**Requirements Testing**

**Overview**

**Content**

[**Content** 2](#_Toc501445409)

[**1.** **GUI** 3](#_Toc501445410)

[**2.** **Detect badge** 4](#_Toc501445411)

[**3.** **Database** 5](#_Toc501445412)

[**4.** **Relate information** 8](#_Toc501445413)

[**5.** **Energy button** 9](#_Toc501445414)

[**6.** **Web page to upload menu** 9](#_Toc501445415)

[**7.** **Web page to manage employees** 10](#_Toc501445416)

[**8** **Convert spreadsheet to database** 12](#_Toc501445417)

[**9.** **Service buttons** 13](#_Toc501445418)

[**10.** **Export table of DB to spreadsheet** 13](#_Toc501445419)

# **GUI**

* 1. **Photo:**
     1. It shall display the photo of employee.
     2. The size of the photo shall be 300x300 pixels.
     3. The name shall not be longer than 9 characters.
     4. The name shall be five numbers, hyphen, one number, hyphen, one number (without spaces). Eg ‘12345-6-7’
     5. It shall be in format PNG.
     6. In case of error when try loading photo, it shall show a guest photo.
     7. It shall be in the folder ‘Photos’ of the BK project.

|  |  |
| --- | --- |
| **Verification:** | **Observations:** |
| 1.1.A. | Initial condition:   * Startup Raspberry (GUI runs automatically) * Press yellow button (canceled button) so the RFID starts to read   Test action:   * Swipe badge of an employee   Expected result:   * In the top left of the GUI screen appears the photo of the respective employee |
| 1.1.B. | In the photos properties shows that the size is 300x300 pixels, as mentioned in the requirement. |
| 1.1.C. | No name of the photos exceeds 9 characters; in fact, they are all 9 characters long, as mentioned in the requirement. |
| 1.1.D. | All the names of the photos respect the condition, as mentioned in the requirement. |
| 1.1.E. | In the photos properties shows that the format is PNG, as mentioned in the requirement. |
| 1.1.F. |  |
| 1.1.G. | All the photos are located in the folder specify in the requirement. |

* 1. **Name:**
     1. It shall display the employee name (first name/last name).
     2. The name shall respect uppercase and lowercase letters.
     3. It shall not be longer than 80 characters.

|  |  |
| --- | --- |
| **Verification:** | **Observations:** |
| 1.2.A. | Initial condition:   * Startup Raspberry (GUI runs automatically) * Press yellow button (canceled button) so the RFID starts to read   Test action:   * Swipe badge of an employee   Expected result:  Next to the photo of the employee is shown the first name and first surname of the respective employee. |
| 1.2.B. | The name respect uppercase and lowercase letters, as mentioned in the requirement. |
| 1.2.C. | None of the names exceed 8 characters when shown, as mentioned in the requirement. |

* 1. **Company:**
     1. It shall display the logo of company’s employee.
     2. The size of the logo shall be 300x72 pixels.
     3. The name of the photo should be the company acronym in uppercase.
     4. It shall be in format PNG.
     5. It shall be stored in the folder ‘Logos’ of the BK project.
     6. In case of error when try loading logo, it shall not show a logo.

|  |  |
| --- | --- |
| **Verification:** | **Observations:** |
| 1.3.A. | Initial condition:   * Startup Raspberry (GUI runs automatically) * Press yellow button (canceled button) so the RFID starts to read   Test action:   * Swipe badge of an employee   Expected result:   * In the bottom of the employee´s photo in the GUI screen appears the company´s logo of the respective employee |
| 1.3.B. | In the photos properties shows that the size is 300x72 pixels, as mentioned in the requirement. |
| 1.3.C. | All the names of the photos respect the condition, as mentioned in the requirement. |
| 1.3.D. | In the photos properties shows that the format is PNG, as mentioned in the requirement. |
| 1.3.E. | All the photos are located in the folder specify in the requirement. |
| 1.3.F. |  |

* 1. **Dish:**
     1. It shall contain the dish name and dish number.
     2. Dish name shall not be longer than 80 characters.
     3. Dish number shall not be longer than 6 characters. Eg. ‘Guiso1’.

