Report for IoT Laboratory

30.05.2020

Cannur Kartum 251585

Laboratory Teacher; Natalia Piórkowska

2. Table of Contents:

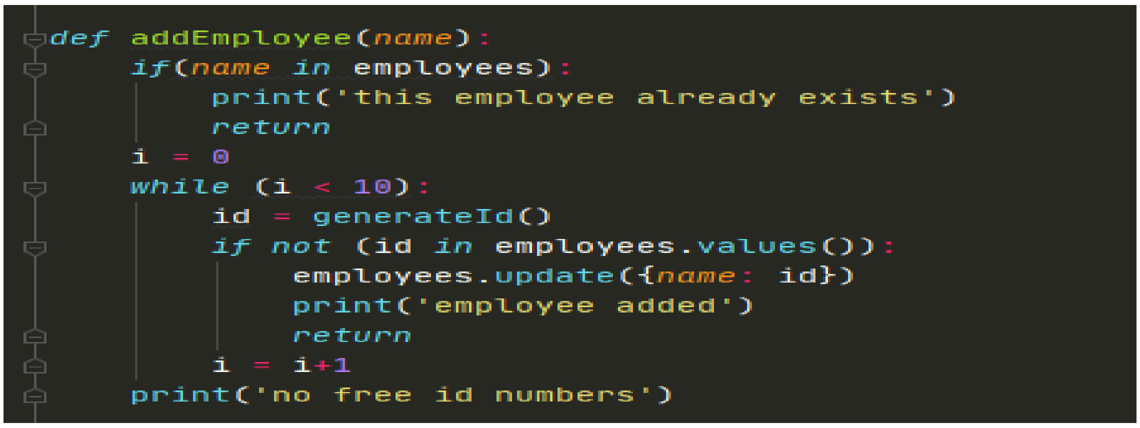
Laboratory 5	Creating a Basic RIFD Card Reader System
Laboratory 6	MQTT Security Protocol for System
Laboratory 7	Encryption of application messages andcontrol packets sent over the network within MQTT
Laboratory 8	In particular, introduction to authentication and authorization access to MQTT
Laboratory 9	Made a Summary for 5-6-7-8

3. Design requirements;

The system should behave like this:

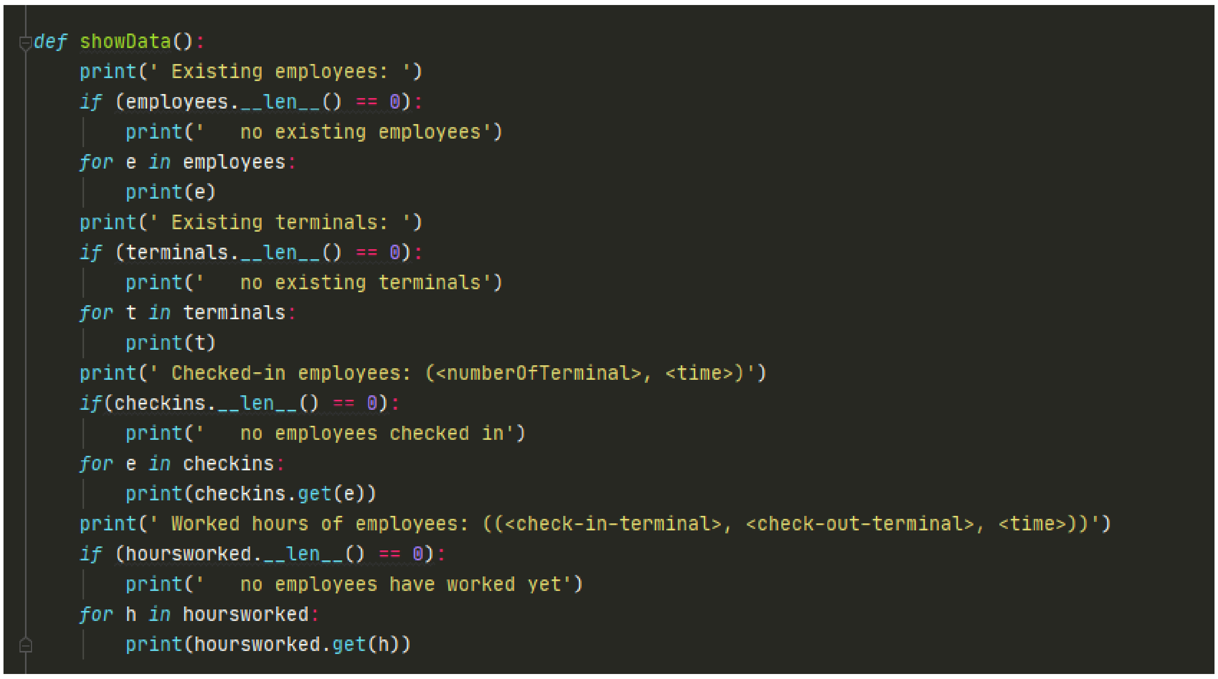
- Allows to create a Employee
- Allows to create a Terminal
- Allows to checkin a existing Employee
- Allows to checkout a existing Employee
- Allows to Show Employee working hours

