Smart Service Development UE

Group Assignment 3 - Project Specification

Lukas D'Angelo 11713269, Patrick Eder 01607220, Benedikt Görgei 11710412

May 11, 2022

Contents

1 Scoping and Structuring

1.1 Abstract

The topic of smart home technology gained importance over the last decade, introducing the concept of networking devices and equipment in domestic areas. Increasing demand for renewable energy and efficient usage creates a need for intelligent smart-home systems to contribute to the goals of EUs *Energy Efficiency Directive* as well as creating a sustainable, reliable, scalable application framework. On the one hand this framework integrates hardware peripherals accessed by an MQTT broker running on a Raspberry Pi. On the other hand a hardware and service engine provides logic and a database for the application. Control and state APIs are defined for standardized communication between hard- and software engines, wherein an asyncAPI is defined for data exchange between MQTT-Server and hardware engine. A user can access the smart home system by entering the right credentials on a website. The web server is created with the flask python library. A graphical dashboards provides the user with weather information requested from MET Norway Weather API v.3, status of sensors and relais and allows to set the status of actuators and define simple timer switching logic in an interactive way. Alternatively a user may also access the hardware engine directly via its exposed API. Overall this smart service is meant to be a contribution to solve the environmental and energy management challenges of the 21'st century.

2 Services and their Interfaces:

2.1 UML Component and Sequence Diagram

Figure 1 shows the main components of the project, Figure 2 illustrates a typical user interaction in form of a sequence diagram.

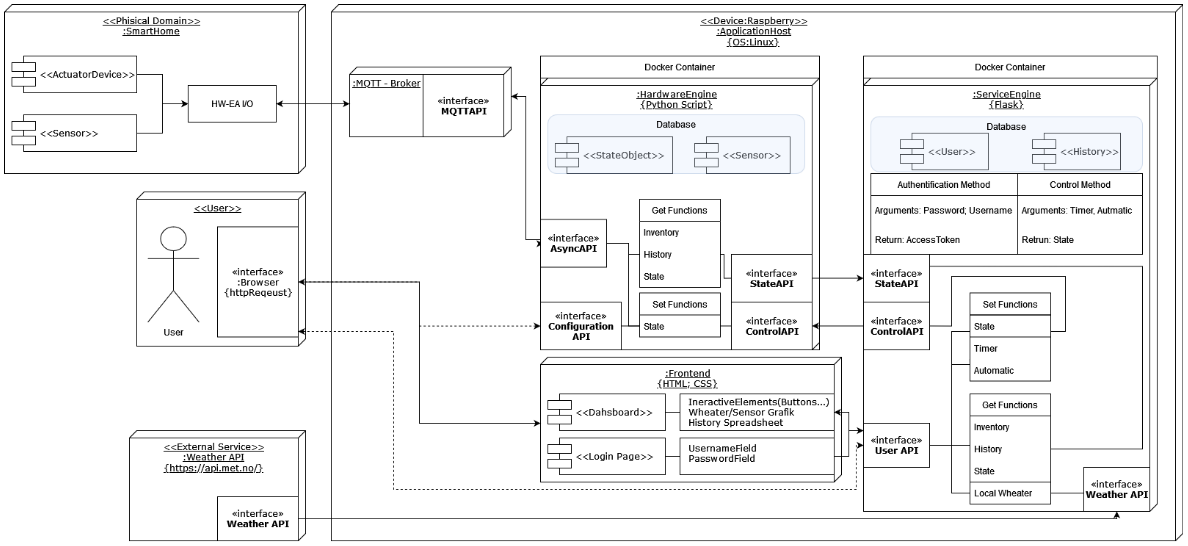

Figure 1: Component diagram of the project

2.2 Physical Domain

Description: The Physical Domain is the service containing all the entities to be controlled and whose state is to be sensed, including the I/O modules, thus provide the interface between the actual hardware to control and the software.

AsyncAPI (MQTT):

```
asyncapi: 2.4.0
info:
  description: |
    Group Assignment 3: Project Specification - MQTT
  version: "0.1"
  title: SSD Project - Physical Domain
  termsOfService: 'http://swagger.io/terms/'
  contact:
    email: lukas.dangelo@student.tugraz.at
  license:
    name: Apache 2.0
    url: 'http://www.apache.org/licenses/LICENSE-2.0.html'
servers:
  test:
    url: tcp://10.0.0.1:1883
    description: Development server
    protocol: mqtt
channels:
```

