U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

# Information Systems – SINF
# 2019 /2020

# LAB GUIDE – SPRINT3

## Graphical User Interface

Gil Gonçalves – gil@fe.up.pt
Luis Neto – lcneto@fe.up.pt
João Reis – jpcreis@fe.up.pt
Vítor Pinto – vitorpinto@fe.up.pt

## 1. Introduction

The current project consists on the development of an information system for home automation, which includes:

1. A multi-threaded software application for sensing/actuation using a Raspberry Pi 3;
2. A relational database in PostgreSQL for persistent storage of information;
3. A web interface, in order to present the information and accept commands of an end-user.

Topics 1 and 2 were already covered in Sprint 1 and Sprint 2. In this Sprint, the goal is to tackle topic 3, which will introduce to students know-how regarding Graphic User Interfaces (GUI), specially Web interfaces. Additionally, students will learn how to perform complex actions over the database using SQL commands in a web interface.

In Sprint 3, students are supposed to work on the GUI for the HAS. The web interface for the HAS is already available on (*https://github.com/SINF-FEUP/HAS-Web-Interface* ) along with the guidelines for its deployment. Students must install the provided web interface and design the SQL queries that provide the required results.

Note that in case you have failed to implement the database for your HAS system in Sprint 2, a sample database will be available soon in the GitHub repository to allow the successful completion of this sprint.

This document focuses on the following aspects:

2. Web Interfaces
3. HAS Web Interface
    a. Pre-Requirements
    b. Characterization
4. Application Questions
5. Application Questions Results
    a. Easy
    b. Medium
    c. Hard
8. Practical Examples
9. Sprint 3 Main Goals

## 2. Web Interfaces

A Graphical User Interface, or GUI, is a type of user interface that allows users to interact with using electronic devices through graphical icons and visual indicators such as secondary notation, instead of text-based user interfaces, typed command labels or text navigation. A web interface is a GUI that consists on the interaction between a user and software, which is running on a web server. In this case, the user interface is the web browser and the web page is downloaded and rendered, i.e., a user can use a web browser to access and view/edit the content of a web page.

Typically, a web page consists in a group of web resources, primarily text documents formatted and annotated with Hypertext Markup Language (HTML). Web pages may also contain images, video, audio and other software components that are rendered in the user's web browser as coherent pages of multimedia content.

A web page must be hosted in a server that can be accessed locally or remotely, using, in this case, the World Wide Web (WWW). The World Wide Web allows to identify web resources by Uniform Resource Locators (URL), interlinked by hypertext links, and accessible via the Internet. Web resources are usually accessed using the Hypertext Transfer Protocol (HTTP), which is one of many Internet communication protocols.

Viewing a web page on the World Wide Web normally begins either by typing the URL of the resource to be accessed into a web browser, or by following a hyperlink to that resource. The web browser then initiates a series of background communication messages to fetch and display the requested resource, such as represented in Figure 1.

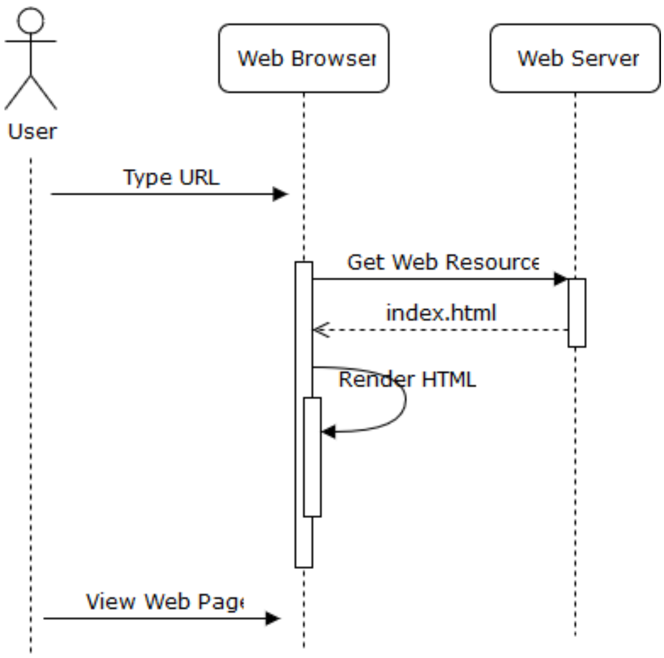



*Figure 1 - Access a Web resource.*

The previous example demonstrates the functioning of a web browser when accessing a web resource at a user introduced URL. The browser resolves the server name of the URL into an Internet Protocol address, using the globally distributed Domain Name System (DNS). This lookup returns an IP address. The browser then requests the resource by sending an HTTP request across the Internet to the server at that address. It requests service from a specific TCP port number that is well known for the HTTP service, so that the receiving host can distinguish an HTTP request from other network protocols it may be servicing. The server receiving the HTTP request delivers it to web server software listening for requests on port 80. If the web server can fulfil the request it sends an HTTP response back to the browser indicating success, followed by the content of the requested page, which is specified using HTML.

The web browser parses the HTML and interprets its markup that surrounds the words to format the text on the screen. Many web pages use HTML to reference the URLs of other resources such as images, other embedded media, scripts that affect page behaviour, such as PHP and JavaScript, and stylesheets, such as Cascading Style Sheets (CSS), that affect page layout. The browser makes additional HTTP requests to the web server for these other Internet media types. As it receives their content from the web server, the browser progressively renders the page onto the screen as specified by its HTML and these additional resources.

An HTML for a basic web page might look like the one represented in Figure 2.

```html
<html>

<head>
    <title>My First Web Page</title>
</head>

<body>
    <h1>My First Web Page</h1>
    <p><b>Hello World Wide Web!</b></p>
    <p><i>Hello World Wide Web!</i></p>
    <p><u>Hello World Wide Web!</u></p>
    <p>This is my first web page.</p>
    <p>HTML tags can give <b><i>various</i></b>
    <u>looks and format</u> to the content of this web page.</p>
</body>

</html>
```

*Figure 2 - Basic HTML File.*

Figure 3 represents this HTML file rendered in a Web browser, if it is hosted in a Web server, to be accessed by users.

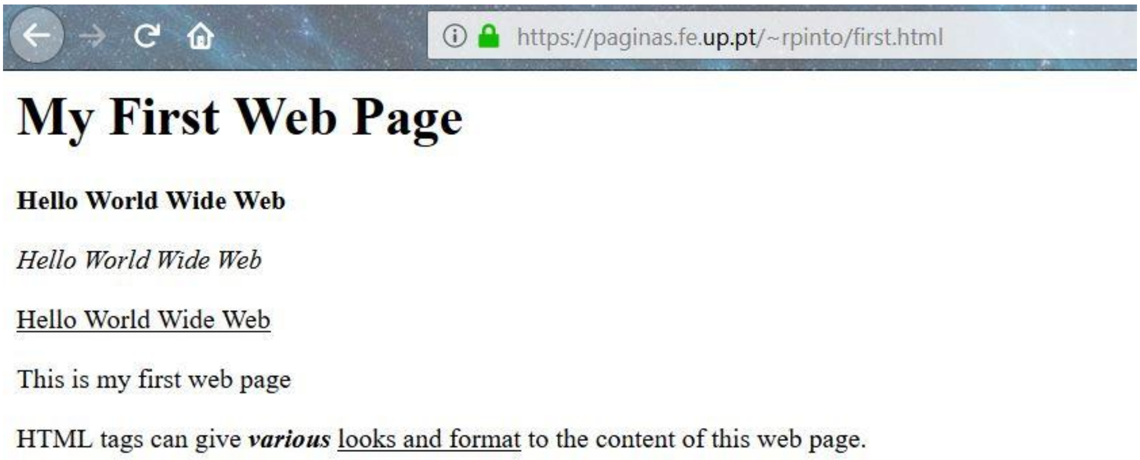



*Figure 3 - Basic HTML File Rendered in Web Browser.*

## 3. HAS Web Interface

In order to make available to users the information managed in the HAS, a website should be implemented. In this case, the website is a collection of related web pages, including multimedia content, identified with a common domain name, and published on at least one web server. The website should be accessible via a public Internet Protocol (IP) network, such as the Internet, or a private local area network (LAN), by referencing an URL that identifies it.

Websites can have many functions and can be used in various fashions. In this case, the website will allow users to access information that primarily exists in the database implemented in Sprint 2. Students already know the *phppgadmin* web interface, from Sprint 2, which is a similar website that allows viewing and editing information from a PostgreSQL database, by accessing the URL *https://db.fe.up.pt/phppgadmin/*.

The web pages of the HAS website are documents, typically composed in plain text interspersed with formatting instructions of HTML. The user may use a web browser, such as Internet Explorer, Mozilla Firefox, Google Chrome, Safari, Opera, Chromium or others, to render the page's content according to its HTML markup instructions onto a display terminal, which can consist in a range of devices, including desktop and laptop computers, tablet computers, smartphones and smart TVs.