|  |  |
| --- | --- |
| **Verification:** | **Observations:** |
| 1.4.A. | Initial condition:   * Startup Raspberry (GUI runs automatically) * Press yellow button (canceled button) so the RFID starts to read   Test action:   * Swipe badge of an employee   Expected result:   * Under the employee´s name appears the dish number ordered by the respective employee * Down centered will appear the dish name ordered by the respective employee |
| 1.4.B. | The dish name respect the length mentioned in the requirement. |
| 1.4.C. | The dish number respect the length mentioned in the requirement. |

* 1. **Interactive background:** Depend of dish number, if it’s VIP or if there is an error.
     1. **Guiso 1:**
        1. It shall show the hex color: #444141 in a rectangle on screen.
        2. Shall named ‘Interactive\_background\_1’
        3. It shall be in format PNG.
        4. It shall be in the folder ‘Texture’ of the BionicKitchen project.

|  |  |
| --- | --- |
| **Verification:** | **Observations:** |
| 1.5.1.A. | Initial condition:   * Startup Raspberry (GUI runs automatically) * Press yellow button (canceled button) so the RFID starts to read   Test action:   * Swipe badge of an employee   Expected result:  When the respective employee choose the dish #1, it show the name of the dish inside of a rectangle of the color mentioned in the requirement. |
| 1.5.1.B. | The names of the photo respect the condition, as mentioned in the requirement. |
| 1.5.1.C. | In the photo properties shows that the format is PNG, as mentioned in the requirement. |
| 1.5.1.D. | All the photos are located in the folder specify in the requirement. |

* + 1. **Guiso 2:**
       1. It shall show the hex color #931A21in a rectangle on screen.
       2. Shall named ‘Interactive\_background\_2’
       3. It shall be in format PNG.
       4. It shall be in the folder ‘Texture’ of the BionicKitchen project.

|  |  |
| --- | --- |
| **Verification:** | **Observations:** |
| 1.5.2.A. | Initial condition:   * Startup Raspberry (GUI runs automatically) * Press yellow button (canceled button) so the RFID starts to read   Test action:   * Swipe badge of an employee   Expected result:  When the respective employee choose the dish #2, it show the name of the dish inside of a rectangle of the color mentioned in the requirement. |
| 1.5.2.B. | The names of the photo respect the condition, as mentioned in the requirement. |
| 1.5.2.C. | In the photo properties shows that the format is PNG, as mentioned in the requirement. |
| 1.5.2.D. | All the photos are located in the folder specify in the requirement. |

* + 1. **Guiso 3:**
       1. It shall show the hex color: #2B2A2A in a rectangle on screen.
       2. Shall named ‘Interactive\_background\_3’
       3. It shall be in format PNG.
       4. It shall be in the folder ‘Texture’ of the BionicKitchen project.

|  |  |
| --- | --- |
| **Verification:** | **Observations:** |
| 1.5.3.A. | Initial condition:   * Startup Raspberry (GUI runs automatically) * Press yellow button (canceled button) so the RFID starts to read   Test action:   * Swipe badge of an employee   Expected result:  When the respective employee choose the dish #3, it show the name of the dish inside of a rectangle of the color mentioned in the requirement. |
| 1.5.3.B. | The names of the photo respect the condition, as mentioned in the requirement. |
| 1.5.3.C. | In the photo properties shows that the format is PNG, as mentioned in the requirement. |
| 1.5.3.D. | All the photos are located in the folder specify in the requirement. |

* + 1. **Error or VIP mode:**
    2. It shall show the color: #501B29 in a rectangle on screen
    3. Shall named ‘Interactive\_background\_4’
    4. It shall be in format PNG.
    5. It shall be in the folder ‘Texture’ of the BK project.
    6. If is error it shall show the text of error in hexadecimal.

|  |  |
| --- | --- |
| **Verification:** | **Observations:** |
| 1.5.4.A. | Initial condition:   * Startup Raspberry (GUI runs automatically) * Press yellow button (canceled button) so the RFID starts to read   Test action:   * Swipe badge of an employee   Expected result:   * When the respective employee vip, it show ‘Select a dish’ inside of a rectangle of the color mentioned in the requirement. * When an error happens, it show a message inside of a rectangle of the color mentioned in the requirement. |
| 1.5.4.B. | The names of the photo respect the condition, as mentioned in the requirement. |
| 1.5.4.C. | In the photo properties shows that the format is PNG, as mentioned in the requirement. |
| 1.5.4.D. | All the photos are located in the folder specify in the requirement. |
| 1.5.4.E. | When an error happens, it show the text of error in hexadecimal inside of a rectangle. |

