System Architecture Specification

TINF21C, Software Engineering I

Practical project 2022/23

Project

AAS-Webclient

Customer

Markus Rentschler, Christian Holder

Rothebühlplatz 41, 70178 Stuttgart

Supplier

Project Leader: Samara Dominik (inf21001@lehre.dhbw-stuttgart.de)

Product Manager: Martin Rittmann (inf21157@lehre.dhbw-stuttgart.de)

System Architect: Marcel Hintze (inf21056@lehre.dhbw-stuttgart.de)

Test-Manager: Anja Niedermeier (inf21097@lehre.dhbw-stuttgart.de)

Developer: Severin Helms (inf21047@lehre.dhbw-stuttgart.de)

Technical documentor: Tom Engelmann (inf21010@lehre.dhbw-stuttgart.de)

Version	*Date*	*Author*	*Comment*
0.1	01.11.2022	Marcel Hintze	Created Document, added structure, …
0.2	04.11.2022	Marcel Hintze	Added Architectural Concept, System Design, Technical Concepts
1.0	05.11.2022	Marcel Hintze	Final Version

CONTENT

Introduction

The goal of this project is to develop a web-based application which allows the user to easily access the Asset Administration Shell.

System Overview

The system provides the user with a graphical user interface with the possibility of sorting products and filtering them by properties. There is a possibility that the user can include his own products by adding his own server.

Software Environment

The frontend of this application relies on the framework React, which uses the back-end JavaScript runtime environment Node.js. When the application is running, the user can access the GUI with any browser supporting the HTML 5 standard.

The backend of the application is a server, which can easily be accessed by a Rest API.

The frontend and backend communicate via the REST API. Therefore, both the frontend and backend must be configured to use available ports.

Quality Concept

This part of the SAS explains and breaks down the problems that usually arise during the further development of software. This includes concepts for dealing with these problems and thus improving the quality of the final product.

Maintainability

Because every system needs maintenance, a major focus of this project lies within dividing the system into maintainable modules. Dividing the software into smaller modules should help to make the software easier to analyze, maintain and modify.

Usability Concept

The usability of the application is the main factor that determines whether users will consider it worth using. In order to assure usability, the following criteria are to be considered and maintained:

Intuitiveness: The user should not require training or much experience with the application in order to understand and use it. Thus, the layout and features should be self-explanatory.

Efficiency: The application should enable users to achieve high productivity when used. Therefore, the application should require low amount of steps to be taken in order for the user to reach a desired result.

Reliability: The application should also have a low error rate, so users can trust it to produce the correct results consistently.

It is critical that these characteristics are incorporated within the application, however, they must be balanced and compromised between them have to be made, while the functionality of the application has to be maintained.

In order for the GUI to maintain these characteristics, the following guidelines are taken into consideration.

Appearance: The GUI should feature a pleasant color palette, featuring a soft primary and matching secondary color. Additional colors are generally to be avoided, but may be used to signal special meaning, such as green for success and red for warnings.

Consistency: The GUI should use the same color palette across all of its elements and display elements objects in similar ways. Likewise, elements that look similar should behave similarly.

Simplicity: The GUI should always contain a limited amount of elements at one time to allow the user to quickly recognize the displayed elements and their functionality. Different parts of the GUI should be accessible with few and non-convoluted steps.

Expressiveness: Texts should generally be kept short and concise in order to convey meaning quickly. When possible, expressive icons should be used to convey meaning visually. Elements representing physical objects should contain images displaying that object. Special elements, such as input fields, should be marked in a special manner to easily distinguish them.