In this case, and since Sprint 3 is very short, a simple generic website will be provided to every team in order to speed up the development of the web interface. This website provides an interface to introduce SQL queries, in order to access the implemented database and present, in a graphical and tabular form, the information managed in the HAS.

The referred website consists on web pages stored on a server in the format that is sent to a user web browser. It is primarily coded in HTML, CSS to control appearance beyond basic HTML, PHP and JavaScript, to make the web pages dynamic and instruct the web browser how to interactively modify the web pages contents.

**Students are not required to make any changes or to fully understand how this web interface is implemented! Students should focus on the development of SQL queries to be used in the system, in order to return results according to several applications questions. Optionally, students may change this web interface at their own risk.**

### a. Pre-Requirements

Users may access the provided website locally or remotely. Accessing remotely the provided web pages imply placing them in an existing web server that is connected to the Internet. Web hosting services are not free and require a payment every month / year. Luckily, FEUP makes available to its students and personal a free web hosting service. In this case, students may use FEUP's server infrastructure as the web server that will enable the access to the website. The server to be used is located at *samba.fe.up.pt*, which have, for each student, a reserved space to place the web pages.

In order to access *samba.fe.up.pt*, one must first be connected to the Internet with an active VPN connection to FEUP's internal network. Sometimes, even within EDUROAM, students must have an active VPN connection, despite already being connected to EDUROAM. For this reason, is best to open a VPN connection to FEUP's internal network every time you want to access FEUP's Web hosting services, such as the *samba.fe.up.pt* server.

1) Configure VPN connection

In https://sigarra.up.pt/feup/pt/web_base.gera_pagina?p_pagina=p%c3%a1gina%20est%c3%a1tica%20gen%c3%a9rica%2063, you can consult several methods to use the FEUP's VPN service. As an example for Win10, go to *Network Management -> VPN -> Add new VPN connection*. Now input the requested data, as presented in Figure 4 (you should use its own username and password).

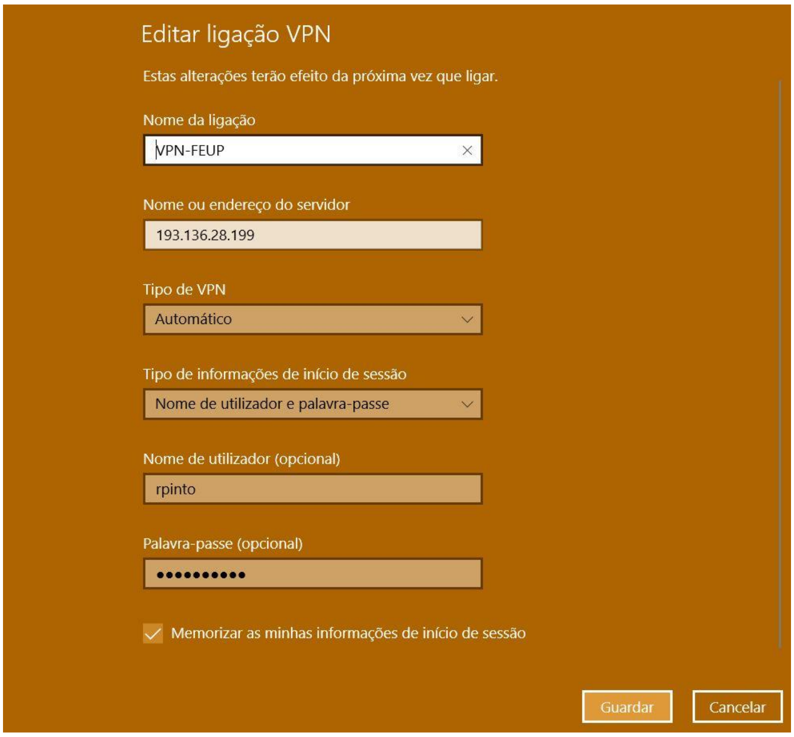



*Figure 4 - FEUP's VPN Configuration.*

2) Access to samba server

In https://sigarra.up.pt/feup/en/web_base.gera_pagina?p_pagina=p%c3%a1gina%20est%c3%a1tica%20gen%c3%a9rica%201140, you can consult how to use the FEUP's Web page hosting service. As an example for Win10, execute the shortcut '*Win + R*' to open the execute console and insert '*samba.fe.up.pt*', as represented in Figure 5.

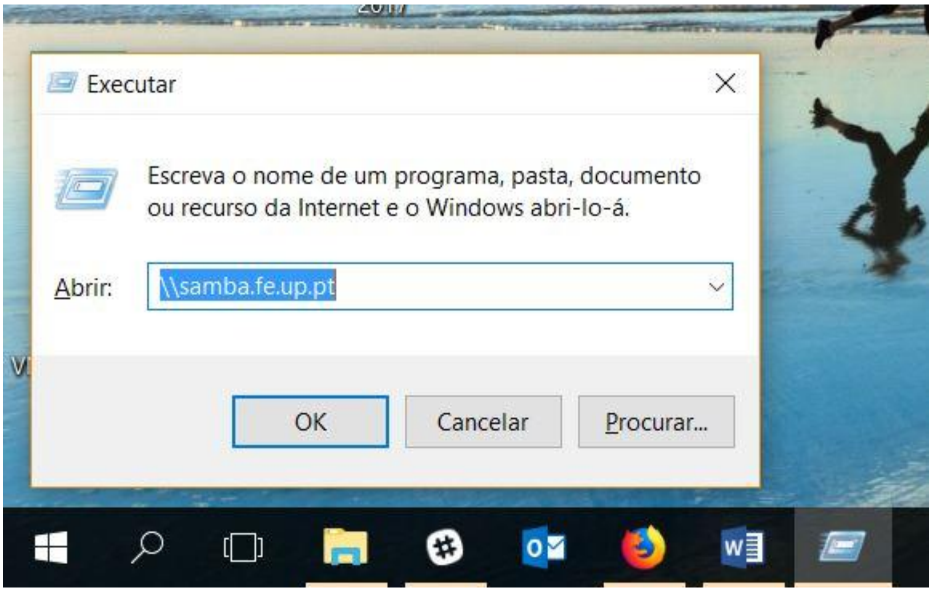



Figure 5 - Access samba Server.

After opening a connection to samba server, you must introduce the requested credentials to get access to the server contents. In this case, you must choose '*Use a different account*' and insert the username and password. Note that the username must be preceded by '*feupsig*\', as represented in Figure 6.

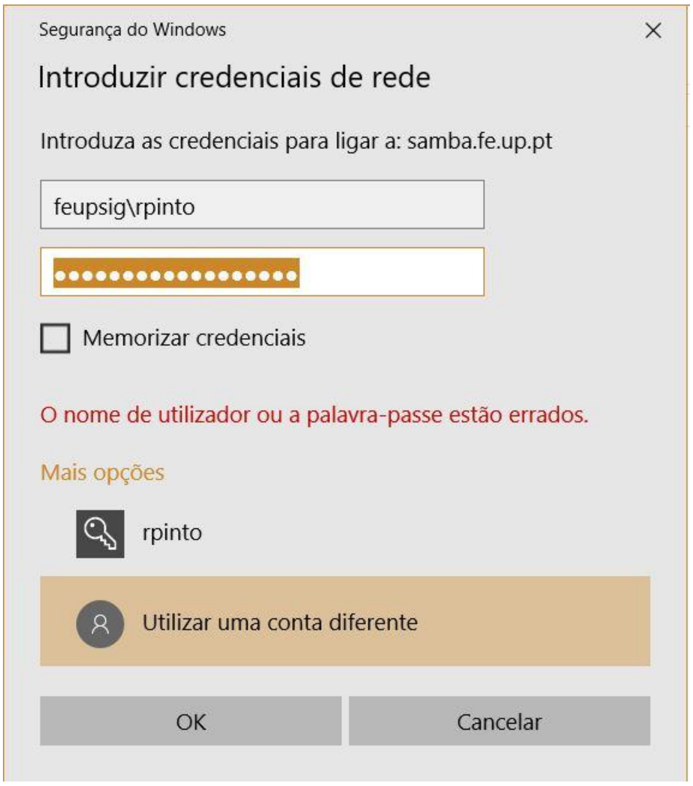



Figure 6 - Connection to samba server.

3) Place website in server

In this stage, there is an open connection to the samba server, and here the user can manage its contents. In this case, it is intended to manage the files that exist in the directory *public_html*. One can notice that inside the *public_html* file, there is already a default web page, in the format of an HTML file, named *index.html*. In that sense, *public_html* should be used to place the HAS website provided. This can be easily accomplished by copy and paste the HAS website, i.e., the entire downloaded folder from GitHub, inside the directory *public_html*.