# **Detect badge**

|  |  |
| --- | --- |
| **Verification:** | **Observation:** |
| 2 | Initial condition:   * Modify ‘rfid.py’ to print the signal before it processed and after * Execute the ‘rfid.py’ in the OS terminal   Test action:   * Swipe badge of an employee   Expected result:   * The first signal printed was: 0x7f3bc40612d0 * The processed signal printed was: 1300788161 |

* 1. It shall be an external hardware.
  2. The RFID Module shall detect the RFID tags of 125 kHz (Employee Badge).
  3. The file ‘rfid.py’ shall receive the tag number of RFID Module via USB(Port ttyUSB0).
  4. The RFID module shall be a RFID technology of low frequency of 125 kHz.
  5. Python script named ‘rfid.py’ shall process signal of the tag and drop noises.
  6. The output of the python script mentioned shall be an hexadecimal of 10 characters (Shall be the tag signal after it processed).

|  |  |
| --- | --- |
| **Verification:** | **Observations:** |
| 2.1 | The hardware device is: Parallax RFID Card Reader, USB (#28340) |
| 2.2 | All the employees tags are of 125 kHz, either the RFID Module wouldn’t read the tag. |
| 2.3 | The RFID Module is connected to the Raspberry via USB port , the the ‘Serial’ library is used to open the port in the Python script and when a tag is swipe, the script (‘rfid.py’) received the signal. |
| 2.4 | The RFID Technology used is of 125 kHz, it specify in the RFID Module official document. |
| 2.5 | The python script after received the signal, process the same to drop noises and just stay with 10 characters |
| 2.6 | The code is in a function, this function after the process the signal, returns the number of tag. |

# **Database**

* 1. **The database** shall be named **‘ODTC\_Service’.**

|  |  |
| --- | --- |
| **Verification:** | **Observation:** |
| 3.1. | The name of the database is ‘ODTC\_Service’, like is mentioned on the requirements. |

* + 1. **Table ‘Employees’** it shall contain the following parameters:

|  |  |
| --- | --- |
| **Verification:** | **Observation:** |
| 3.1.1. | The table ‘Employees’ in the DB ‘ODTC\_Service’ has the fields that are specified, based on what is mentioned in the requirements. |

* + - 1. **employee\_id:**

1. It shall be a type ‘int auto increment primary key’.
2. It shall not be null.

|  |  |
| --- | --- |
| **Verification:** | **Observation:** |
| 3.1.1.1.A | The field ‘employee\_id’ in the table ‘Employees’ has the specified characteristics, based on what is mentioned in the requirements. |
| 3.1.1.1.B |

* + - 1. **name:**

1. It shall be a type Varchar (80).
2. It shall contains the employee full name (first name/last name).
3. It shall not be repeatable.
4. It shall not be null.

|  |  |
| --- | --- |
| **Verification:** | **Observation:** |
| 3.1.1.2.A | The field ‘employee\_id’ in the table ‘Employees’ has the specified characteristics, based on what is mentioned in the requirements. |
| 3.1.1.2.B |
| 3.1.1.2.C |
| 3.1.1.2.D |

* + - 1. **badge:**

1. It shall be a type Varchar (10).
2. It shall contain 10 hexadecimal characters of badge.
3. It shall not be repeatable.
4. It shall not be null.

|  |  |
| --- | --- |
| **Verification:** | **Observation:** |
| 3.1.1.3.A | The field ‘badge’ in the table ‘Employees’ has the specified characteristics, based on what is mentioned in the requirements. |
| 3.1.1.3.B |
| 3.1.1.3.C |
| 3.1.1.3.D |

* + - 1. **company:**

1. It shall be a type ‘company\_id’.
2. It shall contain the company number generated in database table Company.
3. It shall not be null.

|  |  |
| --- | --- |
| **Verification:** | **Observation:** |
| 3.1.1.4.A | The field ‘company’ in the table ‘Employees’ has the specified characteristics, based on what is mentioned in the requirements. |
| 3.1.1.4.B |
| 3.1.1.4.C |

* + - 1. **photo:**

1. It shall be a type Varchar (13).
2. It shall contain the file name of photo with extension.
3. It shall not be repeatable.
4. It shall not be null.

