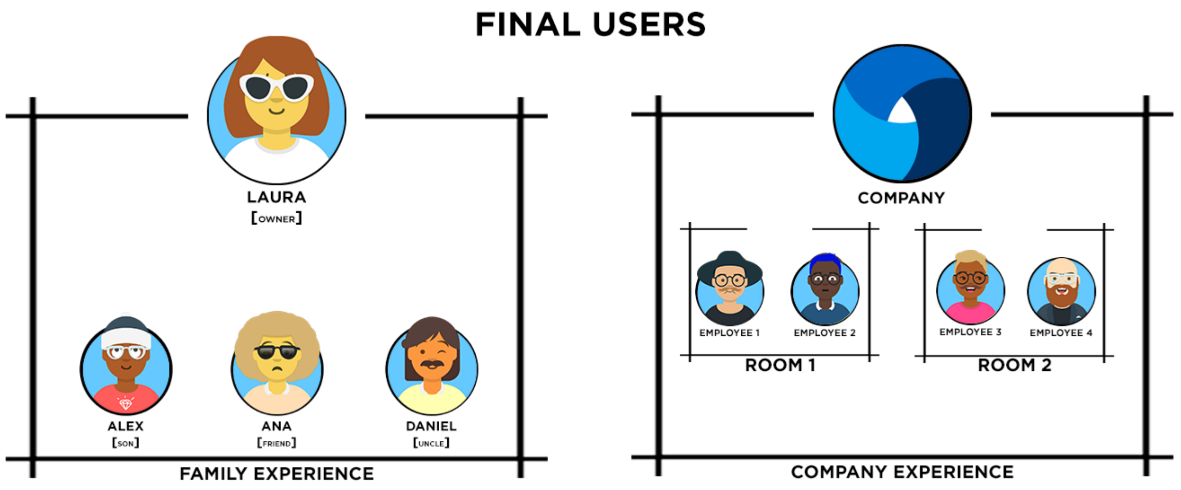
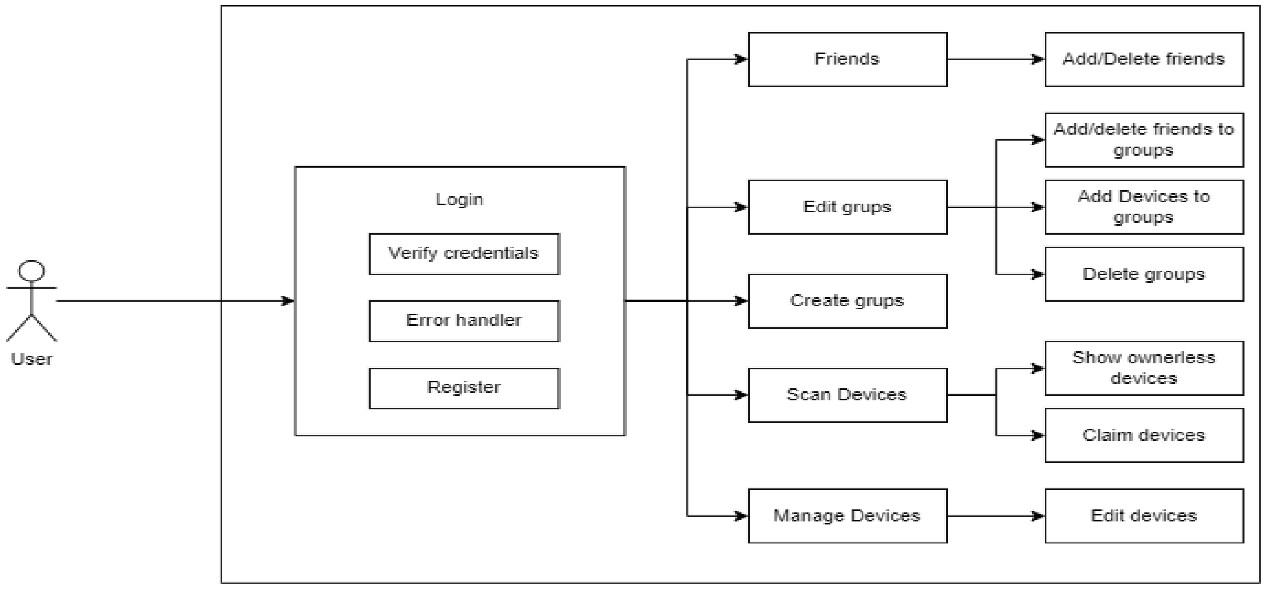
smarthome

1. Introduction

The SmartHome application was created with the purpose of helping the smart devices owners manage and control them in an easy and user-friendly way. The owner can register his own devices (smart bulbs, smart locks, thermostats) and then monitor them and send commands. (ex. change the brightness of the bulb or set a different room temperature) Another feature is to allow the owners to form groups of other users and grant them rights to control their devices.