4) Browse Website

After the website is placed in the web server, you can access it using your favourite web browser. In order to access the default web page that is placed in the *public_html,* open the web browser and enter the URL '*https://paginas.fe.up.pt/~upxxx*'. Note that '*upxxx*' should be replaced by your own username, as represented in Figure 7. In the same way, you can access to the provided HAS website, which was previously placed in the *public_html,* by opening the web browser and enter the URL '*https://paginas.fe.up.pt/~upxxx/sinf*'. Again, note that '*upxxx*' should be replaced by your own username and '*sinf*' is the example name of the folder placed inside *public_html*.

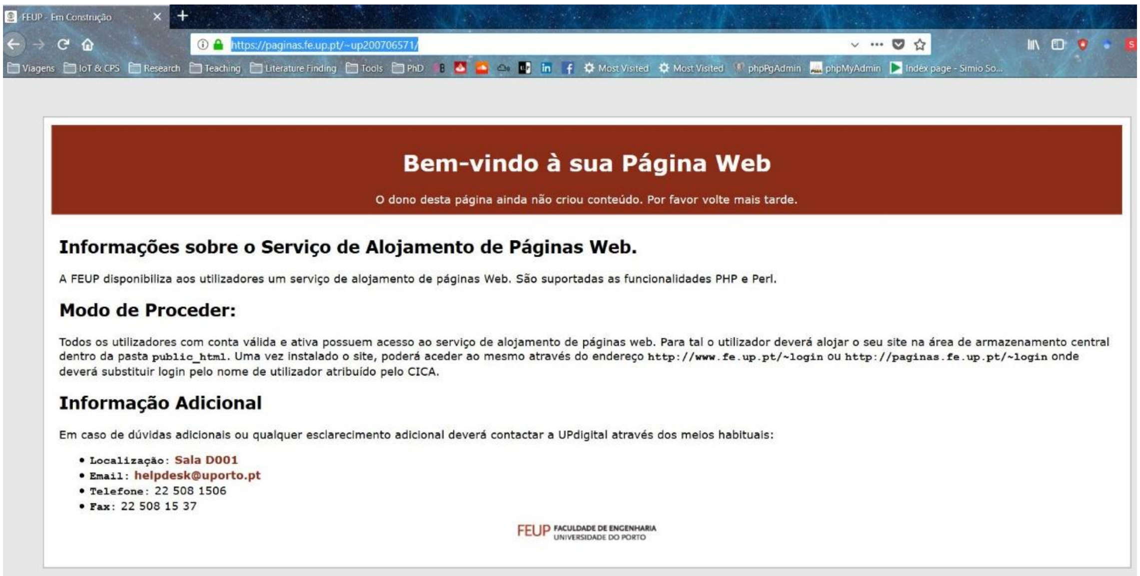



*Figure 7 - Default Personal Webpage.*

### b. HAS Website Characterization

As mentioned before, the main goals of Sprint 3 will cover mostly the development of complex SQL queries, in order to retrieve system data from the developed database in Sprint 2 (in several application questions). In this case, you should use a web interface for the HAS that is already provided in order to deploy those SQL queries.

The provided HAS Web interface is available in GitHub, following the URL (*https://github.com/SINF-FEUP/HAS-Web-Interface*). You should download the file '*HASWebInterface.zip*', where the Website corresponds to all the web content compressed in the zip file. You should extract the content to a folder and deploy the provided web interface, i.e., the folder with the web content unzipped, in a remote web server, namely the *samba.fe.up.pt* server. Then, you can access and render the web interface using a web browser, by accessing the URL '*https://paginas.fe.up.pt/~upxxx/sinf*', where, in this case, '*sinf*' is the name given to the folder with the unzipped content. The index page of the website provided corresponds to an HTML form, which is represented in Figure 8. Note that this website was optimized for Internet Explorer and Mozilla Firefox. It was also tested in Google Chrome and there are some issues regarding rendering, namely in the radio buttons.

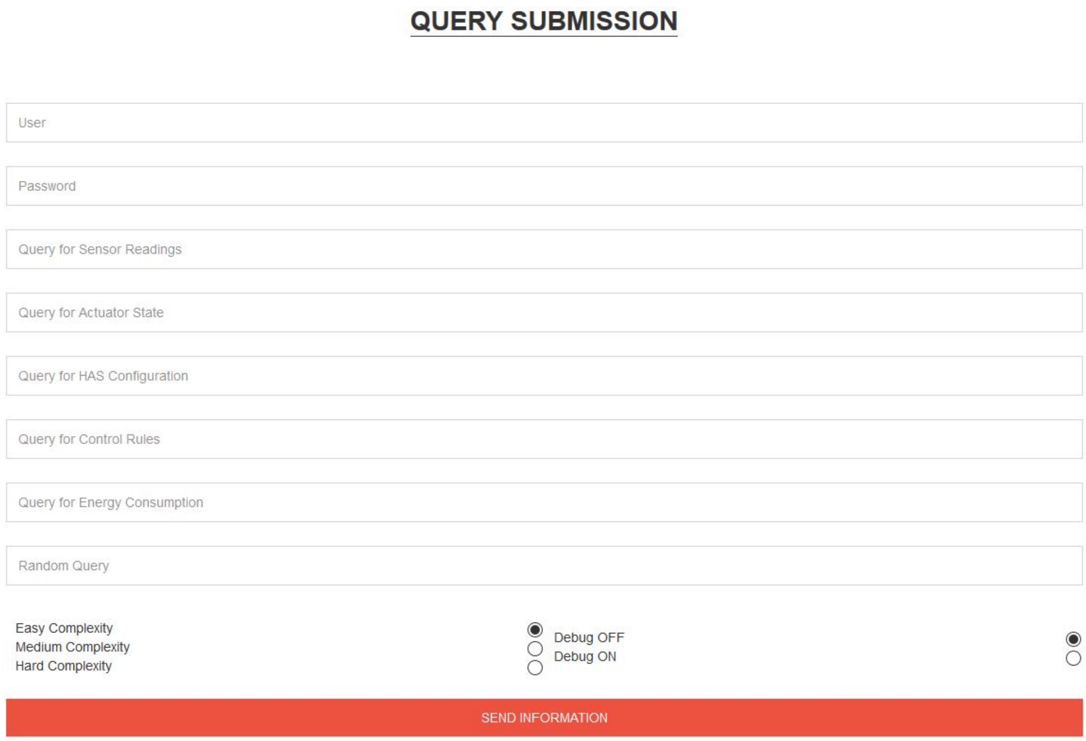



*Figure 8 - HAS Website Index Page.*

The form consists of 5 different areas:

1. **User and Password text input fields**. These inputs are mandatory, and you should introduce the username and password already provided, in order to have access to each team own database;
2. **Six different text input fields**, namely: Query for Sensor Readings; Query for Actuator State; Query for HAS Configuration; Query for Control Rules; Query for Energy Consumption; Random Query. These fields should be used to target the SQL queries necessary to respond to several questions regarding each team's application and targeted for specific topic. Note that the Random Query field may be used to execute general queries, without any specific topic target. More information regarding these questions is provided in the next section;
3. **A radio button for the difficulty level**, namely Easy, Medium and Hard Complexity. This corresponds to the difficulty level of each question / query to be submitted;
4. **A radio button for the activation of a debug option**, namely Debug OFF and Debug ON. Activating debug mode may help the students to get more feedback and resolve issues regarding submitted queries. You may

also use the *phppgadmin* web interface available in https://db.fe.up.pt/phppgadmin/ as a support for query debugging.

5. **A submit button**. This button submits the entire form at once, which means that every different query introduced will be executed at the same time and the results will be presented in the same Web page at once.