Basic functional;



```
def addEmployee(name):
    if(name in employees):
        print('this employee already exists')
        return
    i = 0
    while (i < 10):
        id = generateId()
        if not (id in employees.values()):
            employees.update({name: id})
            print('employee added')
            return
        i = i+1
    print('no free id numbers')
```

Basic non-functional;

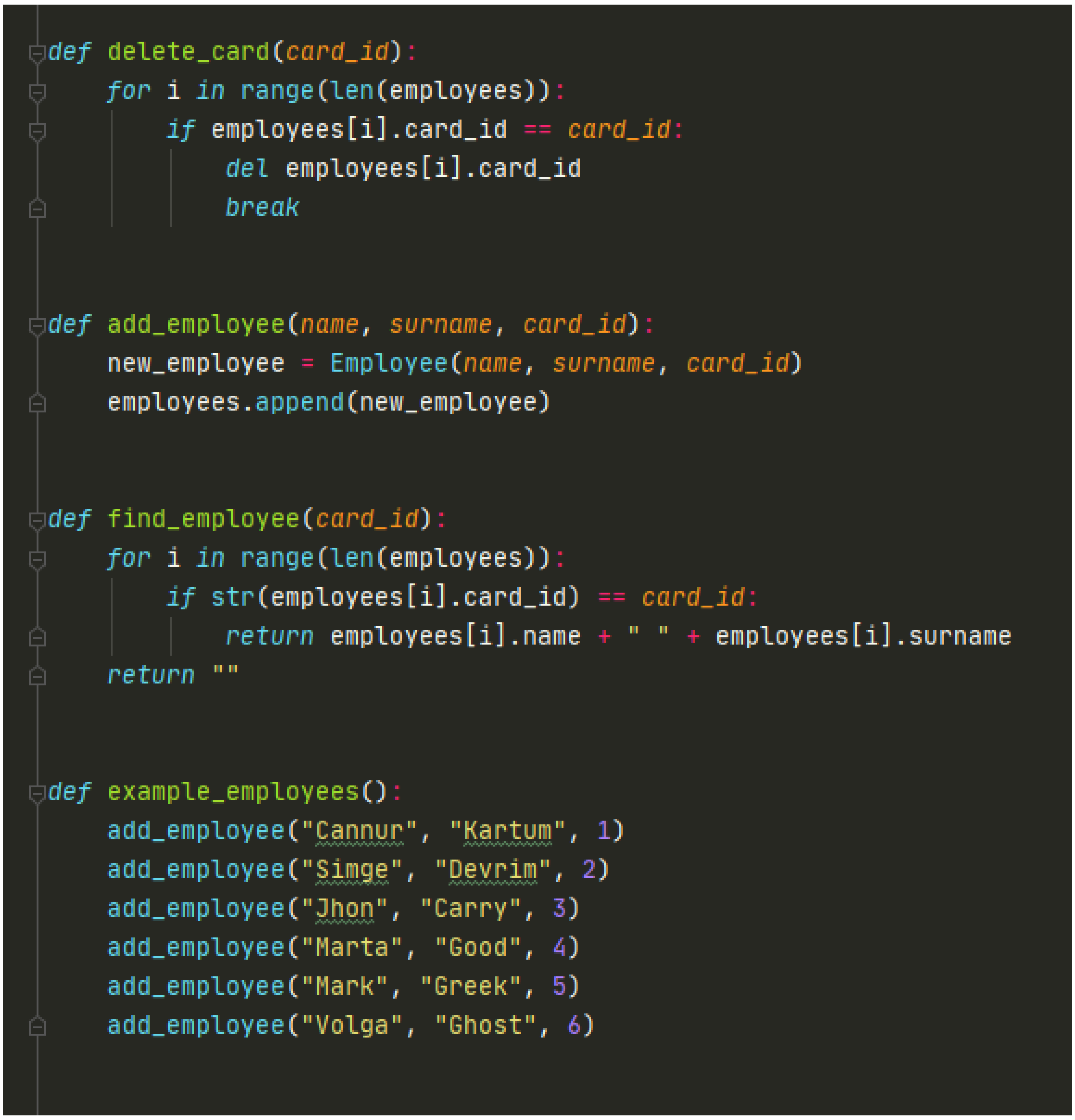


```
def showData():
    print(' Existing employees: ')
    if (employees.__len__() == 0):
        print('   no existing employees')
    for e in employees:
        print(e)
    print(' Existing terminals: ')
    if (terminals.__len__() == 0):
        print('   no existing terminals')
    for t in terminals:
        print(t)
    print(' Checked-in employees: (<numberOfTerminal>, <time>)')
    if(checkins.__len__() == 0):
        print('   no employees checked in')
    for e in checkins:
        print(checkins.get(e))
    print(' Worked hours of employees: ((<check-in-terminal>, <check-out-terminal>, <time>))')
    if (hoursworked.__len__() == 0):
        print('   no employees have worked yet')
    for h in hoursworked:
        print(hoursworked.get(h))
```

