Web Interface – Home Automation

To Control to REAL Devices VIRTUALLY

Software Design Document

Table of Contents

[1. Project Description 3](#_Toc137399243)

[2. System Entities 3](#_Toc137399244)

[3. System Requirements 4](#_Toc137399245)

[4. Out of scope 4](#_Toc137399246)

[5. Assumptions 4](#_Toc137399247)

[6. Proposed Solution 5](#_Toc137399248)

[7. Tech Stack 6](#_Toc137399249)

[8. Sequence Diagrams 7](#_Toc137399250)

[9. Scope of enhancement 7](#_Toc137399251)

[10. Key Screens 8](#_Toc137399252)

# 1. Project Description

Home automation is a web-based system that collects all home appliances and devices that can connect to the internet and be controlled remotely. Here we are having a simple remote which transmits information back and forth, for operating the devices.

This project assumes remote to be configurable screen (like tablet or PC) which can be designed to accommodate different ranges of slots, depending on the size of home. Each slot may be configured with “device transmitter and receiver” which are needed to control the device.

The “device transmitter and receiver” could be added as such too without being assigned to any slot. However to make it operational, it should be assigned to any slot, The devices can be fitted into any slot and the slot similarlycan hold any one device (as per the requirement).

In the document and code, a “slot” is mentioned as “slot” but “device transmitter and receiver” is mentioned as “device”.

# 2. System Entities

There are 2 key entities – 1) Device and 2) Slot

**DEVICE**

a) A device has the following attributes: **ID, Name, Type, Status and information**

b) A device is located in a slot

c) Each device can have a single type (On-Off type / Rotary switch type etc). Currently we are supporting only On-Off type. However the “type” option gives provides extensibility

d) A device always has some Status: On/Off

e) Information: Will represent information for a device.

|  |
| --- |
| When a device is added to slot, it’s considered as “Registered”  When a device is not added to slot, it’s considered as “Unregistered” |

**SLOT**

a) A slot has the following attributes: **ID, Name, and Type**

b) A slot has a name for easier identification

c) Each device can have a single type (small / medium/ large etc). Currently we are supporting only “large” type. However the “type” option gives provides extensibility

# 3. System Requirements

a) Come up with POC having the API used by a remote to control the Home Hub as such it is sufficient for the Home Hub to simulate turning devices on/off by writing to a console or log file.

b) The remote has several slots with two buttons per slot. Customers can assign an arbitrarily supported device to any slot

c) Supported devices can be assigned to any slot and turned on or off using one of the two buttons assigned to a slot

d) The system should support the ability to undo previous actions, as such the remote also features a single “undo” button which will undo the previous user action.

# 4. Out of scope

a) There is no requirement to implement any security mechanism.

b) There is no requirement to implement any multi-tenancy. Assume this is a single-user system.

c) There is no requirement for a database. If you need persistence do whatever is simplest for example in memory or a flat file.

d) There is no requirement to implement any UI or Client.

# 5. Assumptions

1) We are allowing the users to specify the deviceIds and slotIds (as they need not be in sequence)

2) When users try to re-create new slot / device, the pre-existing data / state would be over-written. So user should check the pre-existing state before making the changes.

They can also make use of the "Undo" feature to revert back if user happens to make changes, by mistake

3) Undo operation is restricted to operations on the device (as only those user actions are being captured). Adding a device or slot won't be undone. Only ON / OFF actions on the device can be undone.

Note:: The very first operation can't be undone (as there won't be any previous actions stored

4) The subsequent undo will go proceed from the restored state of previous undo.

Eg. If 10 operations are performed and undo of 3 is given, the state will be in 7th operation. The cache size will also be reduced by 3 (to 7)

So if we give another undo of 4, it will lead us to 3rd operation (and not 6th operation)

The cache size will be further reduced by 4 (to 3)

5) The status of ON or OFF is taken from the value specified in URLs PathVariable (and will be overwritten even if the body has different value, by mistake)

6) It's assumed that device can exist as such. However it can be operated only when associated to a slot.

The details specified in the body get precedence over the deviceIds specified in path (if any mismatch happens), meaning all the operations would happen to the deviceId specified in the body of the message.

# 6. Proposed Solution

The “Home Page” would comprises of 3 controllers

a) Slot Controller: SlotController is used to configure the slots remotely

b) Device Controller: DeviceController is used to configure the IOT devices remotely

c) Iot Controller: IotController is used to operate the IOT devices remotely

**A. Controllers**:

Controller methods are the final destination point that a web request can reach. After being invoked, the controller method starts to process the web request by interacting with the service layer to complete the work that needs to be done.

Each controller flow would be used to handle a specific functionality.