## 4. Application Questions

In this sprint, 15 application questions are proposed. The main goal is for each team, based on its own database implementation, to input the corresponding queries into the website, which will correctly return results that answer to the proposed application questions. These questions are grouped by its level of difficulty (easy, medium and hard) and target topic, i.e., sensor data, actuator data, configuration data, control rules data and energy data. Next it is presented a table that summarizes the proposed application questions.

| Query Topic | Easy | Medium | Hard |
|---|---|---|---|
| Sensor Data | Measurement history of one sensor of type A, which is present in room B, within a time duration C. | Average of all sensors of type A that are present in all rooms, within a time duration C. | Average of all sensors from all sensor types, which are present in every room, within a time duration C. |
| Actuator Data | Current state of all actuators that are installed in every room. | Number of state changes of every actuator of type D, which is installed in every room, within a time duration C. | Within a time duration of C, actuator identification and the duration of time that the actuator that remained the longest in a state E. |
| Configuration Data | Considering only two sensor motes and two rooms, where each sensor mote is present in a different room, change the location of both sensor motes, so each room will be attributed with the sensor mote that was located in the other room. | Number of sensors available in every room. | Number of sensors and actuators available in every room. |
| Control Rules Data | Modify the reference value of a sensor of type A in a given rule. | Number of rules that exist in every room. | Reference value and current measurement of the sensor of type A, which is present in room B, within a time duration C. |
| Energy Data | Consumption of energy (kWh) in a room B, within a time duration C. | Energy cost (in €) in all rooms, within a time duration C. | Energy cost (in €) in all rooms, within a time duration C, considering different energy rates for peak and off-peak hours. |

In the presented application questions:

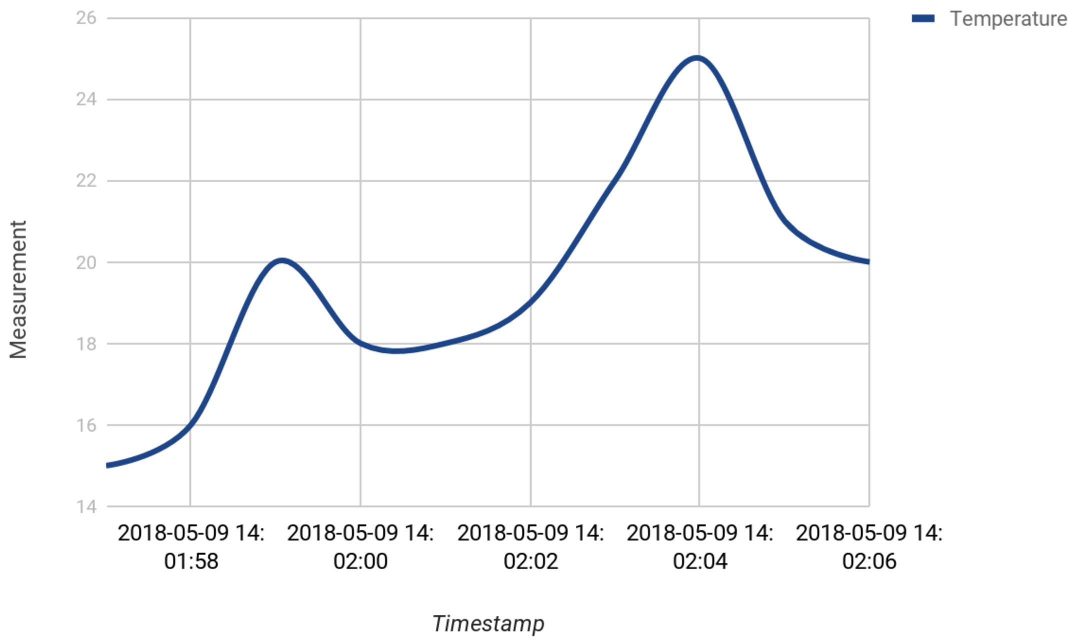
- A type of sensor A refers to the sensor types, namely Temperature, Humidity, Luminosity or Current.
- Room B refers to one of the considered rooms in the developed application, e.g., kitchen, living room, bathroom, etc.
- Time duration C refers to a given time window considered, that depends on the duration that the application was collecting sensor data and/or inferring actuator's state and the amount of data available in the database, e.g., 1min, 10min, 30min, 1h, 1 day, 1 month.
- A type of actuator D refers to a family of actuators, i.e., devices that have a direct impact on a given physical property, e.g., heater and AC that change the room temperature, lightbulb and window blinds that change the room luminosity, etc.
- Actuator state E refers to the considered state of a given actuator, e.g., on/off, open/closed, or even continuous states, such as percentage of intensity, etc.
- The value F refers to the threshold used for the physical property, which will have an impact on the state of actuators of a given type. For example, if one considers the room luminosity as the physical property, the value F refers to the threshold consider for the measurements of the luminosity sensor.

5. **Easy Application Questions Results**

Since each team had implemented its own database, which is naturally different from other teams, the SQL queries to resolve the proposed questions will also be different for each team. Next, for every easy application question, it is present the expected SQL query return.

1) Sensor Data

- Measurement history of one sensor of type _A_, which is present in room _B_, within a time duration _C_.
  - Expected to return a multi array with the timestamp and the measurement of a given sensor. Note that the query must have column alias for the timestamp values as '*timestamp*' and the measurement values as '*measurement*'. Also, consider returning the result ordered by ascending timestamp. The output of the query result will be represented as a line chart.




2) Actuator Data

- Current state of all actuators that are installed in every room.
  - Expected to return a table that contains at least information regarding the actuator name, corresponding room and most recent state of each presented actuator. The output of the query result will be represented as a table.

| Actuator | Room | State |
|---|---|---|
| AC | Living Room | ON |
| Lightbulb | Bedroom | 65% |
| Window blind | Kitchen | OPEN |

3) Configuration Data

- Considering only two sensor motes and two rooms, where each sensor mote is present in a different room, change the location of both sensor motes, so each room will be attributed with the sensor mote that was located in the other room.
    - Expected to modify an initial configuration, in which two sensor mote-room pares **a-b** and **c-d** changes to **a-d** and **c-b**. Note that the query should return the result of the operation performed, in the format of a table, which contains at least information regarding the sensor name and corresponding room. The output of the query result will be represented as a table.

| Sensor | Room |
|---|---|
| Temp_Kitchen | Living Room |
| Temp_LivingRoom | Kitchen |

4) Control Rules Data

- Modify the reference value of a sensor of type <u>*A*</u> in a given rule.
    - Expected to modify a reference value of an existing control rule. Note that the query should return the result of the operation performed, in the format of a table, which contains at least information regarding the rule name or identifier, corresponding room, updated reference value and corresponding sensor. The output of the query result will be represented as a table.

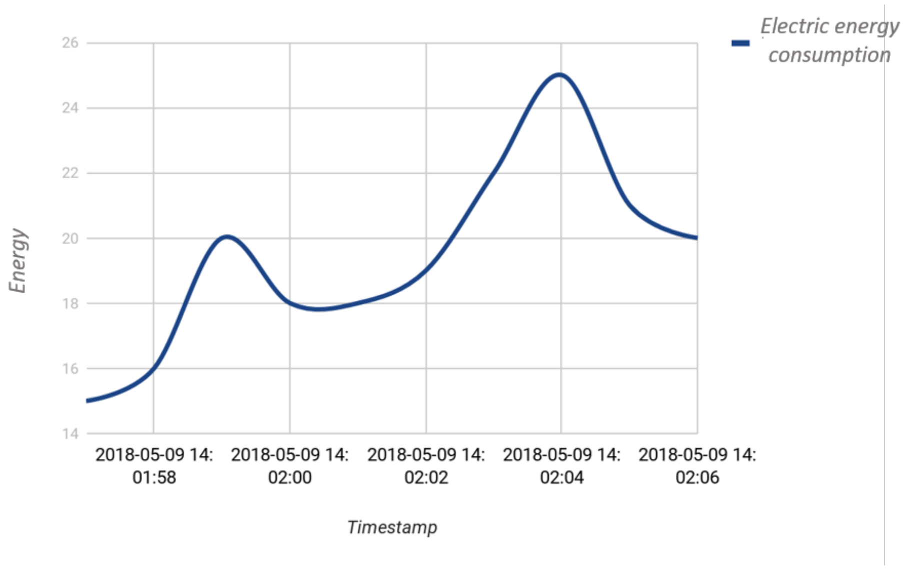
| Rule | Room | Reference Value | Sensor |
|---|---|---|---|
| Rule10 | Living Room | 23 | Temp_LivingRoom |

5) Energy Data

- Consumption of energy in a room <u>*B*</u>, within a time duration <u>*C*</u>.
    - Expected to return a multi array with the timestamp and the energy consumption in a given room. Note that the query must have column alias for the timestamp values as '<u>*timestamp*</u>' and the energy consumption values as '<u>*energy*</u>'. Also, consider returning the result ordered by ascending timestamp. The output of the query result will be represented as a line chart.
    - Note that the electric energy consumption ($E$) is most often measured in $kWh$, and it depends on the power rating in the electrical appliance ($P$), given in $kW$, and its usage time ($\Delta t$), given in $h$. Also, the electric power ($P$), given in $w$, can be calculated simply by the product of the electric current ($I$), given in $A$, times voltage ($V$), given in $V$.

$$E = P \times \Delta t$$
$$P = I \times V$$

    - You may use the electric current information measured by the current sensor available in the motes to calculate the electric power and the corresponding energy consumption. Depending on the number of motes in each room, you can consider that the sum of the current measurement off all motes in a given room corresponds to all electric equipment that exist in that room. If you don't have in your database information related to the operating voltage of the electric equipment, you can consider 220v.