|  |  |
| --- | --- |
| **Verification:** | **Observation:** |
| 3.1.1.5.A | The field ‘photo’ in the table ‘Employees’ has the specified characteristics, based on what is mentioned in the requirements. |
| 3.1.1.5.B |
| 3.1.1.5.C |
| 3.1.1.5.D |

* + - 1. **is\_active:**

1. It shall be a type Varchar (1).
2. It shall contain a ‘1’ or a ‘0’, the initial value shall be ‘0’. ‘1’ is for active employee and ‘0’ is for inactive employee.
3. It shall not be null.

|  |  |
| --- | --- |
| **Verification:** | **Observation:** |
| 3.1.1.6.A | The field ‘is\_active’ in the table ‘Employees’ has the specified characteristics, based on what is mentioned in the requirements. |
| 3.1.1.6.B |
| 3.1.1.6.C |

* + 1. **Table ‘Company’**, it shall contains the following parameters:

|  |  |
| --- | --- |
| **Verification:** | **Observation:** |
| 3.1.2. | The table ‘Company’ in the DB ‘ODTC\_Service’ has the fields that are specified, based on what is mentioned in the requirements. |

* + - 1. **company\_id:**

1. It shall be a type ‘int auto increment primary key’
2. It shall contain the number of comp}any, to relate it to the first table.
3. It shall not be null.

|  |  |
| --- | --- |
| **Verification:** | **Observation:** |
| 3.1.2.1.A | The field ‘company\_id’ in the table ‘Company’ has the specified characteristics, based on what is mentioned in the requirements. |
| 3.1.2.1.B |
| 3.1.2.1.C |

* + - 1. **name:**

1. It shall be a type Varchar (50).
2. It shall contains the company name.
3. It shall not be repeatable.
4. It shall not be null.

|  |  |
| --- | --- |
| **Verification:** | **Observation:** |
| 3.1.2.2.A | The field ‘name’ in the table ‘Company’ has the specified characteristics, based on what is mentioned in the requirements. |
| 3.1.2.2.B |
| 3.1.2.2.C |
| 3.1.2.2.D |

* + 1. **Table ‘Current\_menu’, it shall contains the following parameters:**

|  |  |
| --- | --- |
| **Verification:** | **Observation:** |
| 3.1.3. | The table ‘Current\_menu’ in the DB ‘ODTC\_Service’ has the fields that are specified, based on what is mentioned in the requirements. |

* + - 1. **menu\_id:**

1. It shall be a type ‘int auto increment primary key’.
2. It shall not be null.

|  |  |
| --- | --- |
| **Verification:** | **Observation:** |
| 3.1.3.1.A. | The field ‘menu\_id’ in the table ‘Current\_Menu’ has the specified characteristics, based on what is mentioned in the requirements. |
| 3.1.3.1.B. |

* + - 1. **employee\_id:**

1. It shall be a type ‘employee\_id’.
2. It shall contains the full name generated in database table Employees.
3. It shall not be null.

|  |  |
| --- | --- |
| **Verification:** | **Observation:** |
| 3.1.3.2.A. | The field ‘employee\_id’ in the table ‘Current\_Menu’ has the specified characteristics, based on what is mentioned in the requirements. |
| 3.1.3.2.B. |
| 3.1.3.2.C. |

* + - 1. **dish:**

1. It shall be a type Varchar (80).
2. It shall contain the dish that the employee selected.
3. It shall not be null.

|  |  |
| --- | --- |
| **Verification:** | **Observation:** |
| 3.1.3.3.A. | The field ‘dish’ in the table ‘Current\_Menu’ has the specified characteristics, based on what is mentioned in the requirements. |
| 3.1.3.3.B. |
| 3.1.3.3.C. |

* + - 1. **company\_id:**

1. It shall be a type ‘company\_id’.
2. It shall contains the company number generated in database table Company.
3. It shall not be null.

|  |  |
| --- | --- |
| **Verification:** | **Observation:** |
| 3.1.3.4.A. | The field ‘company\_id’ in the table ‘Current\_menu’ has the specified characteristics, based on what is mentioned in the requirements. |
| 3.1.3.4.B. |
| 3.1.3.4.C. |

* + - 1. **date:**

1. It shall be a type date.
2. It shall contain date in format yyyy-mm-ddthh-mm-ss.
3. It shall not be null.

|  |  |
| --- | --- |
| **Verification:** | **Observation:** |
| 3.1.3.5.A. | The field ‘date’ in the table ‘Current\_menu’ has the specified characteristics, based on what is mentioned in the requirements. |
| 3.1.3.5.B. |
| 3.1.3.5.C. |