Architectural Concept

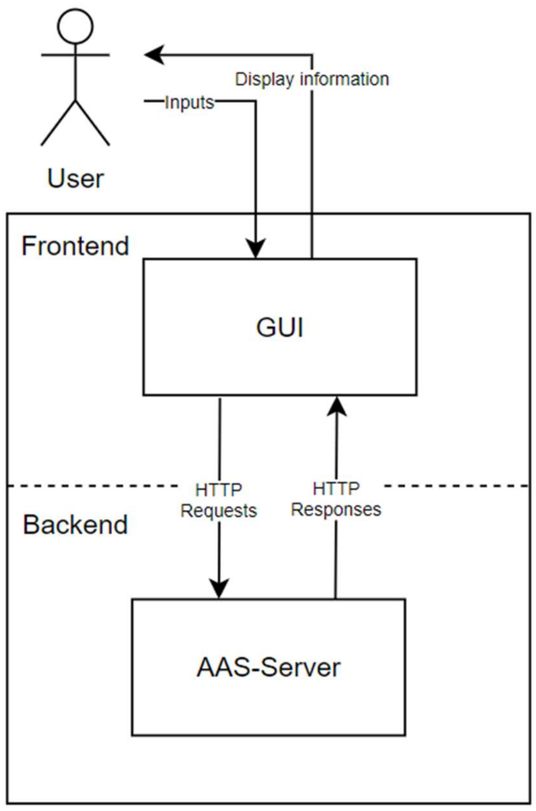
The application is split into two layers, the presentation layer and the execution layer.

The presentation layer consists of the graphical user interface, which is tasked with handling user interaction. It displays information to the user using data received from the execution layer, and sends requested changes to it.

The execution layer contains most of the complex business logic. It receives requests from the presentation layer and responds to these with the requested data.

System Design

The application is designed in a frontend-backend architecture. The frontend represents the presentation layer, while the backend represents the execution layer. They communicate via the Rest API, with the frontend sending HTTP requests and the backend returning matching HTTP responses.



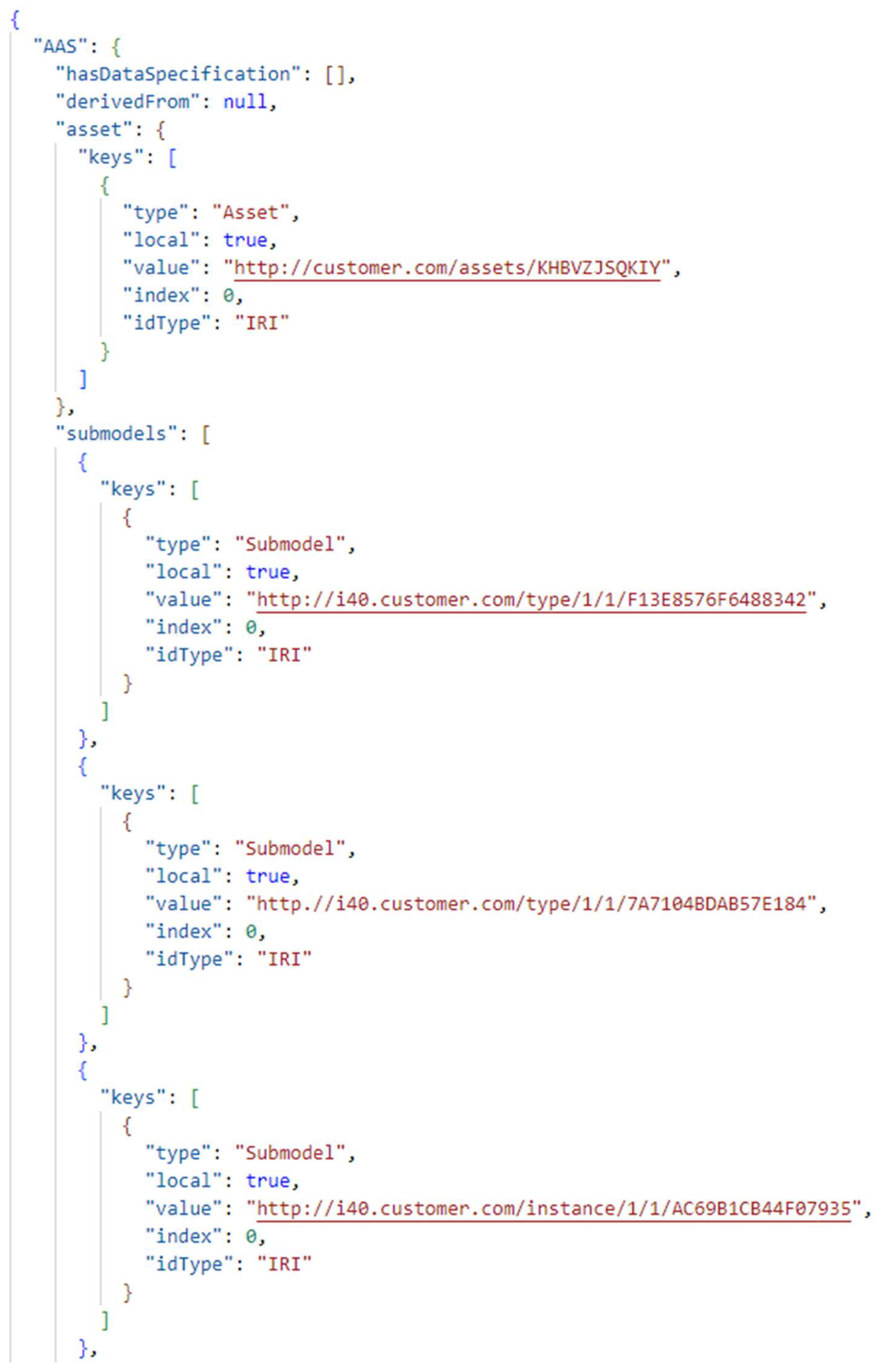
The frontend consists of one module, the GUI. The GUI is an React based website, which can be accessed from the user’s web browser. It is tasked with displaying information to the user and taking their inputs. This works by sending HTTP requests to the backend, which responds with the data to display. The frontend also performs minor logic like filtering and ordering search results.