Picture 1. Users



Picture 2. Use-case diagram

2. Linked Data Principles

The principles are:

- Use URIs as names for things.
- Use HTTP URIs so that people can look up those names.
- When someone looks up a URI, provide useful information.
- Include links to other URIs. so that they can discover more things.

The application is following the linked data principles by using unique guids to identify each device and each user. Furthermore, for each type of device the DeviceSimulator Api is used to provide endpoints where more information about the device can be found. When a device is queried, one also receives data about the owner making it easy to discover more things and keeping the data linked. The entire data regarding one user can be uncovered by traversing the knowledge graph with the user node as a starting point. For example, we could discover if we can get one's devices by listing all the objects that the user has a relation of type "owns" with or we can check if a user has control over a device that he does not owns by going to all the groups that the user has an "inGroup" relation with and checking that the group owner matches the device owner, and then checking the group rights.

3. Technologies utilized

3.1. .NET Core

A well-known cross-platform framework, frequently used with the C# programming language. It allows for the development of a variety of applications, such as mobile, desktop, **web**, cloud, **IoT**, machine learning, **microservices**, games, etc.

3.2. Blazor

Blazor is a relatively new technology that is part of ASP.NET, it allows the developer to build interactive web UIs using C# instead of Javascript. It has a client side version and a server side version. The server side version will be used for this implementation to fully benefit from all the features of the .NET ecosystem, some of those not being accessible in the client side version. The UI components are built on the server and then sent to the client via SignalR. The communication via SignalR makes real-time UI updates possible.

3.3. Stardog

A knowledge graph platform that enables clients to unify all their data, including data sources and databases of every type. It provides high availability, high performance reasoning, and virtualization. Among the supported programming languages are Java, Python, Javascript and C#. Using virtual graphs, Stardog can aggregate multiple data sources into one or more knowledge graphs.

3.4. Microsoft Azure

A cloud computing service provided and operated by Microsoft. It is among the largest and most used cloud services right now, allowing the integration of both Microsoft specific and third party tools and systems. It can be used with various degrees of customer input, providing the well known IaaS, PaaS and SaaS models among many other paradigms. More specifically, the Azure services used in the implementation are as follows:

- Azure AD B2C, a customer identity management solution, taking care of the scaling and security of the whole process. It can be used with social, enterprise or local accounts.
- More services (for example Key Vault) might be integrated as the implementation progresses..

3.5. Flask & Tkinter

These two services were used to have a good management of the devices and their simulation. Together they form a Server API that handles requests from the web side. On the backend side we also created the logic part for each device.

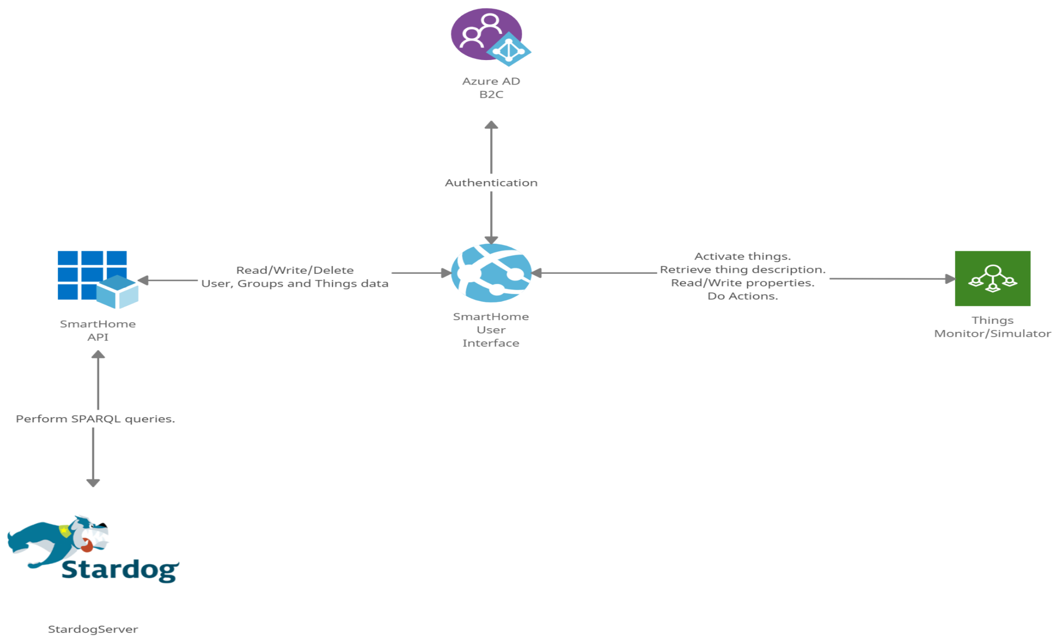
4. Architecture

The application's architecture is a modular one, employing multiple services in the form of APIs for data retrieval and data manipulation and cloud based services for other aspects like security. Also, a service that acts as a bridge between the devices and the other components will be used, it will provide live updates based on the data provided by various devices. The main communication protocol used is HTTP(s).