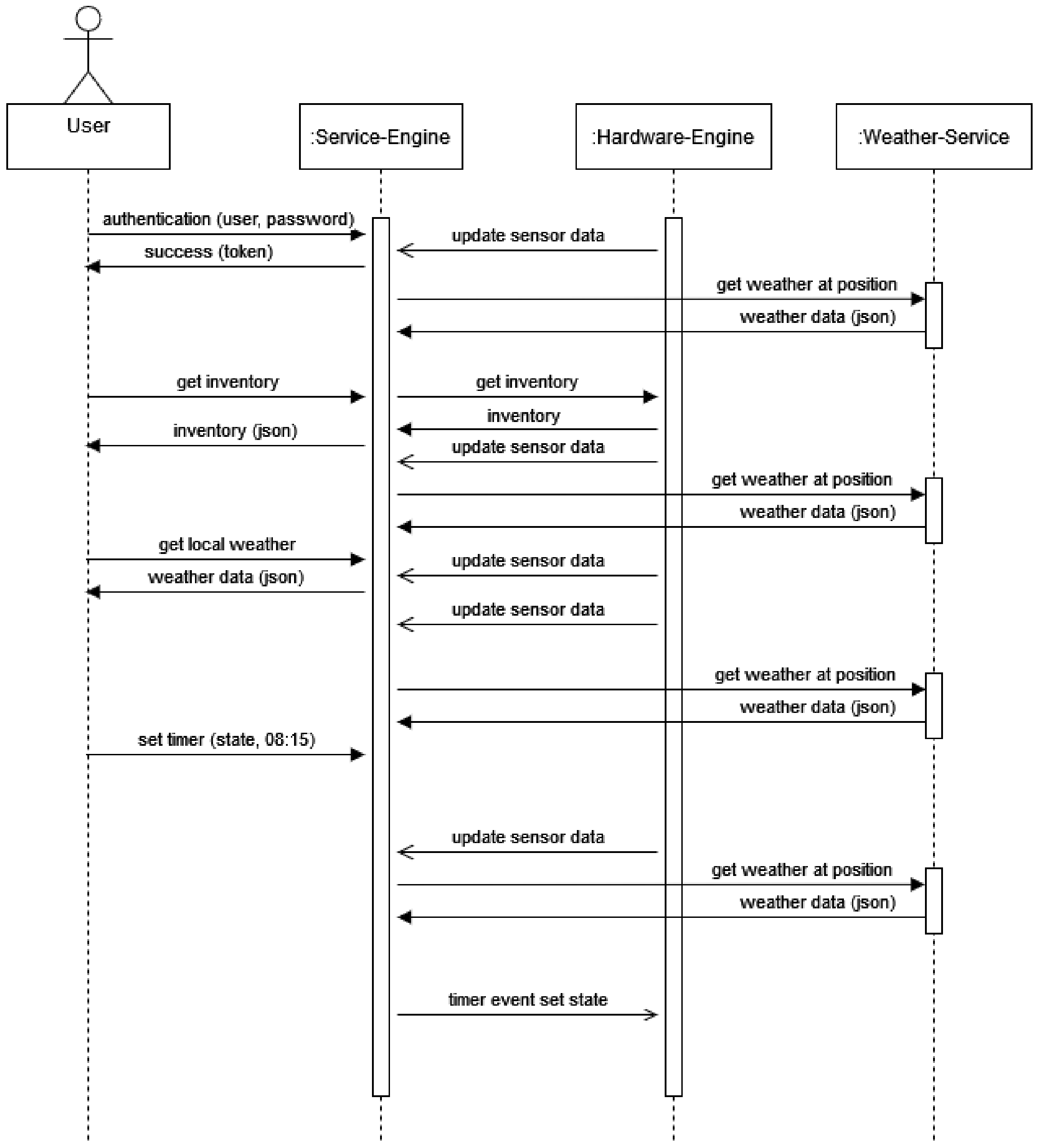

Figure 2: Sequence diagram of the project

```
  ha-ea/module-{module_id}/outputs/K{output_number}:
    parameters:
        module_id:
          $ref: '#/components/parameters/module_id'
        output_number:
          $ref: '#/components/parameters/output_number'
    subscribe:
      message:
        $ref: '#/components/messages/set_output'
  ha-ea/module-{module_id}/outputs/K{output_number}_state:
    parameters:
        module_id:
          $ref: '#/components/parameters/module_id'
        output_number:
          $ref: '#/components/parameters/output_number'
    publish:
      message:
        $ref: '#/components/messages/set_output'
  ha-ea/module-{module_id}/outputs/I{input_number}:
    parameters:
        module_id:
          $ref: '#/components/parameters/module_id'
        input_number:
          $ref: '#/components/parameters/input_number'
    publish:
      message:
        $ref: '#/components/messages/get_input'
  ha-ea/module-{module_id}/bus/sensors:
    parameters:
        module_id:
          $ref: '#/components/parameters/module_id'
    publish:
      message:
        $ref: '#/components/messages/sensor_object'

components:
  messages:
    set_output:
      payload:
        type: string
        description: Value of the output to set
    get_input:
      payload:
        type: string
        description: Value of the input to read
    sensor_object:
      payload:
        type: object
        description: Value of the sensors on the BUS interface on a specified module
        parameters:
          address:
            type: string
        sensor_value:
          type: number
          schema: float
  parameters:
    module_id:
      description: Identifier of the module
      schema:
        type: integer
    output_number:
      description: Number of the output on a specified module
      schema:
        type: integer
    input_number:
      description: Number of the input on a specified module
      schema:
        type: integer
```

2.3 Hardware Engine

Description: Transforms the state data of the physical Domain to data usable for the Service Engine and transforms control data of the Service Engine to data interpretable by the physical domain.

AsyncAPI (MQTT):

```
asyncapi: 2.4.0
info:
  description: |
