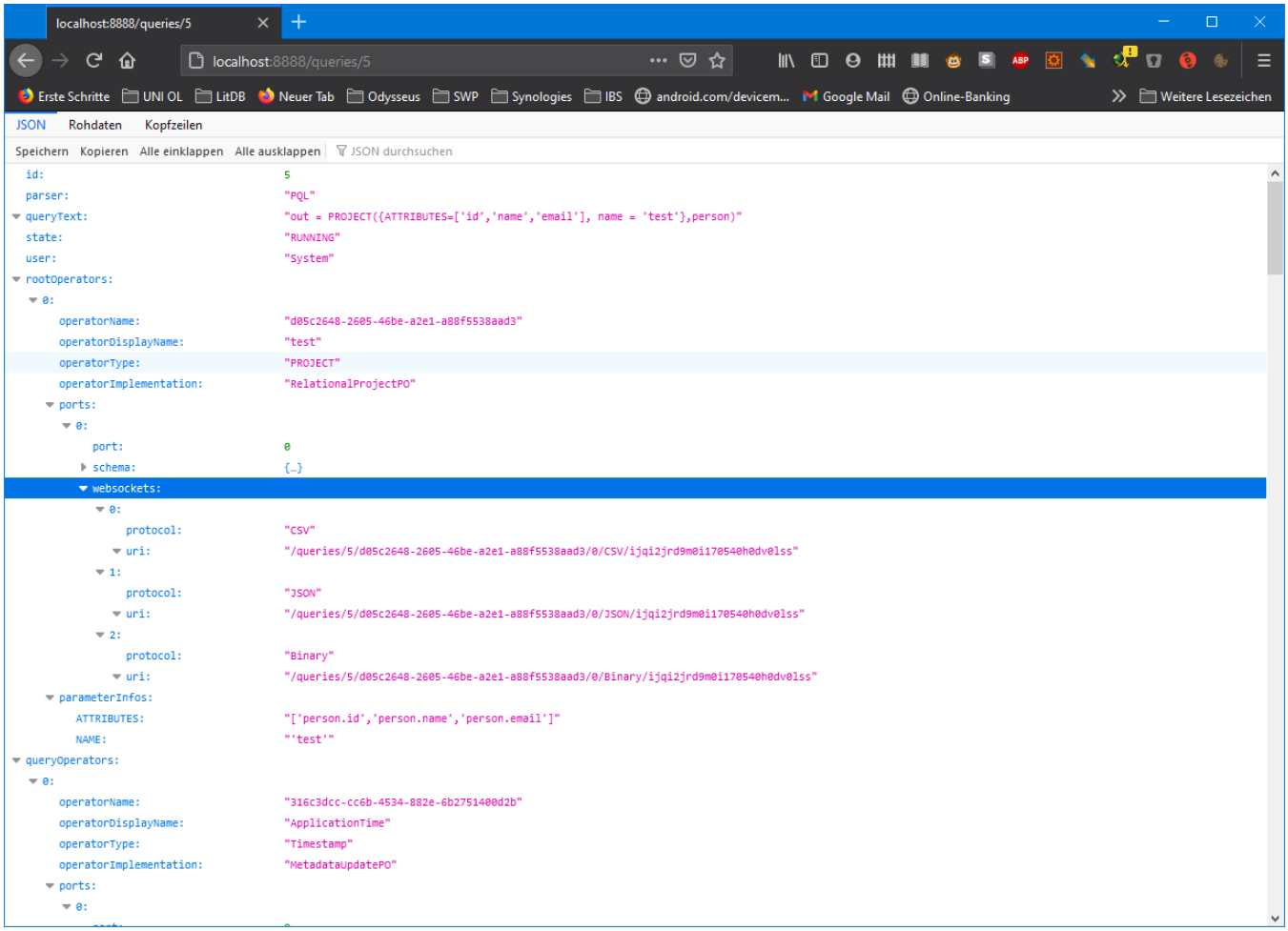
Reading Query Results via REST/Websocket

You can access the installed queries of the server with the queries endpoint. If e.g. the would be a query with id 0, you could access the query with: http://localhost:8888/queries/0

The result will look similar to the following:

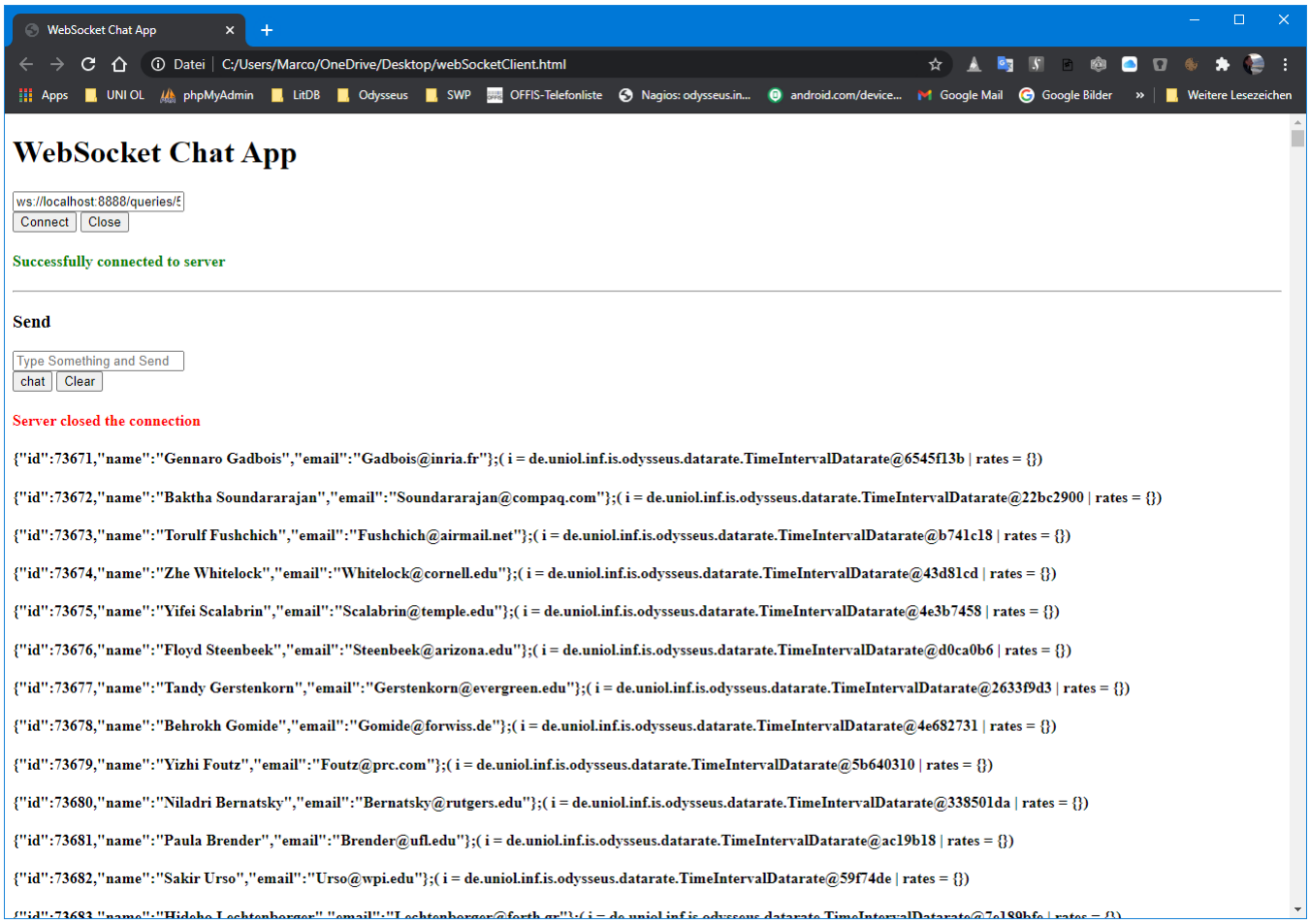


The important part for accessing the results can be found in the websockets section. For each root operator (i.e. the sink of a plan) you could find information how to access the results via a web socket connection.

Different kinds of data can be access. E.g. in der example above, the endpoint "/queries/5/d05c2648-2605-46be-a2e1-a88f5538aad3/0/CSV/ijqi2jrd9m0i170540h0dv0lss" will delivers the results of the query via the CSV protocol handler. (Binary will be the Odysseus Protocol Handler)

Reading in a Browser

You can access the result e.g. in a browser with java script. (see https://github.com/wso2/msf4j/blob/master/samples/websocket/chatApp/js-client/index.html for an example)



Attention: In JSON the result contains the metadata appended after JSON-Document with an ";" as separator.

Reading in Odysseus

You could also use this output as an input to another odysseus operator as in the following

```
#PARSER PQL
#RUNQUERY
in = ACCESS({
        transport = 'WebsocketClient',
        wrapper = 'GenericPush',
        datahandler = 'Tuple',
        protocol = 'CSV',
        source = 'access',
        readMetadata = 'true',
        options = [
                ['uri','ws://localhost:8888/queries/5/d05c2648-2605-46be-a2e1-a88f5538aad3/0/CSV
/ijqi2jrd9m0i170540h0dv0lss']
        ],
        schema = [
                ['person','id', 'INTEGER'],
                ['person','name', 'STRING'],
                ['person','mail', 'STRING']
        ]
        }
)
```

The same information can be found for all other operators in the queryOperators section.