4. Description of system architecture

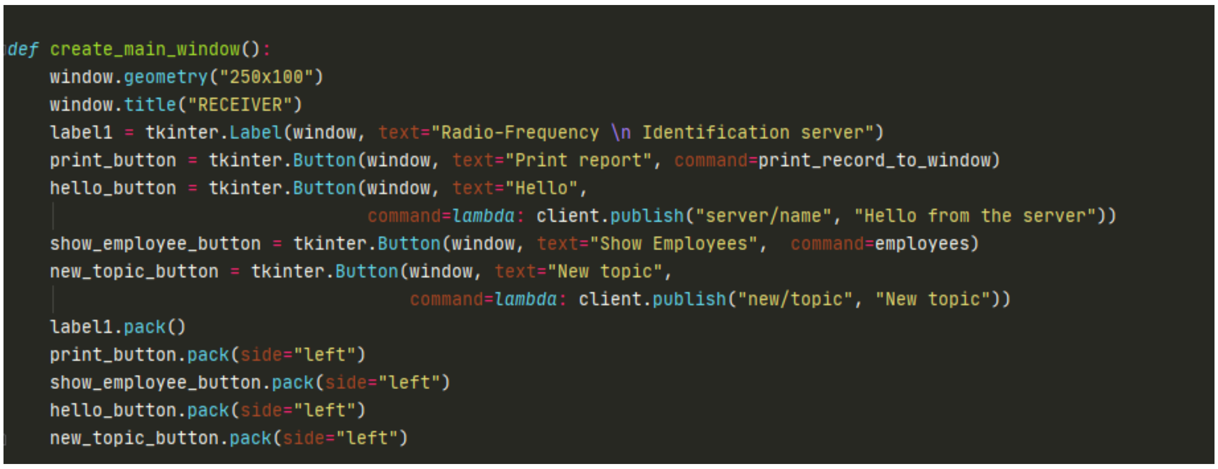
The architecture of the application is complicated but not so hard to do we need to set up database for our employees and we need to show employees in screen in Lab5 but later we need to show this requirements via server/client subscribe to topic. The network architecture is very important because we are using MQTT and its required network settings. We need to setup localhost names for broker and for make it secure we need to create basic SSL for our network connection. When we create SSL connection we need to encrypt passwords and after that we are creating aclconfig file for our topics in MQTT and we are using this file for crypting our topic for made it more secure.