    Group Assignment 3: Project Specification - MQTT
  version: "0.1"
  title: SSD Project - Hardware Engine
  contact:
    email: lukas.dangelo@student.tugraz.at
  license:
    name: Apache 2.0
    url: 'http://www.apache.org/licenses/LICENSE-2.0.html'
servers:
  test:
    url: tcp://localhost:1883
    description: Development server
    protocol: mqtt
channels:
  ha-ea/module-{module_id}/outputs/K{output_number}:
    parameters:
        module_id:
          $ref: '#/components/parameters/module_id'
        output_number:
          $ref: '#/components/parameters/output_number'
    publish:
      message:
        $ref: '#/components/messages/set_output'
  ha-ea/module-{module_id}/outputs/K{output_number}_state:
    parameters:
        module_id:
          $ref: '#/components/parameters/module_id'
        output_number:
          $ref: '#/components/parameters/output_number'
    subscribe:
      message:
        $ref: '#/components/messages/set_output'
  ha-ea/module-{module_id}/outputs/I{input_number}:
    parameters:
        module_id:
          $ref: '#/components/parameters/module_id'
        input_number:
          $ref: '#/components/parameters/input_number'
    subscribe:
      message:
        $ref: '#/components/messages/get_input'
  ha-ea/module-{module_id}/bus/sensors:
    parameters:
        module_id:
          $ref: '#/components/parameters/module_id'
    subscribe:
      message:
        $ref: '#/components/messages/sensor_object'
components:
  messages:
    set_output:
      payload:
        type: string
        description: Value of the output to set
    get_input:
      payload:
        type: string
        description: Value of the input to read
    sensor_object:
      payload:
        type: object
        description: Value of the sensors on the BUS interface on a specified module
        parameters:
          address:
            type: string
          sensor_value:
            type: number
            schema: float
  parameters:
    module_id:
      description: Identifier of the module
      schema:
        type: integer
    output_number:
      description: Number of the output on a specified module
      schema:
        type: integer
    input_number:
      description: Number of the input on a specified module
      schema:
        type: integer
```