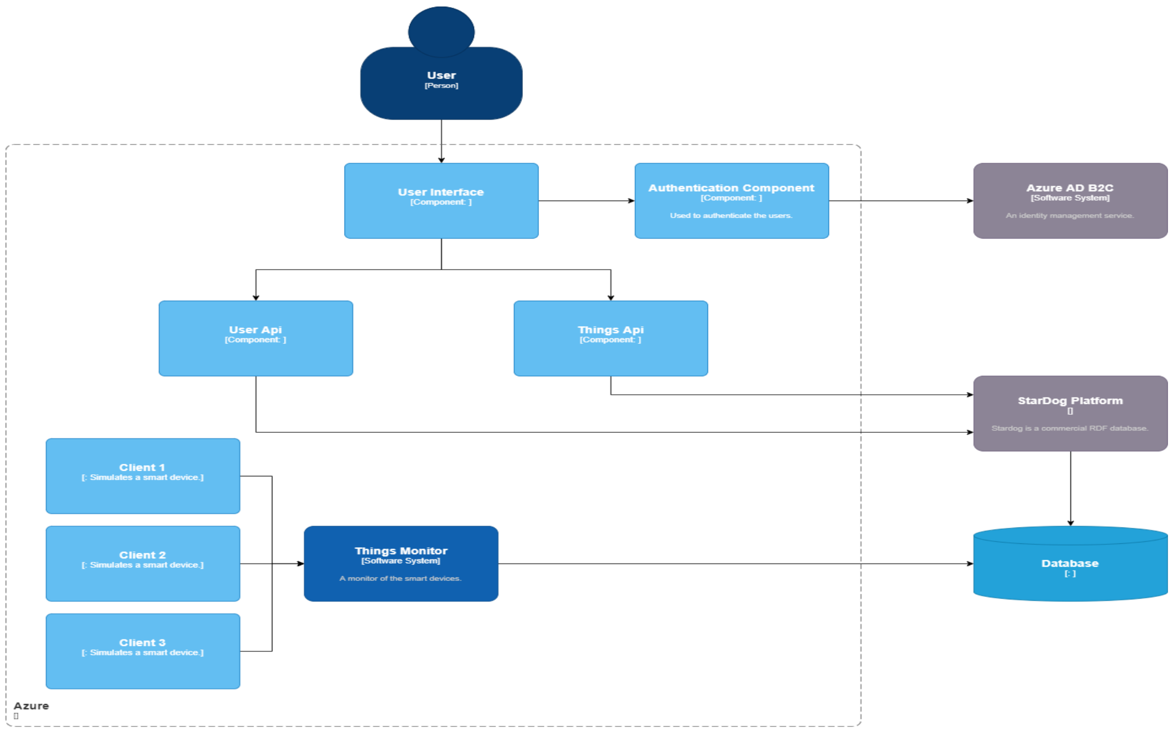
.

- **Smart Home User Interface**, a visual interface that will be presented to the end user, allowing him to query and alter certain sections of the knowledge graph. This component will implement user authentication via Azure AD B2c. An authenticated user will be able to view and alter the devices that he owns and the devices that he has access to via group claims. The user can also manage the groups that he owns (family, friends, associates, etc) by adding or removing other users or claims.
- **SmartHome API**, a service that performs CRUD operations on the user, group and thing data. It exposes endpoints related to the users,groups,group claims and things. Once a thing is activated/claimed by a user, it will be stored locally by performing a request to thi component.
- **Stardog** will be used to persist the user, group and thing data. It will also provide the possibility to query, and alter data via SPARQL.
- **Things Monitor/Simulator**,a Python application that will act as both a bridge of communication between the SmartHome application and a simulator for things. It provides the available things that can be activated and then provides the thing description in JSON format if the activation key is correct. After that, it will simulate the behavior of various devices offering support for reading properties, writing properties and doing actions.

The pictures below present a general view of the application's architecture and it's containerization (including the user input entry point).