Although the controllers handle the requests, they won’t have any business logic associated. This is done to make it light-weight and more focused on the routing functionalities

**B. Services**:

The service layer in an application that facilitates communication between a Controller and the persistence layer, Repository. It manages the business logic, for each of the flows.

**C. Repositories**:

Repositories are classes or components that encapsulate the logic required to access data sources. Repository layer is added between the domain and data mapping layers to isolate domain objects from details of the database access code and to minimize scattering and duplication of query code. The Repository pattern is especially useful in systems where number of domain classes is large or heavy querying is utilized.

**D. Domain Objects and DTOs**:

DTOs and Domain Objects can be designed to use the same class, even represented by the same object for the same entity.

Domain model pattern provides an object-oriented way of dealing with complicated logic wherein a complex domain object is constructed using multiple simple domain objects.

**E. Caching mechanism**:

As it’s a POC, we are not using any database and using in-memory caching of data. As we may need 2 tables one for Device and another one for slot, we are having 2 hashmaps, in this POC.

The device is associated to a slot and any slot at any instance of time can hold only one device. So they have one-to-one relationship wherein their ids can be used as primary key in the corresponding table and as foreign key, in the other table.

**F. Undo mechanism**:

This is done using “ArrayDeque”. All the operations done via “Iot Controller” are stored in ArrayDeque whose capacity is configurable (currently kept as 10)

The key reason behind going with ArrayDeque instead of stack is

a) ArrayDeque has better performance than stack

b) Deque has inbuilt mechanism to handle the full and empty scenarios while stack throws exception, in both the cases

c) ArrayDeque can also be used to carry out “Redo” operations if in case, some requirement comes in future.

|  |
| --- |
| Code snippet |
| ArrayDeque<ExecutionStateRecorder> temp = new ArrayDeque<>(); temp.addLast(ExecutionStateRecorder.*operationsRegister*.pollLast());  for(long i=0; i<undoCount; i++) {  ExecutionStateRecorder deviceState = ExecutionStateRecorder.*operationsRegister*.getLast();  if(i+1 == undoCount) {  oldStateDeviceDTO = deviceState.getDevice();  }  // Removing the data of previous state as they are no longer needed  temp.addFirst(ExecutionStateRecorder.*operationsRegister*.pollLast()); } temp.stream().forEach( x -> ExecutionStateRecorder.*operationsRegister*.add(x)); |

This ArrayDeque currently stores “device” data which gets recorded into “ExecutionStateRecorder”. This “ExecutionStateRecorder” can be extended to capture other data too like the details of slots etc, as and when the requirement changes.

The current code supports 3 types of UNDO operations

1) Single Undo: Used to go back to previous state

2) Multiple Undo:

3) Multilevel Undo:

# 7. Tech Stack

1. Spring Boot

2. Java 11

3. Apache maven for building the project

4. Swagger for documentation

5. REST API operations

6. Logging

7. Jakarta for request validation

8. Actuator for application monitoring

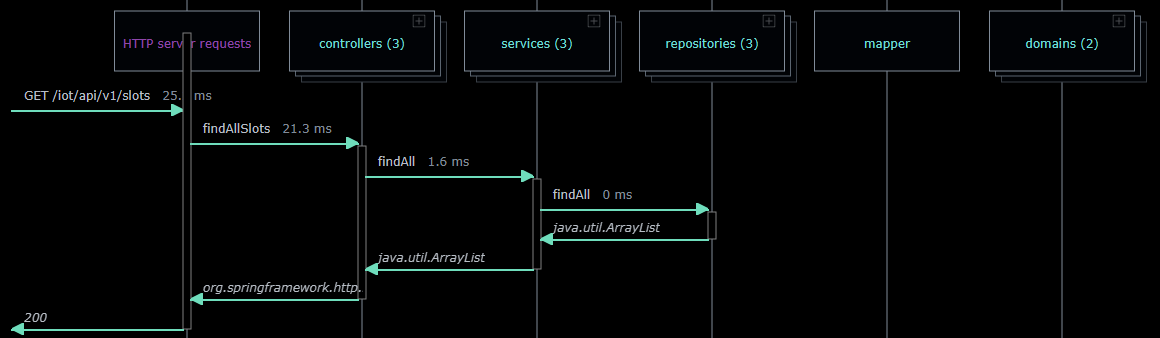
9. Testing using Postman, CURL and RESTTemplate

10. Junit and Mockito for unit tests

11. Jacoco for test coverage

# 8. Sequence Diagrams

Please find the sequence diagram for “Getting all the slots”



For other sequence diagrams, refer the below link

<https://github.com/deepak2all/HomeAutomation/blob/master/documents/Sequence%20Diagrams.svg>