The backend is a simple server, which handles the incoming HTTP requests by sending the responses back to the frontend. The responses are mostly in a JSON style.

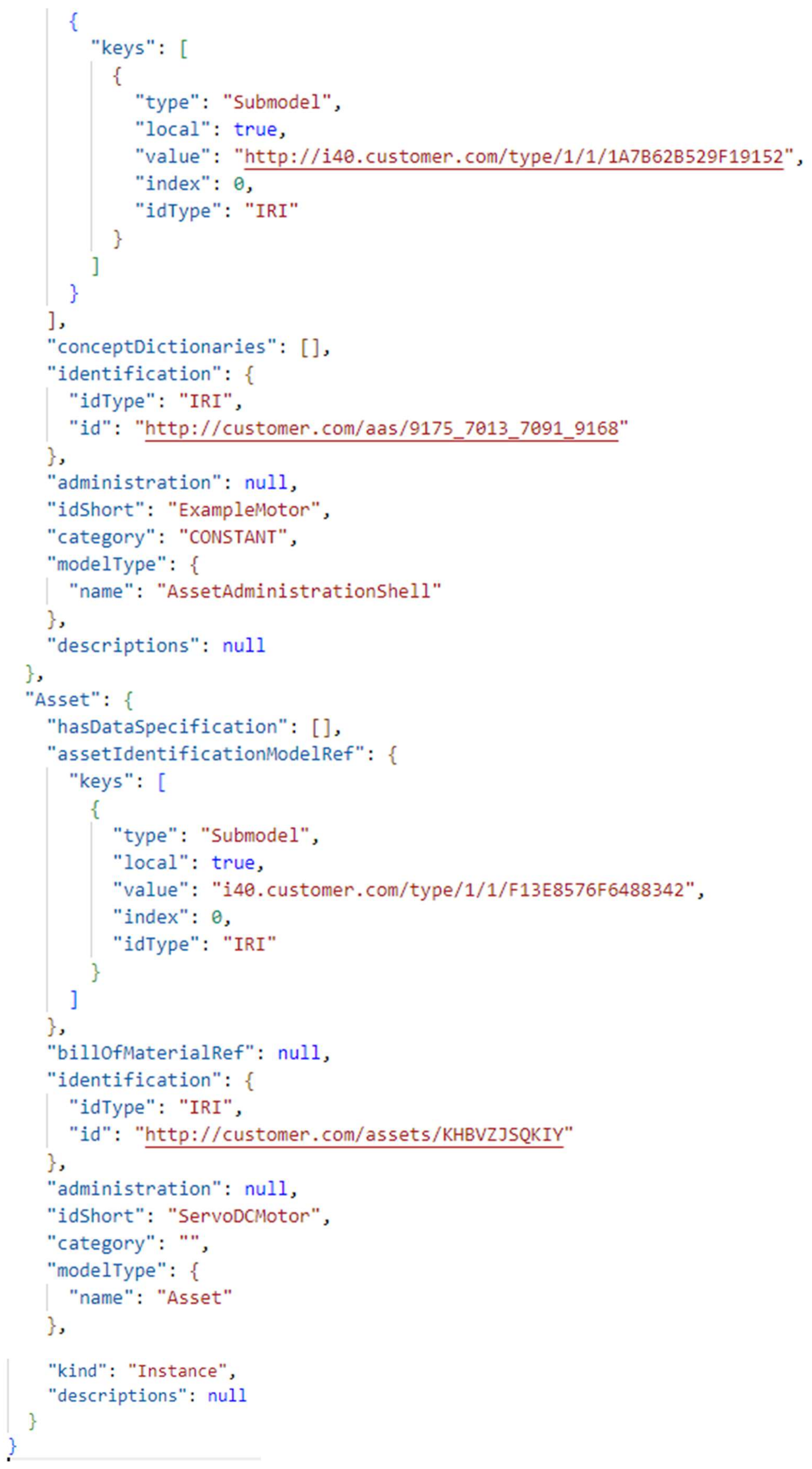
Technical Concepts

Data model

An approximate overview of the data model can be found in the SRS. Considering the aspects that are necessary a JSON model has been created that is usable to communicate between frontend and backend. The model looks as follows:



```
{
  "AAS": {
    "hasDataSpecification": [],
    "derivedFrom": null,
    "asset": {
      "keys": [
        {
          "type": "Asset",
          "local": true,
          "value": "http://customer.com/assets/KHBVZJSQKIY",
          "index": 0,
          "idType": "IRI"
        }
      ]
    },
    "submodels": [
      {
        "keys": [
          {
            "type": "Submodel",
            "local": true,
            "value": "http://i40.customer.com/type/1/1/F13E8576F6488342",
            "index": 0,
            "idType": "IRI"
          }
        ]
      },
      {
        "keys": [
          {
            "type": "Submodel",
            "local": true,
            "value": "http://i40.customer.com/type/1/1/7A7104BDAB57E184",
            "index": 0,
            "idType": "IRI"
          }
        ]
      },
      {
        "keys": [
          {
            "type": "Submodel",
            "local": true,
            "value": "http://i40.customer.com/instance/1/1/AC69B1CB44F07935",
            "index": 0,
            "idType": "IRI"
          }
        ]
      },
```



```
      {
        "keys": [
          {
            "type": "Submodel",
            "local": true,
            "value": "http://i40.customer.com/type/1/1/1A7B62B529F19152",
            "index": 0,
            "idType": "IRI"
          }
        ]
      }
    ],
    "conceptDictionaries": [],
    "identification": {