State, Control & Configuration (REST):

```
openapi: 3.0.0
info:
  description: |
    Group Assignment 3: Project Specification - REST
  version: "0.1"
  title: SSD Project - Hardware Engine
  contact:
    email: lukas.dangelo@student.tugraz.at
  license:
    name: Apache 2.0
    url: 'http://www.apache.org/licenses/LICENSE-2.0.html'
servers:
  - description: Test Instance
    url: http://localhost:10000
tags:
  - name: inventory
    description: Get physical Entity Inventory
  - name: states
    description: Get Sensor and Input State
  - name: control
    description: Set Output Control
  - name: configuration
    description: Hardware Engine Configuration Interface
paths:
  /api/inventory:
    get:
      tags:
        - inventory
      summary: Get a list of all available entities
      operationId: show_inventory
      responses:
        '200':
          description: successful operation
          content:
            application/json:
              schema:
                  $ref: '#/components/schemas/inventory'
        '400':
          description: Invalid request
        '401':
          description: Unauthorized
      security:
        - ApiKeyAuth: []
  /api/states/sensor/{id}:
    get:
      tags:
        - states
      summary: Get a the value of a sensor entity
      operationId: show_sensor
      parameters:
        - name: id
          in: path
          description: identifier of the entity to return
          required: true
          schema:
            type: string
      responses:
        '200':
          description: successful operation
          content:
            application/json:
              schema:
                  $ref: '#/components/schemas/sensor_entity'
        '400':
          description: Invalid request
        '401':
          description: Unauthorized
        '404':
          description: Entity does not exist
      security:
        - ApiKeyAuth: []
  /api/states/input/{id}:
    get:
      tags:
        - states
      summary: Get a the value of a input entity
      operationId: show_input
      parameters:
        - name: id
          in: path
          description: identifier of the entity to return
          required: true
          schema:
            type: string
      responses:
        '200':
          description: successful operation
          content:
            application/json:
              schema:
                  $ref: '#/components/schemas/input_entity'
        '400':
          description: Invalid request
        '401':
          description: Unauthorized
        '404':
          description: Entity does not exist
      security:
        - ApiKeyAuth: []
```

```yaml
/api/states/output/{id}:
  get:
    tags:
      - states
    summary: Get a the value of a output entity
    operationId: show_output
    parameters:
      - name: id
        in: path
        description: identifier of the entity to return
        required: true
        schema:
          type: string
    responses:
      '200':
        description: successful operation
        content:
          application/json:
            schema:
                $ref: '#/components/schemas/output_entity'
      '400':
        description: Invalid request
      '401':
        description: Unauthorized
      '404':
        description: Entity does not exist
    security:
      - ApiKeyAuth: []
/api/control/output:
  post:
    tags:
      - control
    summary: Set a the value of a output entity
    operationId: set_output
    requestBody:
      description: Output key value pair
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/output_entity'
    responses:
      '200':
        description: successful operation
        content:
          application/json:
            schema:
                $ref: '#/components/schemas/output_entity'
      '400':
        description: Invalid request
      '401':
        description: Unauthorized
      '404':
        description: Entity does not exist
    security:
      - ApiKeyAuth: []
/api/configuration/mqtt:
  post:
    tags:
      - configuration
    summary: Set a the parameters for the MQTT connection
    operationId: set_mqtt_conn
    requestBody:
      description: MQTT Connection Paramters
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/mqtt_connection_parameters'
    responses:
      '200':
        description: successful operation
      '401':
        description: Unauthorized
      '400':
        description: Invalid request
    security:
      - ApiKeyAuth: []
  get:
    tags:
      - configuration
    summary: Get the parameters for the MQTT connection
    operationId: get_mqtt_conn
    responses:
      '200':
        description: successful operation
        content:
          application/json:
            schema:
                $ref: '#/components/schemas/mqtt_connection_parameters'
      '401':
        description: Unauthorized
      '400':
        description: Invalid request
    security:
      - ApiKeyAuth: []
/api/configuration/register:
  post:
    tags:
      - configuration
    summary: Register a new entity
    operationId: register_entity
    requestBody:
      description: Register entity Parameters
```

```yaml
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/register_entity'
      responses:
        '200':
          description: successful operation
        '401':
          description: Unauthorized
        '400':
          description: Invalid request
        '409':
          description: Entity already registered
      security:
        - ApiKeyAuth: []
  /api/configuration/unregister/{id}:
    delete:
      tags:
        - configuration
      summary: Unregister a registered entity
      operationId: unregister_entity
      parameters:
        - name: id
          in: path
          description: Identifier of the entity to unregister
          required: true
          schema:
            type: string
      responses:
        '200':
          description: successful operation
        '401':
          description: Unauthorized
        '400':
          description: Invalid request
        '404':
          description: Entity does not exist
      security:
        - ApiKeyAuth: []

components:
  schemas:
    inventory:
      type: array
      items:
        $ref: "#/components/schemas/inventory_element"
    inventory_element:
      type: object
      properties:
        id:
          type: string
        type:
          type: string
          enum: ["sensor", "input", "output"]
    sensor_entity:
      type: object
      properties:
        id:
          type: string
        value:
          type: number
          format: float
        address:
          type: string
    input_entity:
      type: object
      properties:
        id:
          type: string
        value:
          type: number
          format: boolean
    output_entity:
      type: object
      properties:
        id:
          type: string
        value:
          type: number
          format: boolean
    mqtt_connection_parameters:
      type: object
      properties:
        server:
          type: string
          description: MQTT Server IP address or hostname
        port:
          type: integer
          format: int32
          description: TCP Port Number for MQTT
        username:
          type: string
          description: MQTT username
        password:
          type: string
          description: MQTT password
    register_entity:
      type: object
      properties:
        id:
          type: string
        mqtt_topic:
          type: string
```

```
          description: MQTT Topic of the entity to register
        address:
          type: string
          description: Optional, used for sensors
        type:
          type: string
          enum: ["sensor", "input", "output"]
  securitySchemes:
    ApiKeyAuth:
      type: apiKey
      in: header
      name: X-API-KEY
```

2.4 Service Engine

Description:

2.5 External Services

Description: Local weather data - Get the actual weather data from the area in which the smart home service is installed. (Also displayed in the dashboard)

API: `https://api.met.no/`

2.6 Web dashboard as Graphical User Interface

Description: Interactive elements and history in spreadsheet format is provided in a dashboard for the user as a visual interface for the user api.