Picture 3. SmartHome Architecture diagram



Picture 4. SmartHome C4-Container diagram

5. Data Schema

The data will be stored into a stardog database hosted on a stardog server. Every operation (reading, inserting and deleting data) will be performed using SPARQL. The entities are modeled as follows:

1. User

   ?user a foaf:Person

   foaf:familyName ?familyName;
   foaf:givenName ?givenName;
   foaf:openid ?openid;
   foaf:mbox ?mbox;
   foaf:accountName ?accountName

2. Group

   ?group a foaf:Group

   foaf:name ?name;
   foaf:openid ?openid;
   foaf:maker ?owner

3. Thing

   ?thing a td:Thing;

   td:title ?title;
   td:description ?description;
   td:id ?id

4. Thing Property

```
?property a td:PropertyAffordance;
        td:title ?title;
        td:description ?description;
        hctl:hasOperationType ?type;
        jsonschema:readOnly ?readOnly.
        OPTIONAL{?property jsonschema:minimum ?minimum;
                           jsonschema:maximum ?maximum. }.
```

5. Thing Action

```
?action a td:ActionAffordance;
        td:title ?title;
        td:description ?description;
        hctl:hasOperationType ?type
```

6. Action Parameter

```
 ?parameter td:name ?name;
        hctl:hasOperationType ?type.
        OPTIONAL{?parameter jsonschema:minimum ?minimum;
                            jsonschema:maximum ?maximum.}
```

The relations between these entities are modeled as follows:

- User2 is a friend of User1: *?user1 foaf:knows ?user2*
- User is a member of a group: *?user foaf:member ?group*
- User owns thing: ?thing *fibo:owner ?owner*
- Group has access to thing: ?group *fibo:topic_interest ?thing*
- Thing has property: *?thing td:hasPropertyAffordance ?property*
- Thing has action: *?thing td:hasActionAffordance ?action*
- Action has parameter: *?action td:uriVariables ?parameter*

Obs:A thing represented in JSON format can be found in the Appendix section.

6. Conclusion

Nowadays, technology advances very fast, new things always appear, and are more and more sophisticated.

An application like SmartHome is welcome in our lives, it makes life easier for users, either at home or in companies.SmartHome makes managing all of the smart devices a lot easier by unifying the access to their controls and their status into a single central point, taking care of aggregating their data and their specific communication protocols.

7. Team component

- Chirica Demetra Bianca
- Miron Robert Andrei
- Susan Stefan Claudiu

8. Appendix

Thing in JSON format (as received from the Thing Monitor/Simulator):

```
{
IRI:
"http://127.0.0.1:5010/Thermostat-f80ceba6-80ea-4c21-8e4a-e4
074c9f2c39",
access_code: "224563",
actions: [
{
IRI:
"http://127.0.0.1:5010/Thermostat-f80ceba6-80ea-4c21-8e4a-e4
074c9f2c39/turnOnHeating",
description: "Turn on heating for a number of minutes",
title: "turnOnHeating",
type: "void",
uriVariables: [
{
maximum: "180",
minimum: "15",
name: "duration",
type: "int"
}
]
},
{
IRI:
"http://127.0.0.1:5010/Thermostat-f80ceba6-80ea-4c21-8e4a-e4
074c9f2c39/suggestTemperature",
description: "Suggests the ideal temperature based on the
outside temperature.",
title: "suggestTemperature",
type: "int",
uriVariables: [ ]
}
],
connected: "False",
description: "An intelligent thermostat",
id: "Thermostat-f80ceba6-80ea-4c21-8e4a-e4074c9f2c39",
properties: [
{
IRI:
"http://127.0.0.1:5010/Thermostat-f80ceba6-80ea-4c21-8e4a-e4
074c9f2c39/currentTemperature",
description: "The current temperature in the room.",
maximum: "100",
minimum: "0",
readOnly: "true",
title: "currentTemperature",
type: "int",
value: "0"
},
{
IRI:
"http://127.0.0.1:5010/Thermostat-f80ceba6-80ea-4c21-8e4a-e4
074c9f2c39/minimumTemperature",
description: "The minimum temperature required for the
heating to start.",
maximum: "100",
minimum: "0",
readOnly: "false",
title: "minimumTemperature",
type: "int",
value: "0"
},
{
IRI:
"http://127.0.0.1:5010/Thermostat-f80ceba6-80ea-4c21-8e4a-e4
074c9f2c39/status",
description: "Turn the thermostat On/Off.",
maximum: "1",
minimum: "0",
readOnly: "false",
title: "status",
type: "boolean",
value: "false"
}
],
title: "Thermostat"
}
}
```