* + - 1. **served:**

1. It shall be a type Varchar (1).
2. It shall contain if the employee was served or not.
3. It shall not be null.

|  |  |
| --- | --- |
| **Verification:** | **Observation:** |
| 3.1.3.6.A. | The field ‘served’ in the table ‘Current\_menu’ has the specified characteristics, based on what is mentioned in the requirements. |
| 3.1.3.6.B. |
| 3.1.3.6.C. |

# **Relate information**

* 1. **Badge to Employee:** To identify the employee, it shall match the number of his badge with the information on the table “Employees” of the DB project.
  2. **Employee to Menu:** To select the dish that the employee selected, the information of the employee (previews detected) shall match with table ‘Current\_Menu’ in the DB, where is the information of the Menu of the day.

|  |  |
| --- | --- |
| **Verification:** | **Observation:** |
| 4.1. | The script ‘grabInfo.py’ imports the function to read badge, the value returned is matched to the employee information with query's to the tables ‘Employees’ & ‘Company’ in the DB ‘OTDC\_Sercvice’. |
| 4.2. | The script ‘grabInfo.py’ matches the employee information, with query's to the table ‘Current\_Menu’ in the DB ‘OTDC\_Sercvice’. |

# **Energy button**

* 1. Python Script shall Turn on and Turn off raspberry, when button is pressed.
  2. It shall close all the programs before Raspberry is turn off to avoid damages in the files.
  3. It shall automatically run main program (GUI) of the BionicKitchen project, when Raspberry is turn on.
  4. Type of connection of the button shall be Pull Down.

# **Web page to upload menu**

* 1. The web page shall be mounted on raspberry.
  2. Shall be accessed of another computer on the local network.
  3. The name of the web page shall be ‘BK Service’.
  4. Between of name shall show the logo of ‘Obregon Technology Development Center’.
  5. Shall be written in HTML5 for text.
  6. Shall be written in CSS3 for design.

1. The page shall have a similar program design to ‘BionicKitchen’ GUI.
   1. Shall be written in PHP7 to upload the file. The script ‘upload.php’ to upload spreadsheet of the menu, with follow verifications:
2. It shall save in folder ‘uploads’.
3. The spreadsheet shall only can be upload each 14 days, if try upload a file early or after time this refused the file and show a warning in the web.
4. It shall only accepts XLSX format, if try upload another type of file this refused the file and show a warning in the web.
5. It shall only accepts spreadsheet with this name: “Encuesta\_de\_comedor”, if try upload a file with another file name this refused the file and show a warning in the web.
6. It shall not be greater than 20MB.
   1. Shall be written in JavaScript for AJAX.
   2. Shall be written in Python2.7 for execute validations.
   3. Shall only accepts XLSX format.
   4. Shall only can upload one document each two weeks.

# **Web page to manage employees**

* 1. The web page shall be mounted on raspberry.
  2. Shall be accessed of another computer on the local network.
  3. The name of the web page shall be ‘BK Management’.
  4. Between of name shall show the logo of ‘Obregon Technology Development Center’.
  5. Shall be written in HTML5 for text.
  6. Shall be written in CSS3 for design.

1. The page shall be a modern design.
   1. Shall be written in PHP7.
      1. It shall connecting to database
      2. It shall update information of employee of the database.
      3. It shall eliminate employee of the database.
      4. It shall register employee of the database.
      5. It shall search employee of the database.
      6. It shall show information in page of employee.
      7. It shall verification all fields of placeholders.
   2. Shall be written in JavaScript for AJAX.
   3. Shall be written in python for execute validations.
   4. Shall be able to register employees to the DB. To register an employee shall request this information:
      1. Full name:
2. Shall be written correctly
3. Shall respect upper case and lower case
   * 1. Company:
4. Shall be written correctly
5. Shall respect upper case and lower case
6. Shall verify that the company is in the DB, either shall send warning.
   * 1. Badge (Employee badge):
7. Shall not be repeated, either shall send warning.
8. Shall be provided by the RFID reader
   * 1. Photo:
9. The photo shall be grab of database.
   1. Shall be able to modify employee information of DB. To modify an employee shall search employee and change the necessary information:
      1. Full name:
10. Shall be written correctly
11. Shall respect upper case and lower case
    * 1. Company:
12. Shall be written correctly
13. Shall respect upper case and lower case
14. Shall verify that the company is in the DB, either shall send warning.
    * 1. Badge (Employee badge):
15. Shall not be repeated, either shall send warning.
16. Shall be provided by the RFID reader
    * 1. Photo:
17. The photo shall be grab of the database.
    1. Shall be able to eliminate employee of DB
    2. Employee need to be searched before
    3. If employee information is correct, a button shall eliminate the employee of DB
    4. Shall send a warning confirmation before employee is deleted.
    5. Shall be able to search employees of the DB
       1. Search shall be by the employees name
       2. If search is successful, shall display employees information.
       3. If search is failed, shall display that the employee isn’t in the DB.
    6. Shall send a warning to confirm that the information is correct, before is uploaded.