User API (REST):

```
openapi: 3.0.0
info:
  description: |
    User API
  version: "1.0.0"
  title: User API
tags:
  - name: sensor
    description: All about sensors
  - name: actor
    description: All about switches
  - name: functions
    description: Operations
  - name: user
    description: Operations about user
paths:
  /sensor/{id}:
    get:
      tags:
        - sensor
      parameters:
        - name: id
          in: path
          description: ID of IO
          required: true
          schema:
            type: string
      responses:
        '200':
          description: successful operation
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Actor'

        '400':
          description: Invalid ID supplied
        '404':
          description: Pet not found

  /actor/{id}:
    get:
      tags:
        - actor
      summary: Get the state current switch sate of the actor
      parameters:
        - name: id
          in: path
          description: ID of Actuator
          required: true
          schema:
            type: string
      responses:
        '200':
          description: successful operation
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Actor'

        '400':
          description: Invalid ID supplied
        '404':
          description: Actuator not found
  /actor/{id}/{state}:
    put:
      tags:
        - actor
      summary: Change the state of actor
      parameters:
        - name: id
          in: path
          description: ID of Actuator
          required: true
          schema:
```

```
          type: string
        - name: state
          in: path
          description: State of Actuator
          required: true
          schema:
            type: boolean
      responses:
        '200':
          description: successfully set state
        '500':
          description: state operation not successful
        '404':
          description: id not found
  /actor/{id}/time:
    put:
      tags:
        - actor
      summary: Set a switch time for an actor
      parameters:
        - name: id
          in: path
          description: ID of Actuator
          required: true
          schema:
            type: string
        - name: time
          in: query
          description: set datime to switch actuator
          required: true
          schema:
            type: integer

      responses:
        '200':
          description: Successfully set timer
        '404':
          description: ID not found


  /user/login:
    get:
      tags:
        - user
      summary: Logs user into the system
      operationId: loginUser
      parameters:
        - name: username
          in: query
          description: The user name for login
          required: true
          schema:
```

```
          type: string
        - name: password
          in: query
          description: The password for login in clear text
          required: true
          schema:
            type: string
      responses:
        '200':
          description: successful operation
          headers:
            X-Rate-Limit:
              description: calls per hour allowed by the user
              schema:
                type: integer
                format: int32
            X-Expires-After:
              description: date in UTC when token expires
              schema:
                type: string
                format: date-time
          content:
            application/json:
              schema:
                type: string
        '400':
          description: Invalid username/password supplied
  /user/logout:
    get:
      tags:
        - user
      summary: Logs out current logged in user session
      operationId: logoutUser
      responses:
        default:
          description: successful operation
components:
  schemas:
    Order:
      type: object
      properties:
        id:
          type: integer
          format: int64
        petId:
          type: integer
          format: int64
        quantity:
          type: integer
          format: int32
        shipDate:
          type: string
```

```yaml
      format: date-time
    status:
      type: string
      description: Order Status
      enum:
        - placed
        - approved
        - delivered
    complete:
      type: boolean
      default: false
  xml:
    name: Order
Category:
  type: object
  properties:
    id:
      type: integer
      format: int64
    name:
      type: string
Data:
  type: object
  properties:
    status:
      type: boolean


User:
  type: object
  properties:
    id:
      type: integer
      format: int64
    usernam:
      type: string
    lastName:
      type: string
    email:
      type: string
    password:
      type: string
    phone:
      type: string
    userStatus:
      type: integer
      format: int32
      description: User Status
  xml:
    name: User
Tag:
  type: object
```

```
    properties:
      id:
        type: integer
        format: int64
      name:
        type: string
    xml:
      name: Tag
  Actor:
    type: object
    required:
      - id
    properties:
      id:
        type: string
      status:
        type: boolean
        servers:
# Added by API Auto Mocking Plugin
- description: SwaggerHub API Auto Mocking
  url: https://virtserver.swaggerhub.com/Bennys-Quarter/UserAPI/1.0.0
```