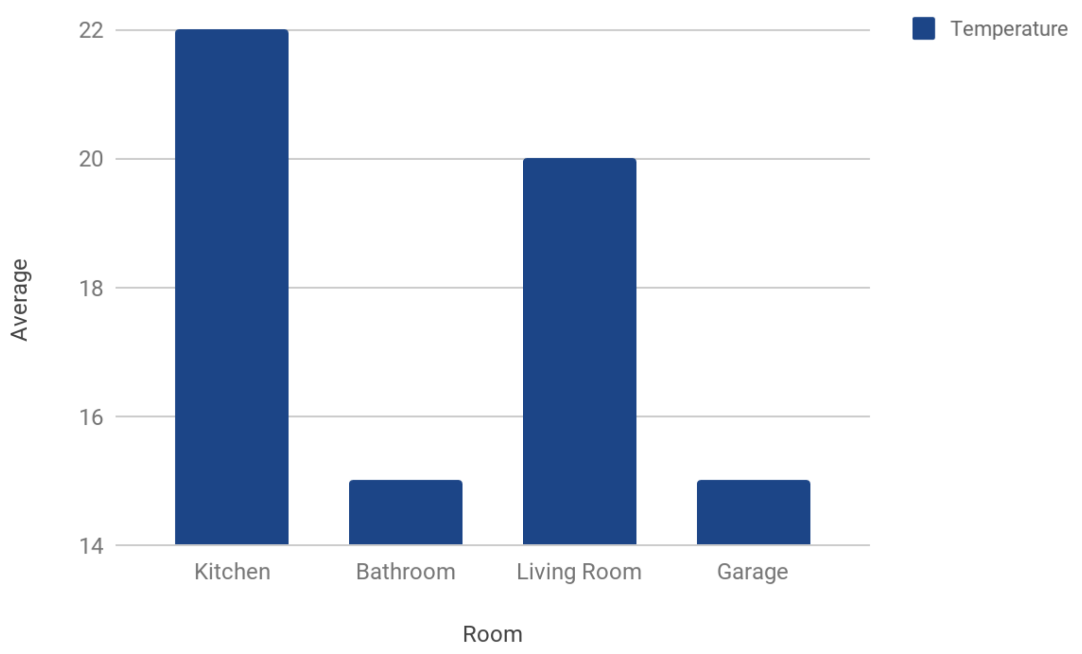


## 6. **<u>Medium</u> Application Questions Results**

Since each team had implemented its own database, which is naturally different from other teams, the SQL queries to resolve the proposed questions will also be different for each team. Next, for every medium application question, it is present the expected SQL query return.

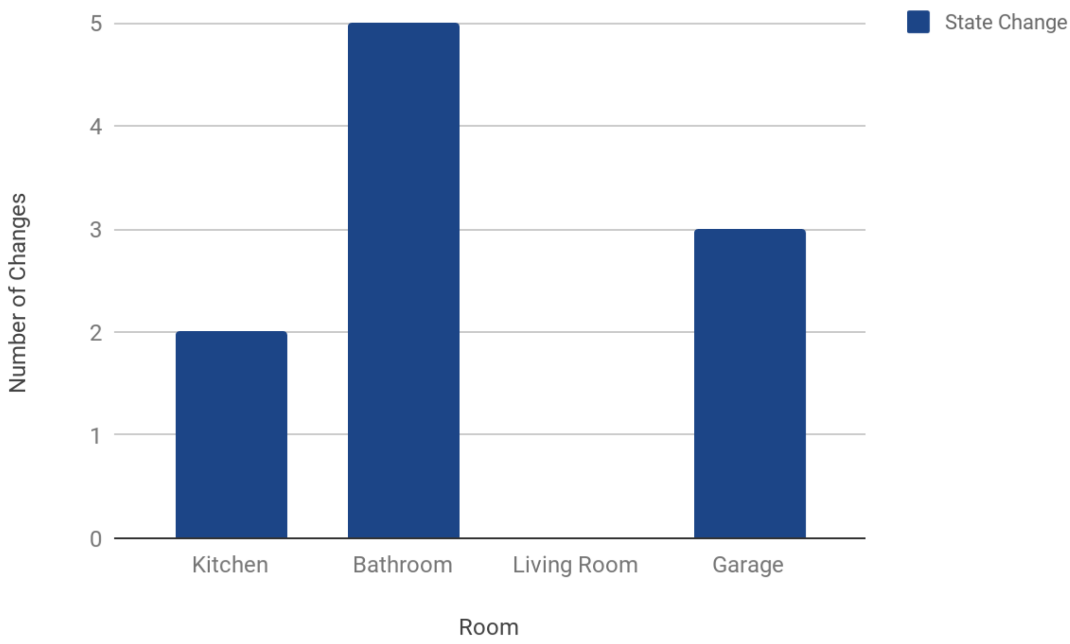
1) Sensor Data

- Average of all sensors of type *A* that are present in all rooms, within a time duration *C*.
  - Expected to return a multi array with the room and average of the sensors of a given type within a given time duration. Note that the query must have column alias for the room as '*room*' and the average values as '*average*'. The output of the query result will be represented as a bar chart.




2) Actuator Data

- Number of state changes of every actuator of type *D*, which is installed in every room, within a time duration *C*.
  - Expected to return a multi array with the room and the corresponding count of state changes of a family of actuators that is installed in it, in a given time duration. Note that the query must have column alias for the room as '*room*' and the changing count as '*change*'. The output of the query result will be represented as a bar chart.



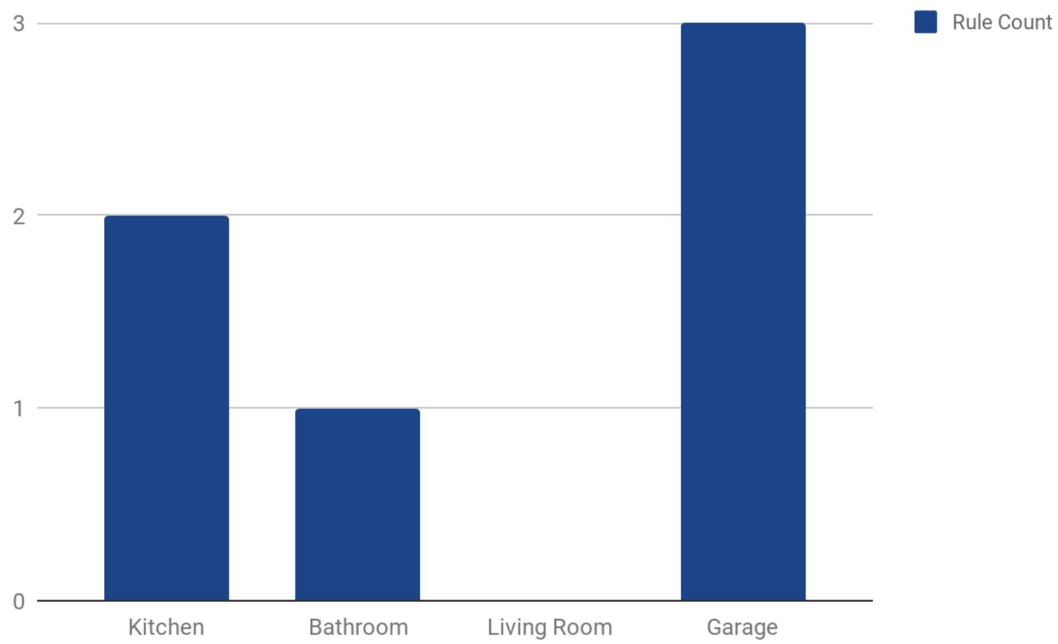

3) Configuration Data

- Number of sensors available in every room.
    - Expected to return a table that contains at least information regarding the room, mote and sensor count. The output of the query result will be represented as a table.

| Room | Mote Count | Sensor Count |
|---|---|---|
| Living Room | 1 | 4 |
| Kitchen | 2 | 6 |

4) Control Rules Data

- Number of rules that exist in every room.
    - Expected to return a multi array with the room and the corresponding count of associated rules. Note that the query must have column alias for the room as ‘*room*’ and the rule count as ‘*rules*’. The output of the query result will be represented as a bar chart.



