN n,r,m
```

7. Evolution of SC

- Configuration:
  - Title: Evolution of SC
  - Type: Line Chart
  - Cypher query:

```
MATCH (n:System)
WITH collect(n) as nodes
WITH apoc.coll.zip(nodes, range(0, size(nodes))) as
pairs
UNWIND pairs as pair
RETURN pair[0].SC as SC, pair[1] as ID
```

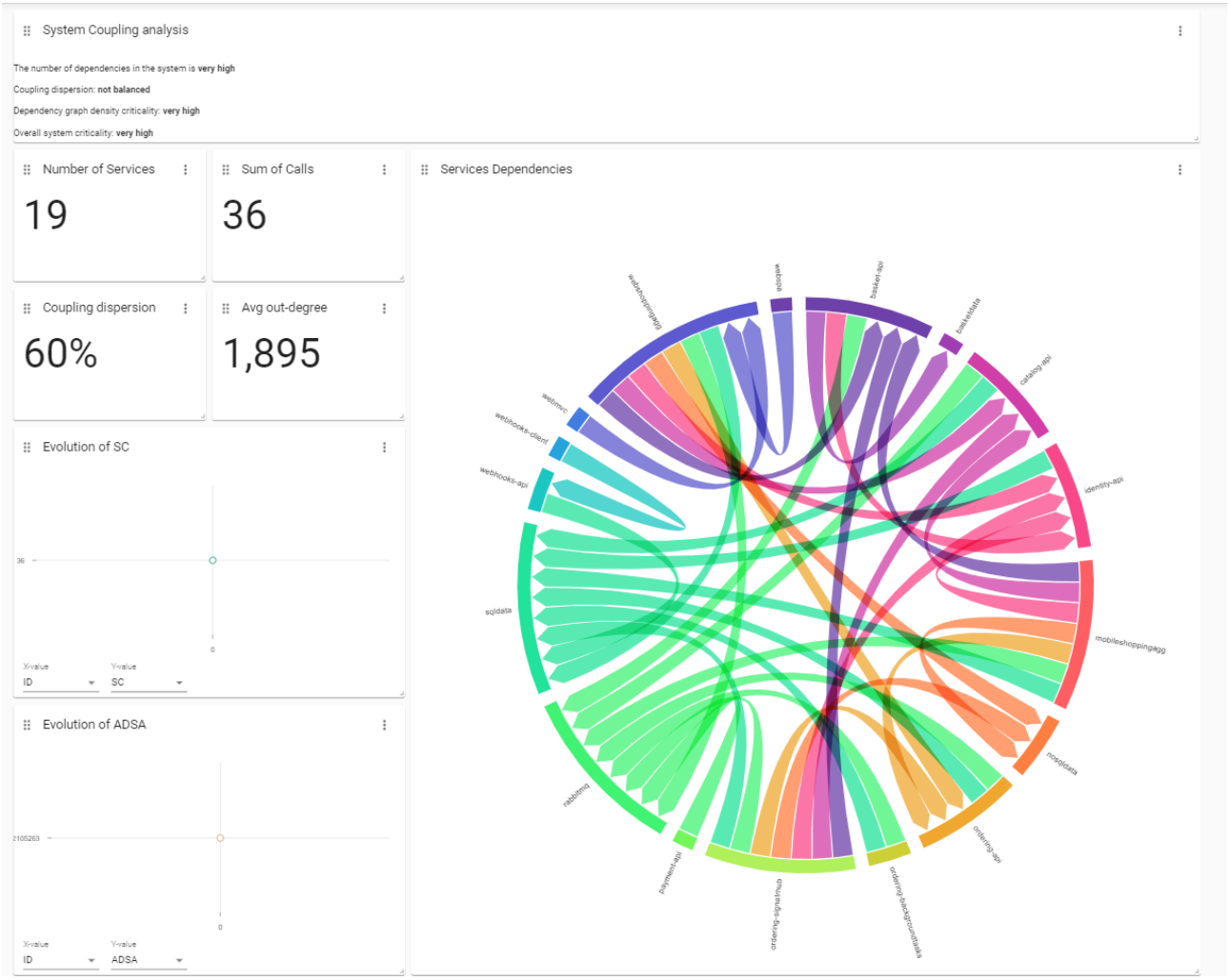
8. Evolution of ADSA

- Configuration:
  - Title: Evolution of ADSA
  - Type: Line Chart
  - Cypher query:

```
MATCH (n:System)
WITH collect(n) as nodes
WITH apoc.coll.zip(nodes, range(0, size(nodes))) as
pairs
UNWIND pairs as pair
RETURN pair[0].ADSA as ADSA, pair[1] as ID
```

Layout

Arrange and resize the views to match the layout shown in the following figure.



Step 3: create the views for the Service level coupling page
Move to the "Service level coupling" page and follow the next steps.

Views creation

1. Markdown view
   - Configuration:
     - Title: *insert two spaces to make the title disappear
     - Type: Markdown
     - Markdown text:

```
# Dependencies between services
```

2. Services' ADS
   - Configuration:
     - Title: Services' ADS
     - Type: Bar Chart
     - Cypher query:

```
MATCH (s:Service)
WHERE s.isResource = false
RETURN s.name AS Title, s.ADS AS ADS
ORDER BY s.ADS DESC
```

- Advanced settings
    - Margin Right (px) : 50
    - Margin Bottom (px): 120

3. Services' Dependencies (outgoing)

- Configuration:
    - Title: Services' Dependencies (outgoing)
    - Type: Chord Diagram Single Service
    - Cypher query:

```
MATCH (n:Service)-[r]->(m:Service)
RETURN n,r,m
ORDER BY n.ADS DESC
```

4. Services' AIS
- Configuration:
    - Title: Services' AIS
    - Type: Bar Chart
    - Cypher query:

```
MATCH (s:Service)
WHERE s.isResource = false
RETURN s.name AS Title, s.AIS AS AIS
ORDER BY s.AIS DESC
```

- Advanced settings
    - Margin Right (px) : 50
    - Margin Bottom (px): 120

5. Services' Dependencies (ingoing)

- Configuration:
    - Title: Services' Dependencies (ingoing)
    - Type: Chord Diagram Single Service
    - Cypher query:

```
MATCH (n:Service)<-[r]-(m:Service)
WHERE n.isResource = false
RETURN n, r, m
```

6. Markdown view
   - Configuration:
     - Title: *insert two spaces to make the title disappear
     - Type: Markdown
     - Markdown text:

```
# Shared resources
```

7. Resources in-degree
   - Configuration:
     - Title: Resources in-degree
     - Type: Bar Chart
     - Cypher query:

```
MATCH (n:Service)-[r]->(m:Service)
WHERE m.isResource = true
WITH m, COLLECT(r) AS relationships
RETURN m AS Database,
```

   - Advanced settings
     - Margin Right (px) : 50
     - Margin Bottom (px): 120

8. Dependencies with resources
   - Configuration:
     - Title: Dependencies with resources
     - Type: Chord Diagram Single Service
     - Cypher query:

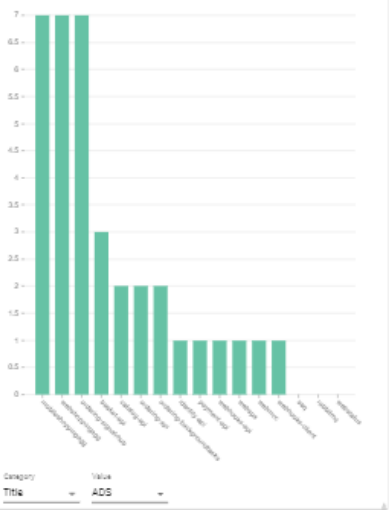
```
MATCH (n:Service)-[r]->(m:Service)
WHERE m.isResource = true
WITH m, COLLECT(r) AS relationships, n, r
RETURN m, r, n, size(relationships) as inDegree
ORDER BY inDegree DESCORDER BY n.AIS DESC
```

Layout

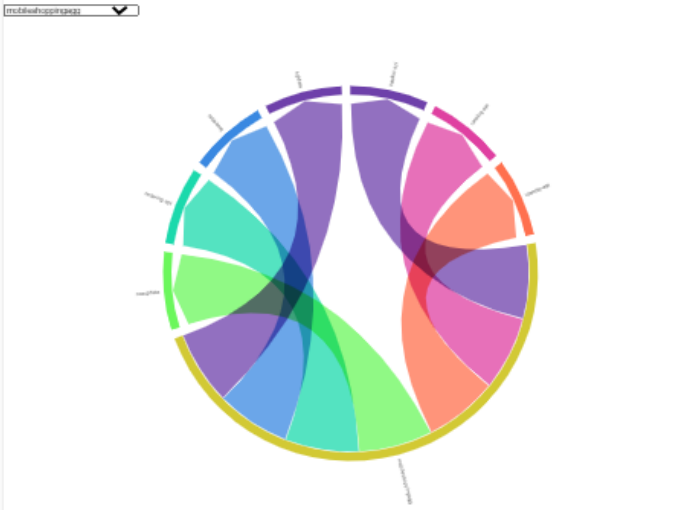
Arrange and resize the views to match the layout shown in the following figure.

Dependencies between services

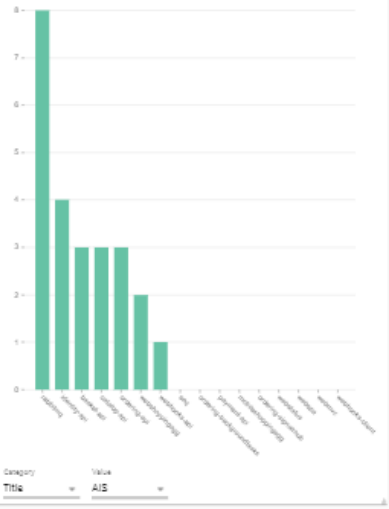
Services' ADS



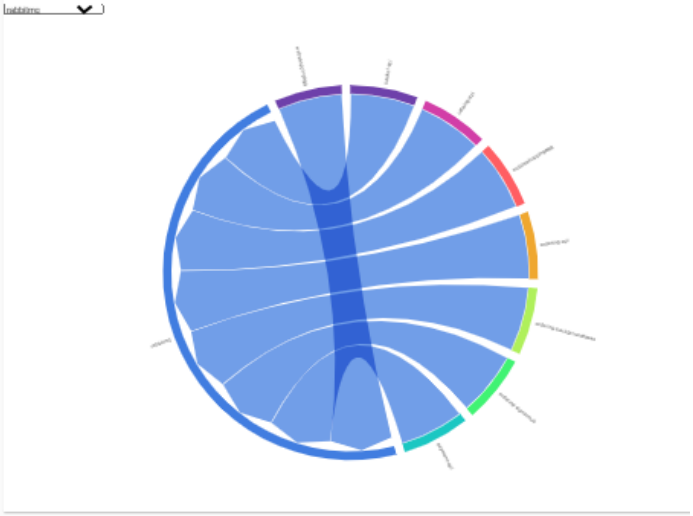
Services' Dependencies (outgoing)



Services' AIS

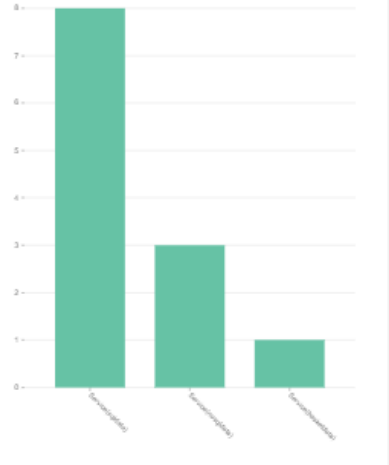


Services' Dependencies (ingoing)

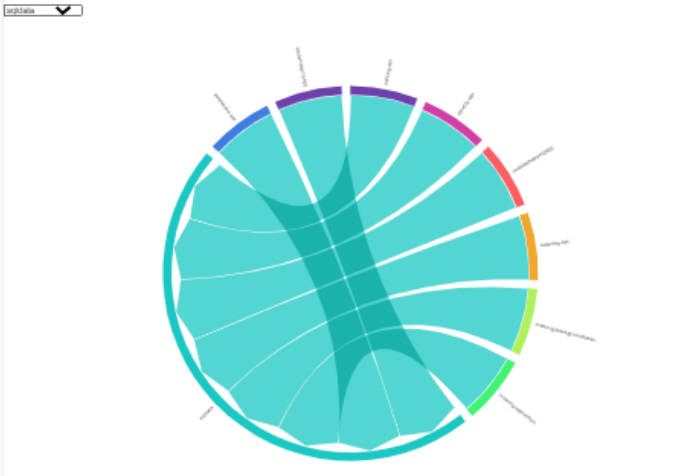


Shared resources

Resources in-degree



Dependencies with resources



Step 4: create the views for the System level coupling page
Move to the "Service level coupling" page and follow the next steps.

Views creation

1. Markdown view
   - Configuration:
     - Title: *insert two spaces to make the title disappear
     - Type: Markdown
     - Markdown text:

```
# Hub-Like Dependencies
```

2. Hub-Like Dependencies Graphs
   - Configuration:
     - Title: Hub-Like Dependencies Graphs
     - Type: Graph
     - Cypher query:

```
MATCH (n:Service)-[r]-(m:Service), (S:System)
WHERE n.ADS > S.adsMedian AND n.AIS > S.aisMedian AND
abs(n.ADS - n.AIS) < (n.degree / 4)
WITH n, collect(r) AS rels, collect(m) AS deps,
count(distinct n) + count(distinct m) AS size
RETURN n, rels, deps, size
ORDER BY size DESC
```

     The formula used to identify the Hub-Like Dependencies can be found at (F. A. Fontana 2016)

   - Advanced settings
     - Architectural Smell: Hub-Like Dependency
3. Markdown view
   - Configuration:
     - Title: *insert two spaces to make the title disappear
     - Type: Markdown
     - Markdown text:

```
# Cyclic Dependencies
```

4. Cyclic Dependencies Graphs
   - Configuration:
     - Title: Cyclic Dependencies Graphs

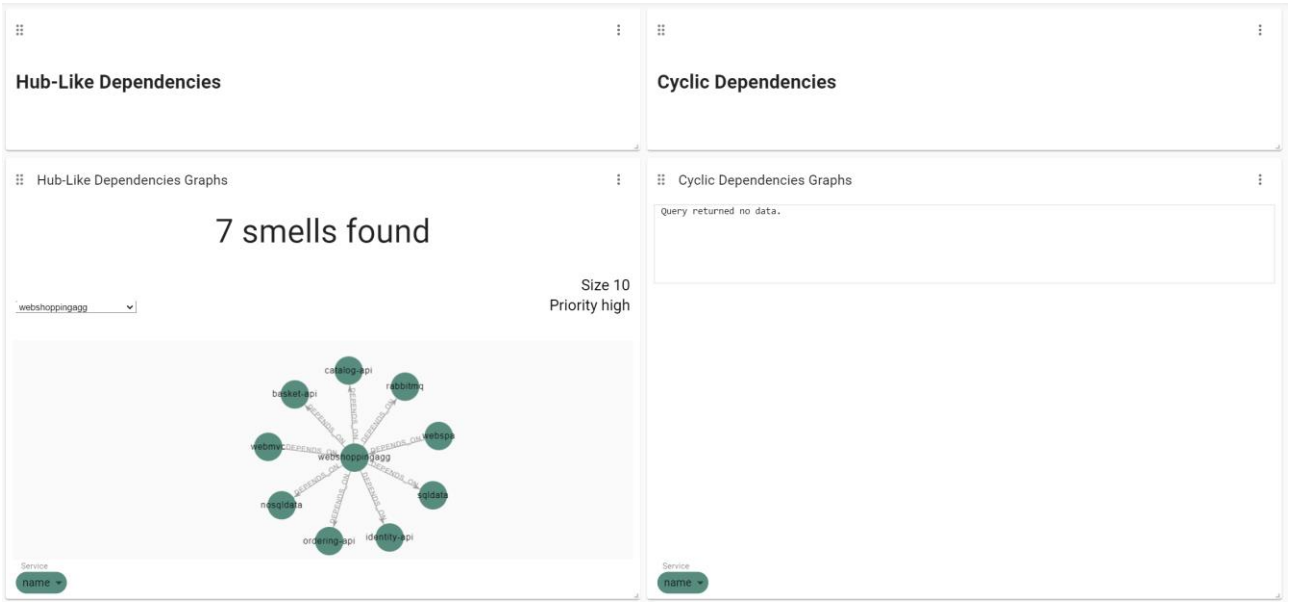
- Type: Graph
- Cypher query:

```
MATCH p=(n)-[r:DEPENDS_ON*]->(n)
WITH n, nodes(p) AS deps, r, length(p) AS size
ORDER BY size DESC
RETURN n, deps, r, size
```

- Advanced settings
  - Architectural Smell: Cyclic Dependency

Layout

Arrange and resize the views to match the layout shown in the following figure.



References

F. A. Fontana, I. Pigazzini, R. Roveda and M. Zanoni. “Automatic Detection of Instability Architectural Smells.” Raleigh, NC, USA: IEEE International Conference on Software Maintenance and Evolution (ICSME), 2016. 433-437. *doi: 10.1109/ICSME.2016.33.*