# **Convert spreadsheet to database**

|  |  |
| --- | --- |
| **Verification:** | **Observation:** |
| 8. | Initial condition:   * Load the ‘ServiceUpdate.html’ (Web page to upload excel document)   Test action:   * Upload the respective excel document * NOTE: The document only can be uploaded specific days, and needs to be an specific document also.   Expected result:   * The excel document document is uploaded, then his information is processed and inserted to the DB |

* 1. Excel document ‘Encuesta\_de\_Comedor.xlsx’ shall be in the folder ‘uploads’.
  2. The script shall iterates in undefined columns and 11 rows of ‘Encuesta\_de\_Comedor.xlsx’, to extract name of employee, dish number and dish name according to rows of date.
  3. Shall connect to ‘OTDC\_Service’ to grab the ‘employee\_id’ and ‘company id’ related with the respective employee.
  4. Information selected shall get into a tuple. The tuple shall contain ‘employee\_id, dish, company\_id, date, served’.
  5. Shall connect to ‘OTDC\_Service’ in table ‘Current\_menu’ and add tuple with information.

|  |  |
| --- | --- |
| **Verification:** | **Observations:** |
| 8.1. | An excel document named ‘Encuesta\_de\_Comedor’ is storage in the folder uploads (file update through the web page) |
| 8.2. | The script ‘uploadExcel.py’ iterates with multiple condition sentences the excel document rows to extract the required information. |
| 8.3. | The script connect to the DB and with a query grabs the information required |
| 8.4. | The information extract from the excel document it is adding to a tuple |
| 8.5. | The script connects to the DB and with a query inserts the information in the tuples to the table ‘Current\_menu’ |

# **Service buttons**

Shall wait an answer of served o canceled buttons, either doesn´t continue reading badges.

* + 1. Served button:
  1. Shall indicate that the employee was served.
  2. Shall permit the RFID Module to read another badge.
  3. Shall update the served status in the database.
  4. Shall be color green
     1. Canceled button:
  5. Shall indicate that the employee wasn’t served.
  6. Shall permit the RFID Module to read another badge.
  7. Shall be color yellow (TBD)
     1. Close program button:
  8. Shall close the main program (GUI).
  9. Shall save information, for avoid loss of information.
  10. Shall be color red (TBD)

# **Export table of DB to spreadsheet**

The spreadsheet ‘Current\_menu.xlsx’ shall export daily and have the following columns:

|  |  |
| --- | --- |
| **Verification:** | **Observation:** |
| 10. | An excel document named ‘Current\_menu’ is create when the program (‘GUI’) is closed. It is not daily as such. |

* 1. Employee´s name, it shall be only full name.
  2. Employee´s company, it shall be only name of company.
  3. Dish, it shall be the id of dish and name of dish in parenthesis.
  4. Date, it shall only contain date in format yyyy-mm-ddthh-mm-ss.
  5. Status, it shall only contain a ‘1’ for served and ‘0’ for not served.

|  |  |
| --- | --- |
| **Verification:** | **Observations:** |
| 10.1. | The script in charge of this process (‘toXLSX.py’) grabs all the information form the table ‘Current\_menu’ of the DB (‘OTDC\_Service’), and insert it in the corresponding fields mentioned in the requirement. |
| 10.2. |
| 10.3. |
| 10.4. |
| 10.5. | Initial condition:   * Startup Raspberry (GUI runs automatically) * Press yellow button (canceled button) so the RFID starts to read   Test action:   * Swipe badge of employees * Press the served button for some employees   Expected result:  The employees that were served, were reflected with a 1 in the field 'status' and those that were not served were reflected with a 0 |