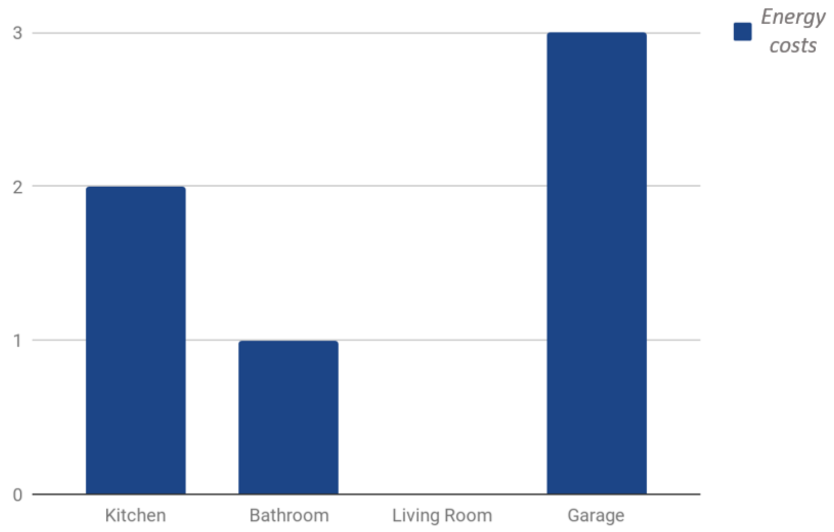

5) Energy Data

- Energy cost in all rooms, within a time duration <u>*C*</u>.
    - Expected to return a multi array that contains at least information regarding the room and energy bill. Note that the query must have column alias for the room as '<u>*room*</u>' and energy cost as <u>*'cost'*</u>. The output of the query result will be represented as a bar chart.
    - Note that the electric energy consumption $(E)$ is most often measured in $kWh$, and it depends of the power rating in the electrical appliance $(P)$, given in $kW$, and its usage time $(\Delta t)$, given in $h$. Also, the electric power $(P)$, given in $w$, can be calculated simply by the product of the electric current $(I)$, given in $A$, times voltage $(V)$, given in $V$. Finally, the total cost $(C)$ of the energy consumption, given in €, can be calculated by the product of the rate $(R)$ of the energy source company, given in €, times the energy consumption $(E)$ in a given period of time, given by $kWh$.

$$E = P \times \Delta t$$
$$P = I \times V$$

$$C = R \times E$$

    - You may use the electric current information measured by the current sensor available in the motes to calculate the electric power and the corresponding energy consumption. Depending on the number of motes in each room, you can consider that the sum of the current measurement off all motes in a given room corresponds to all electric equipment that exist in that room. If you don't have in your database information related to the operating voltage of the electric equipment, you can consider 220v. You may also consider 0.1544€ as the rate of a unit of energy consumed.


Energy costs
3
2
1
0
Kitchen
Bathroom
Living Room
Garage

## 7. **Hard Application Questions Results**

Since each team had implemented its own database, which is naturally different from other teams, the SQL queries to resolve the proposed questions will also be different for each team. Next, for every hard application question, it is present the expected SQL query return.

1) Sensor Data

- Average of all sensors from all sensor types, which are present in every room, within a time duration *C*.
    - Expected to return a table with the room and average of the all sensors of every type within a given time duration. The result should contain at least information regarding the room, sensor type and average of measurements. The output of the query result will be represented as a table.

| Room | Sensor Type | Average |
|---|---|---|
| Kitchen | Humidity | 60 |
| Living Room | Temperature | 23 |

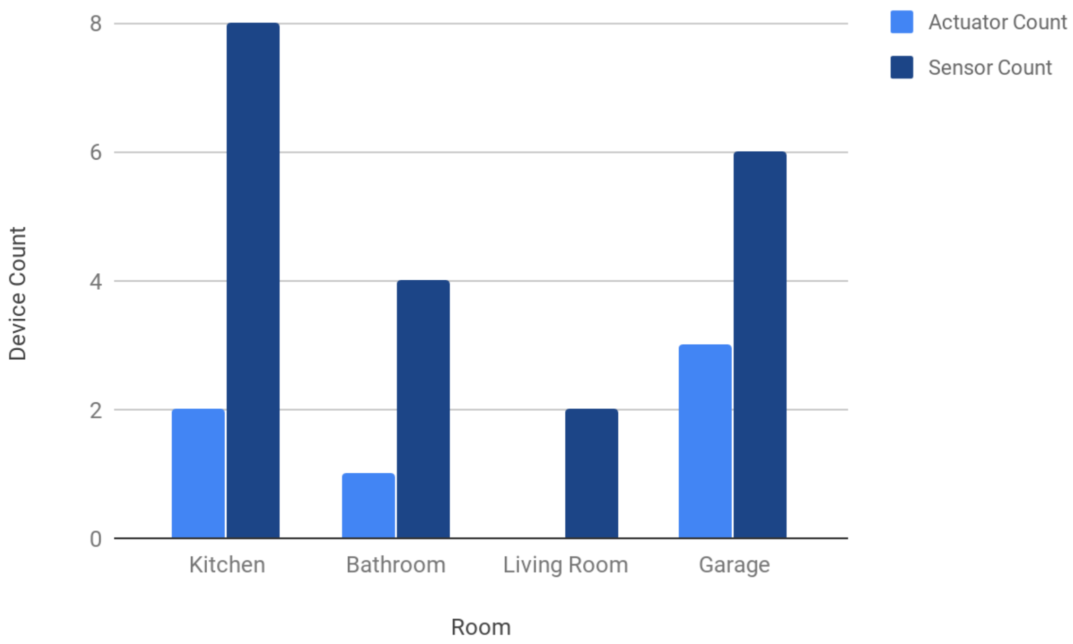
2) Actuator Data

- Within a time duration of *C*, actuator identification and the duration of time that the actuator that remained the longest in a state *E*.
    - Expected to return a table that should contains at least information regarding the actuator name, actuator state and sum of time duration in that state. The output of the query result will be represented as a table and should contain only one row, which corresponds to the actuator that remained the longest in a given state.

| Actuator Name | State | Duration |
|---|---|---|
| Window blind | CLOSE | 16 |

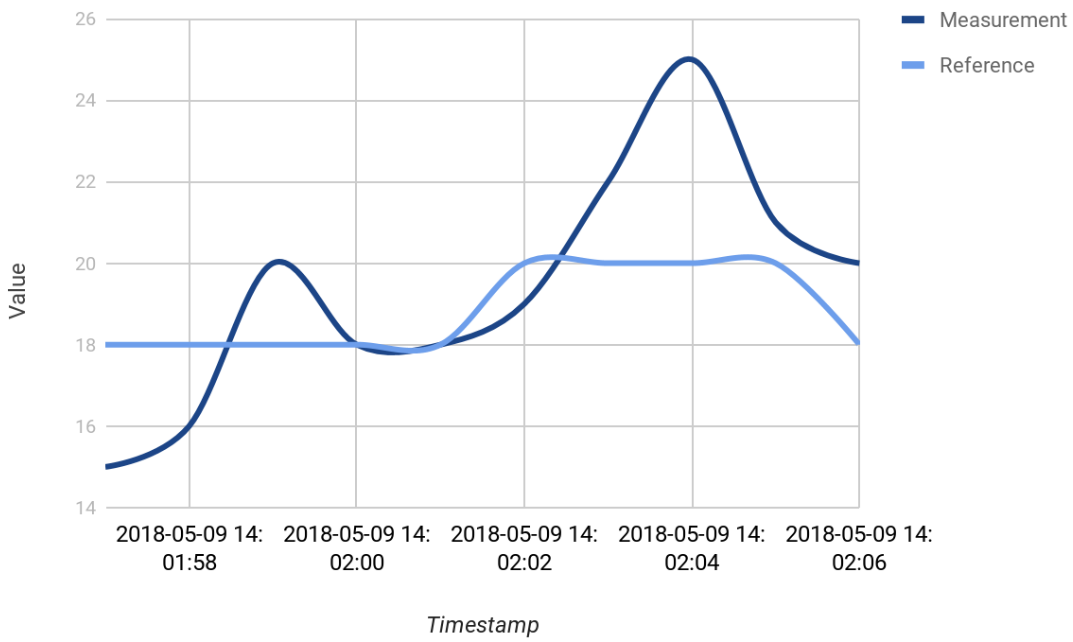
3) Configuration Data

- Number of sensors and actuators available in every room.
    - Expected to return a multi array that contains at least information regarding the room, actuator and sensor count. Note that the query must have column alias for the room as '*room*', actuator count as '*actuator*' and sensor count as '*sensor*'. The output of the query result will be represented as a bar chart.




4) Sensor Rules Data

- Reference value and current measurement of the sensor of type <u>*A*</u>, which is present in room <u>*B*</u>, within a time duration <u>*C*</u>.
  - Expected to return a multi array with the timestamp, the measurement of a given sensor and the corresponding reference value in a given room, within a given time duration. Note that the query must have column alias for the timestamp values as '<u>*timestamp*</u>', the measurement values as '<u>*measurement*</u>' and the reference value as '<u>*reference*</u>'. Also, consider returning the result ordered by ascending timestamp. The output of the query result will be represented as a line chart.