# 9. Scope of enhancement

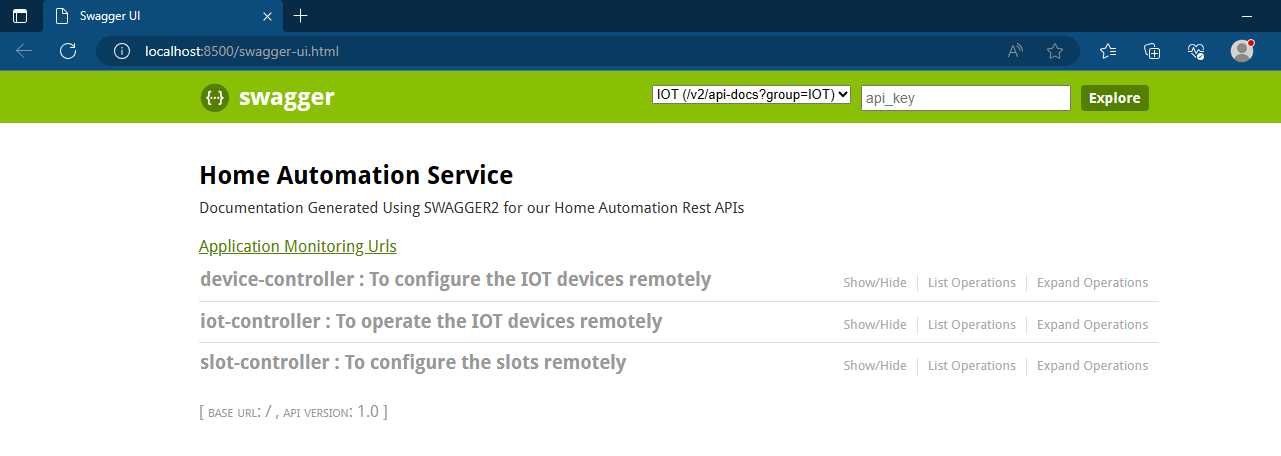
The design of the system is catered for enhancements in future.

The details of the same are shared in the ppt

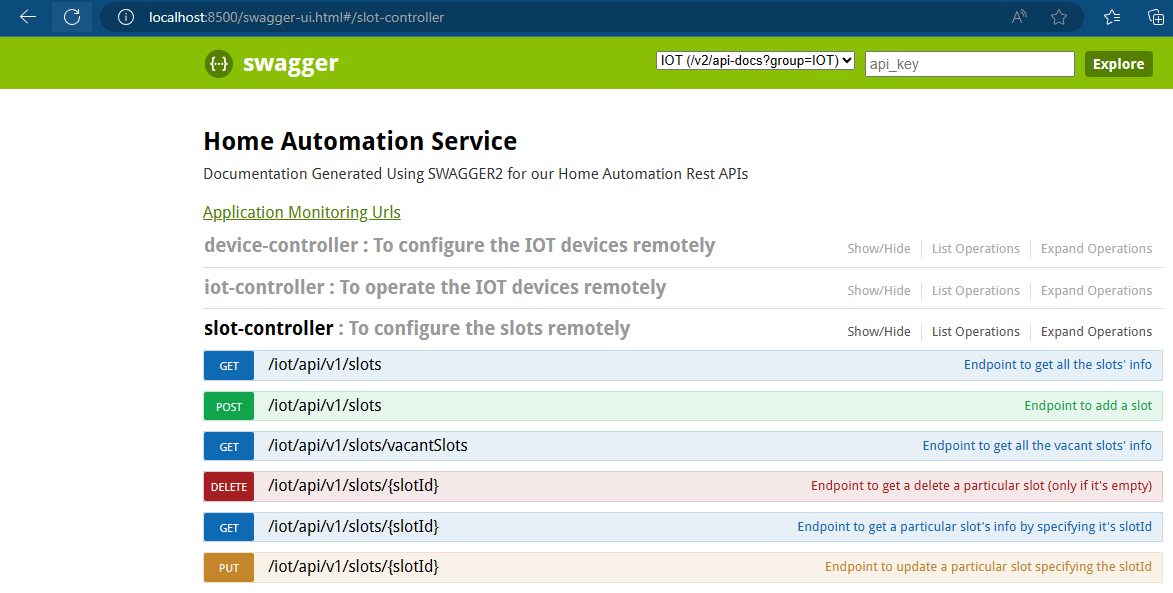
Moreover, as with this use case, we need a user to operate a device. This can be further integrated with AI tools which can automate this operation depending on some conditions. Eg. Automatically turn-on / turn-off the lights / heater depending on the weather.