      "idType": "IRI",
      "id": "http://customer.com/aas/9175_7013_7091_9168"
    },
    "administration": null,
    "idShort": "ExampleMotor",
    "category": "CONSTANT",
    "modelType": {
      "name": "AssetAdministrationShell"
    },
    "descriptions": null
  },
  "Asset": {
    "hasDataSpecification": [],
    "assetIdentificationModelRef": {
      "keys": [
        {
          "type": "Submodel",
          "local": true,
          "value": "i40.customer.com/type/1/1/F13E8576F6488342",
          "index": 0,
          "idType": "IRI"
        }
      ]
    },
    "billOfMaterialRef": null,
    "identification": {
      "idType": "IRI",
      "id": "http://customer.com/assets/KHBVZJSQKIY"
    },
    "administration": null,
    "idShort": "ServoDCMotor",
    "category": "",
    "modelType": {
      "name": "Asset"
    },

    "kind": "Instance",
    "descriptions": null
  }
}
```

Persistence

The application does not require storing data between sessions. The data is read but never modified by the application. The files containing this data can be manually edited to change the product.

User Interface

The graphical user interface (GUI) is the interface between user and program logic. The user can use this GUI to list all parts on the server and sort them. The user can also get more detailed information about the parts.

Ergonomics

It is important for an ergonomic GUI to be intuitive, and the user experience as appealing as possible. This includes making sure that the GUI is visually appealing and that it is designed in such a way that the user intuitively understands how to use it.

Testability

The software is composed of different modules. These modules are tested separately. The system tests the system test plan will provide more information, and the system test report will contain the results.

Availability

The software is only distributed on GitHub.