5. Description of implementation and solutions used



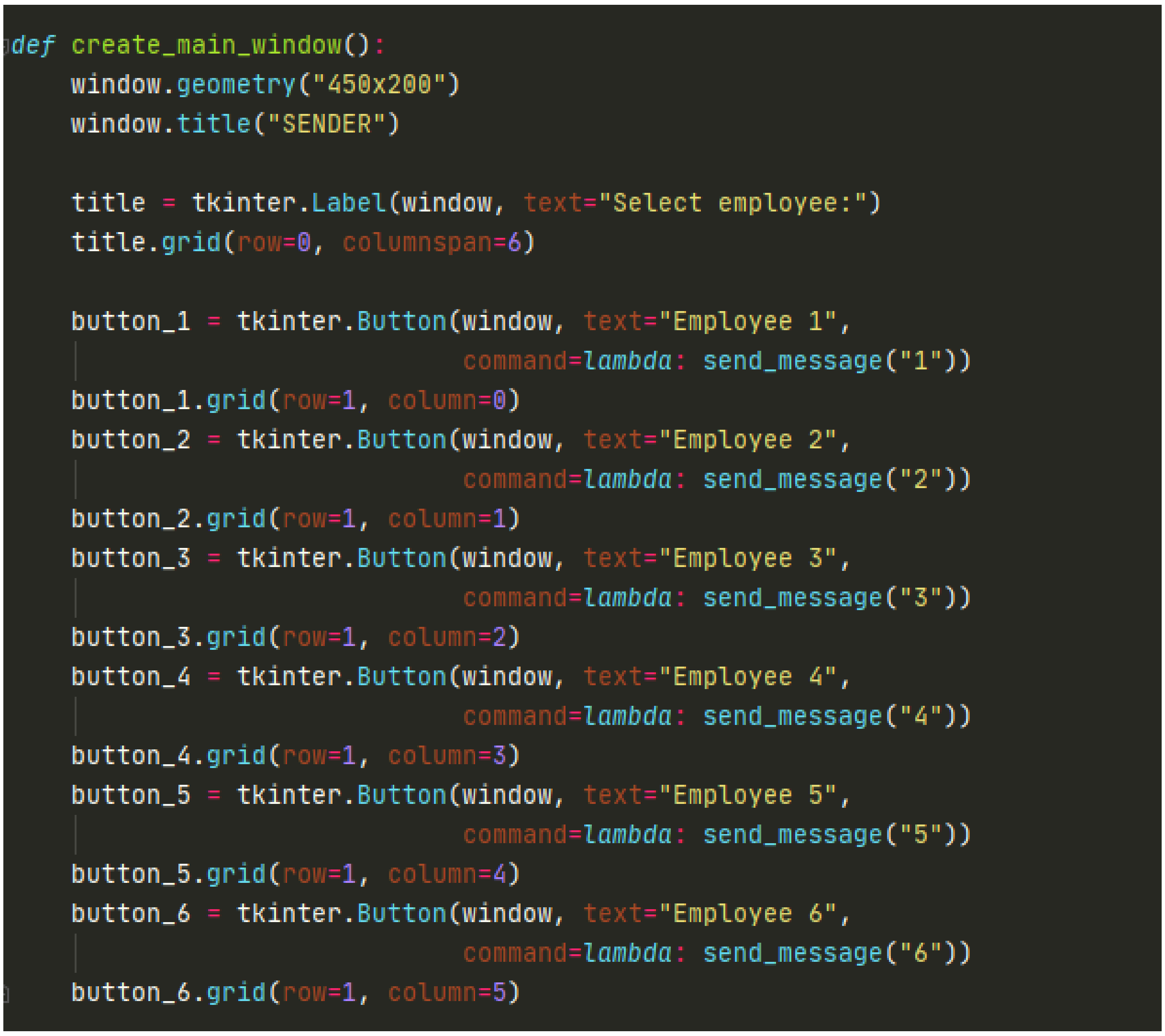
```python
def delete_card(card_id):
    for i in range(len(employees)):
        if employees[i].card_id == card_id:
            del employees[i].card_id
            break


def add_employee(name, surname, card_id):
    new_employee = Employee(name, surname, card_id)
    employees.append(new_employee)


def find_employee(card_id):
    for i in range(len(employees)):
        if str(employees[i].card_id) == card_id:
            return employees[i].name + " " + employees[i].surname
    return ""


def example_employees():
    add_employee("Cannur", "Kartum", 1)
    add_employee("Simge", "Devrim", 2)
    add_employee("Jhon", "Carry", 3)
    add_employee("Marta", "Good", 4)
    add_employee("Mark", "Greek", 5)
    add_employee("Volga", "Ghost", 6)
```