# 10. Key Screens

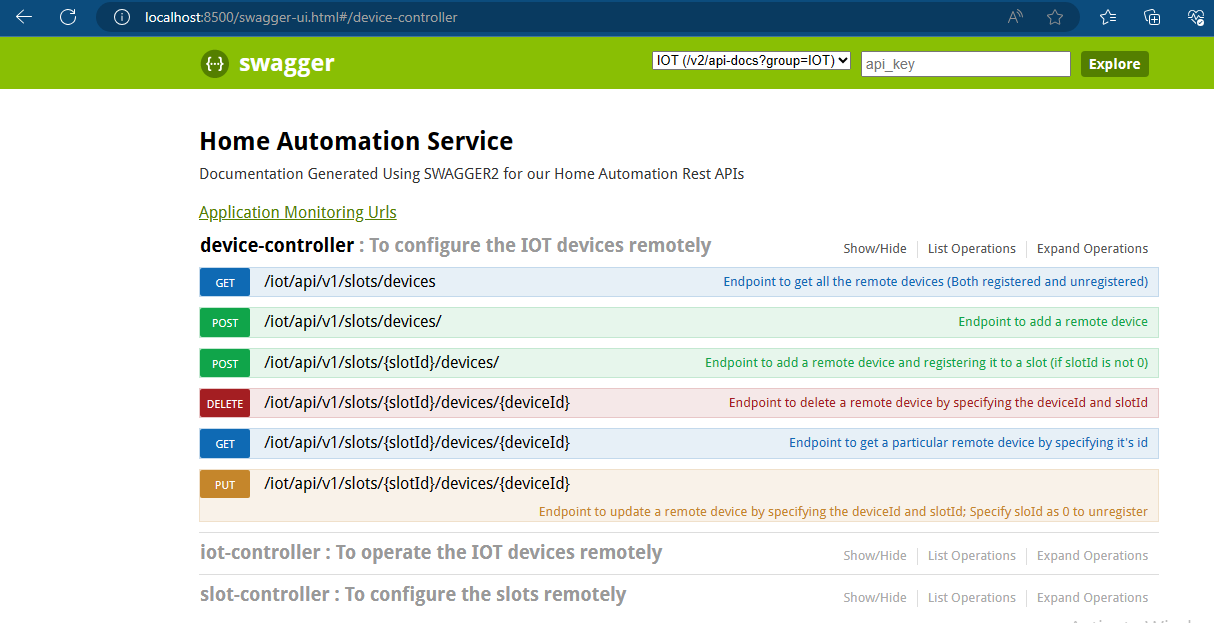
Home Screen



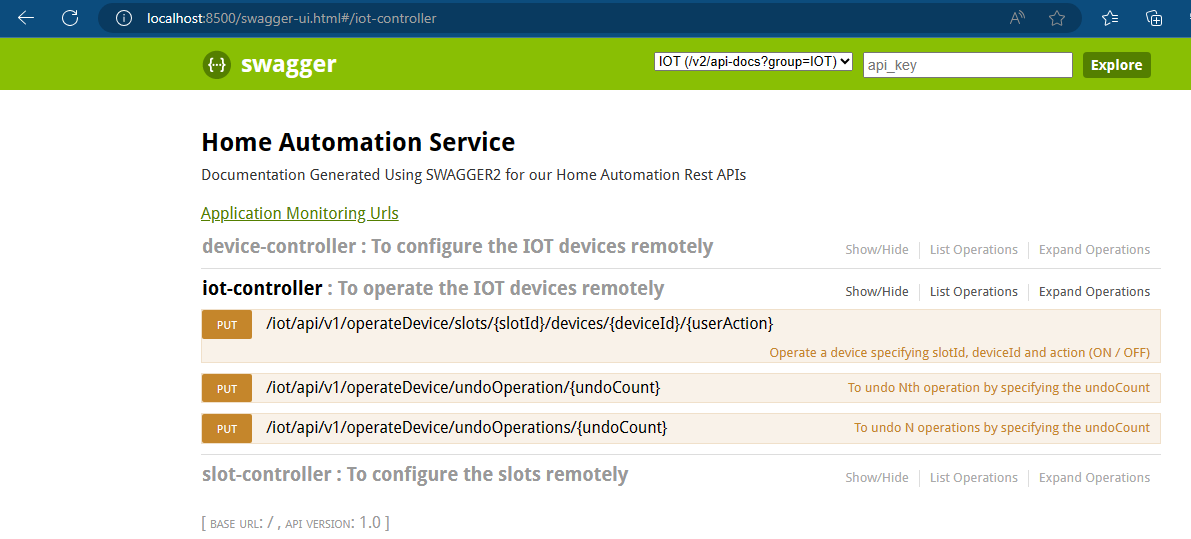
Endpoints of slot-controller



Endpoints of device-controller



Endpoints of iot-controller



Info page