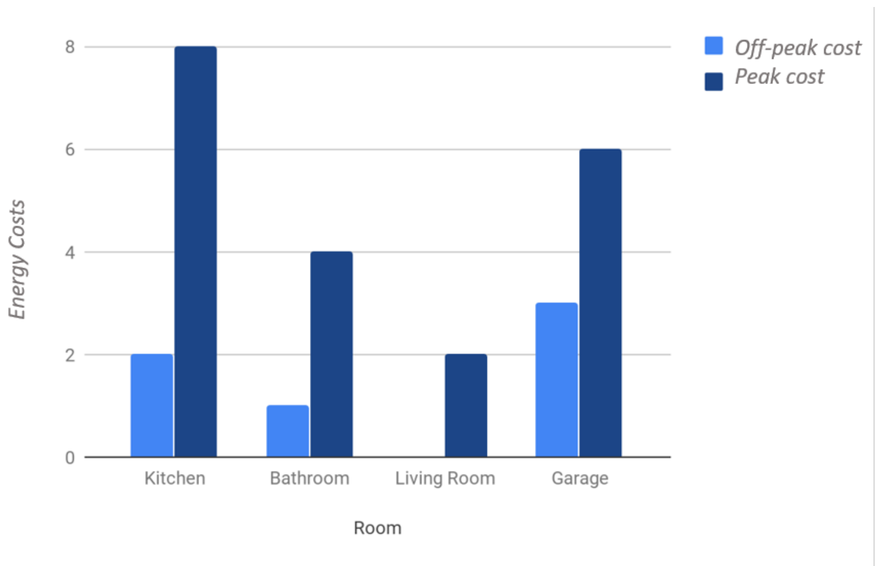

5) Energy Data

- Energy cost in all rooms, within a time duration C, considering different energy rates for peak and off-peak hours.
  - Expected to return a multi array that contains at least information regarding the room, off-peak and peak costs. Note that the query must have column alias for the room as '*room*', off-peak costs as *'offpeak'* and peak costs as *'peak'*. The output of the query result will be represented as a bar chart.
  - Note that the electric energy consumption ($E$) is most often measured in $kWh$, and it depends of the power rating in the electrical appliance ($P$), given in $kW$, and its usage time ($\Delta t$), given in $h$. Also, the electric power ($P$), given in $w$, can be calculated simply by the product of the electric current ($I$), given in $A$, times voltage ($V$), given in $V$. Finally, the total cost ($C$) of the energy consumption, given in €, can be calculated by the product of the rate ($R$) of the energy source company, given in €, times the energy consumption ($E$) in a given period of time, given by $kWh$.

$$E = P \times \Delta t$$
$$P = I \times V$$

$$C = R \times E$$

  - You may use the electric current information measured by the current sensor available in the motes to calculate the electric power and the corresponding energy consumption. Depending on the number of motes in each room, you can consider that the sum of the current measurement off all motes in a given room corresponds to all electric equipment exist in that room. If you don't have in your database information related to the operating voltage of the electric equipment, you can consider 220v. You may also consider 0.1870€ as the peak rate and 0.1100€ as the off-peak rate of a unit of energy consumed.



## 8. Practical Examples

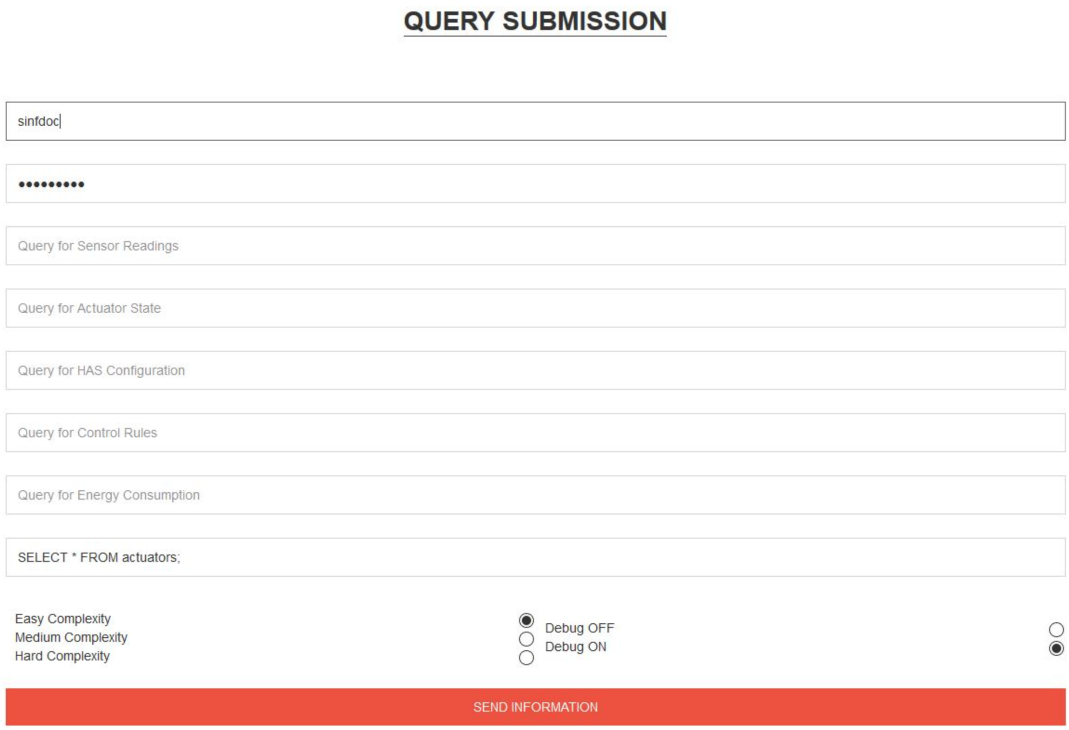
In this section is presented two simple examples that represent the process of query submission and result presentation.

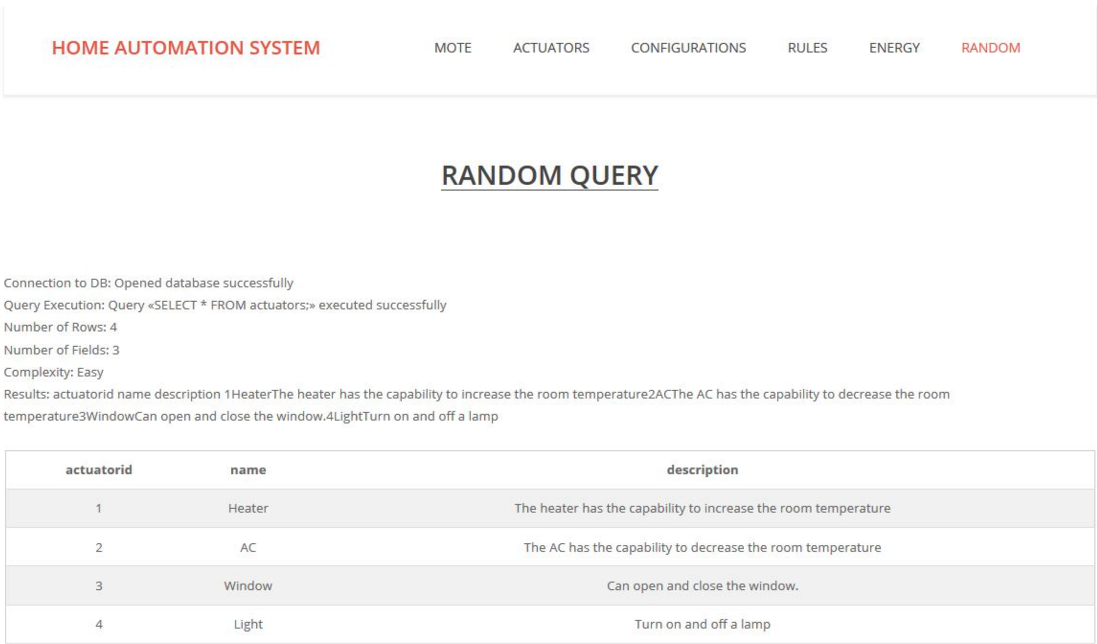
### a. Measurement history of a temperature sensor for 10s

To resolve this question, the SQL query '*SELECT timestamp AS timestamp, value AS measurement FROM sensordata*' was introduced in the '*Query for Sensor Reading*' text field, with the '*Easy Complexity*' selected. The result is presented in the '*Mote'* tab of the Website.

b. **Actuator listing with information regarding name and description**

To resolve this question, the SQL query '*SELECT * FROM actuators*' was introduced in the '*Random Query*' text field. Note that, when using the '*Random Query*', the complexity choice doesn't matter. The result is presented in the '*Random Query*', along with debug information.




| actuatorid | name | description |
|---|---|---|
| 1 | Heater | The heater has the capability to increase the room temperature |
| 2 | AC | The AC has the capability to decrease the room temperature |
| 3 | Window | Can open and close the window. |
| 4 | Light | Turn on and off a lamp |

## 9. Sprint Main Goals

According to the Projects Dashboard, the main goals for Sprint 3 are represented in Figure 9.

| Attendance Grade | Goal | Appreciation |
|---|---|---|
| 6 points | Project Planning (Sprint Backlog) | 4% |
| | Setup and deploy the provided Web interface | 5% |
| | Design and test the queries - Easy | 12% |
| 8 points | Design and test the queries - Medium | |
| 10 points | Design and test the queries - Hard | |

*Figure 9 - Project's Dashboard Goals.*

All goals are related to several technical tasks (except the project planning goal). Next, more detail is given regarding the technical tasks of each goal.

1) **Setup and deploy the provided Web interface:** The HAS Web interface is available to download in GitHub. This interface should be deployed in the samba server, in order to access it remotely using a Web browser. You can see more information in Sections 3. **HAS Web Interface** and **Error! Reference source not found.**. **Error! Reference source not found.**.

2) **Design and test the queries – easy:** Develop the SQL queries of easy complexity and test them into the HAS Web interface. Note that these SQL queries should be placed in your GitHub repository and is mandatory to validate them in the HAS Web interface. You can see more information in Section 4**Error! Reference source not found.**. **Application Questions - Error! Reference source not found.**. The SQL queries of easy complexity are:

   a. Measurement history of one sensor of type <u>*A*</u>, which is present in room <u>*B*</u>, within a time duration <u>*C*.</u>

   *b.* Current state of all actuators that are installed in every room.
   c. Considering only two sensor motes and two rooms, where each sensor mote is present in a different room, change the location of both sensor motes, so each room will be attributed with the sensor mote that was located in the other room.
   d. Modify the reference value of a sensor of type <u>*A*</u> in a given rule.
   e. Consumption of energy in a room <u>*B*</u>, within a time duration <u>*C*</u>.

6) Design and test the queries – medium: Develop the SQL queries of medium complexity and test them into the HAS Web interface. Note that these SQL queries should be placed in your GitHub repository and is mandatory to validate them in the HAS Web interface. You can see more information in Section 45)5). **Application Questions** - Energy Data

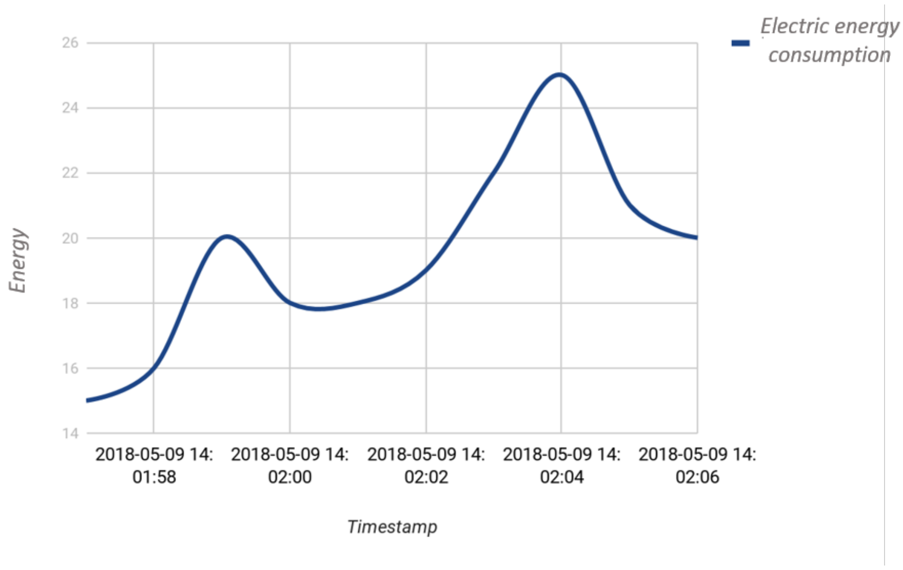
- Consumption of energy in a room <u>*B*</u>, within a time duration <u>*C*</u>.
  - Expected to return a multi array with the timestamp and the energy

consumption in a given room. Note that the query must have column alias for the timestamp values as ‘*timestamp*’ and the energy consumption values as ‘*energy*’. Also, consider returning the result ordered by ascending timestamp. The output of the query result will be represented as a line chart.

- Note that the electric energy consumption ($E$) is most often measured in $kWh$, and it depends on the power rating in the electrical appliance ($P$), given in $kW$, and its usage time ($\Delta t$), given in $h$. Also, the electric power ($P$), given in $w$, can be calculated simply by the product of the electric current ($I$), given in $A$, times voltage ($V$), given in $V$.

$$E = P \times \Delta t$$
$$P = I \times V$$

- You may use the electric current information measured by the current sensor available in the motes to calculate the electric power and the corresponding energy consumption. Depending on the number of motes in each room, you can consider that the sum of the current measurement off all motes in a given room corresponds to all electric equipment that exist in that room. If you don’t have in your database information related to the operating voltage of the electric equipment, you can consider 220v.




3) . The SQL queries of medium complexity are:

   a. Average of all sensors of type *A* that are present in all rooms, within a time duration *C.*

   b. Number of state changes of every actuator of type *D*, which is installed in every room, within a time duration *C*.

   c. Number of sensors available in every room.

   d. Number of rules that exist in every room.

e. Energy cost in all rooms, within a time duration <u>*C*</u>.

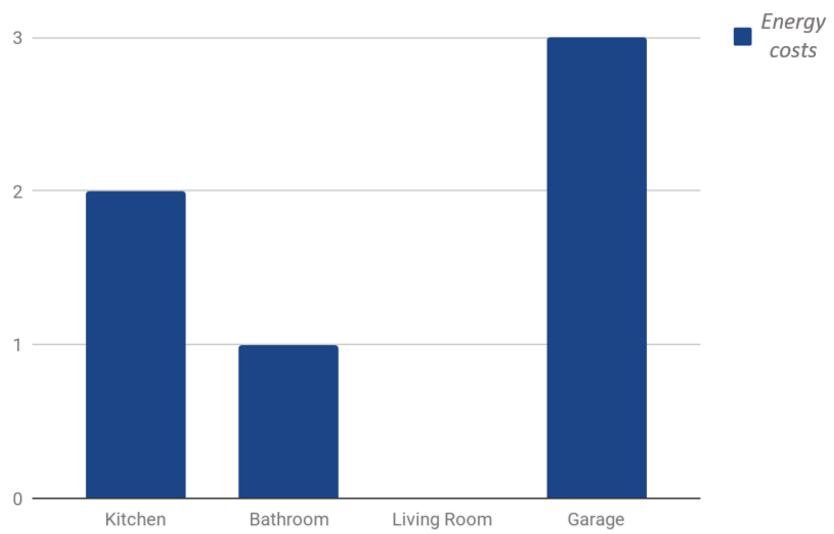
6) Design and test the queries – hard: Develop the SQL queries of hard complexity and test them into the HAS Web interface. Note that these SQL queries should be placed in your GitHub repository and is mandatory to validate them in the HAS Web interface. You can see more information in Section 45). **Application Questions** - Energy Data

- Energy cost in all rooms, within a time duration <u>*C*</u>.
  - Expected to return a multi array that contains at least information regarding the room and energy bill. Note that the query must have column alias for the room as '<u>*room*</u>' and energy cost as <u>*'cost'*</u>. The output of the query result will be represented as a bar chart.
  - Note that the electric energy consumption $(E)$ is most often measured in $kWh$, and it depends of the power rating in the electrical appliance $(P)$, given in $kW$, and its usage time $(\Delta t)$, given in $h$. Also, the electric power $(P)$, given in $w$, can be calculated simply by the product of the electric current $(I)$, given in $A$, times voltage $(V)$, given in $V$. Finally, the total cost $(C)$ of the energy consumption, given in €, can be calculated by the product of the rate $(R)$ of the energy source company, given in €, times the energy consumption $(E)$ in a given period of time, given by $kWh$.

$$E = P \times \Delta t$$
$$P = I \times V$$

$$C = R \times E$$

  - You may use the electric current information measured by the current sensor available in the motes to calculate the electric power and the corresponding energy consumption. Depending on the number of motes in each room, you can consider that the sum of the current measurement off all motes in a given room corresponds to all electric equipment that exist in that room. If you don't have in your database information related to the operating voltage of the electric equipment, you can consider 220v. You may also consider 0.1544€ as the rate of a unit of energy consumed.




4) . The SQL queries of hard complexity are:

   a. Average of all sensors from all sensor types, which are present in every room, within a time duration C.

   b. Within a time duration of C, actuator identification and the duration of time that the actuator that remained the longest in a state E.

   c. Number of sensors and actuators available in every room.

   d. Reference value and current measurement of the sensor of type A, which is present in room B, within a time duration C.

   e. Energy cost in all rooms, within a time duration C, considering different energy rates for peak and off-peak hours.