This is the code for creating example employees and basic configurations for employee, card settings.



```python
def create_main_window():
    window.geometry("250x100")
    window.title("RECEIVER")
    label1 = tkinter.Label(window, text="Radio-Frequency \n Identification server")
    print_button = tkinter.Button(window, text="Print report", command=print_record_to_window)
    hello_button = tkinter.Button(window, text="Hello",
                                  command=lambda: client.publish("server/name", "Hello from the server"))
    show_employee_button = tkinter.Button(window, text="Show Employees",  command=employees)
    new_topic_button = tkinter.Button(window, text="New topic",
                                      command=lambda: client.publish("new/topic", "New topic"))
    label1.pack()
    print_button.pack(side="left")
    show_employee_button.pack(side="left")
    hello_button.pack(side="left")
    new_topic_button.pack(side="left")
```

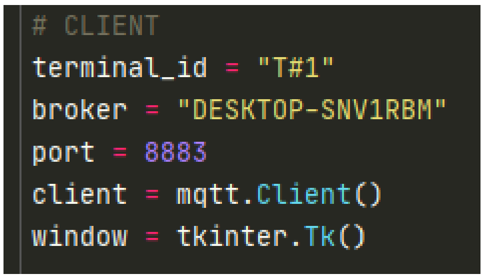
We are using Tkinter for creating our windows GUI. We are showing here our server.



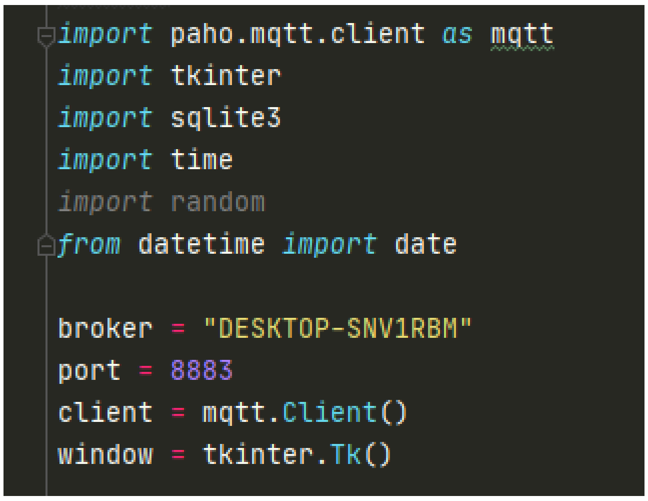
```python
def create_main_window():
    window.geometry("450x200")
    window.title("SENDER")

    title = tkinter.Label(window, text="Select employee:")
    title.grid(row=0, columnspan=6)

    button_1 = tkinter.Button(window, text="Employee 1",
                              command=lambda: send_message("1"))
    button_1.grid(row=1, column=0)
    button_2 = tkinter.Button(window, text="Employee 2",
                              command=lambda: send_message("2"))
    button_2.grid(row=1, column=1)
    button_3 = tkinter.Button(window, text="Employee 3",
                              command=lambda: send_message("3"))
    button_3.grid(row=1, column=2)
    button_4 = tkinter.Button(window, text="Employee 4",
                              command=lambda: send_message("4"))
    button_4.grid(row=1, column=3)
    button_5 = tkinter.Button(window, text="Employee 5",
                              command=lambda: send_message("5"))
    button_5.grid(row=1, column=4)
    button_6 = tkinter.Button(window, text="Employee 6",
                              command=lambda: send_message("6"))
    button_6.grid(row=1, column=5)
```

And here for our client windows GUI.

For MQTT implementation;



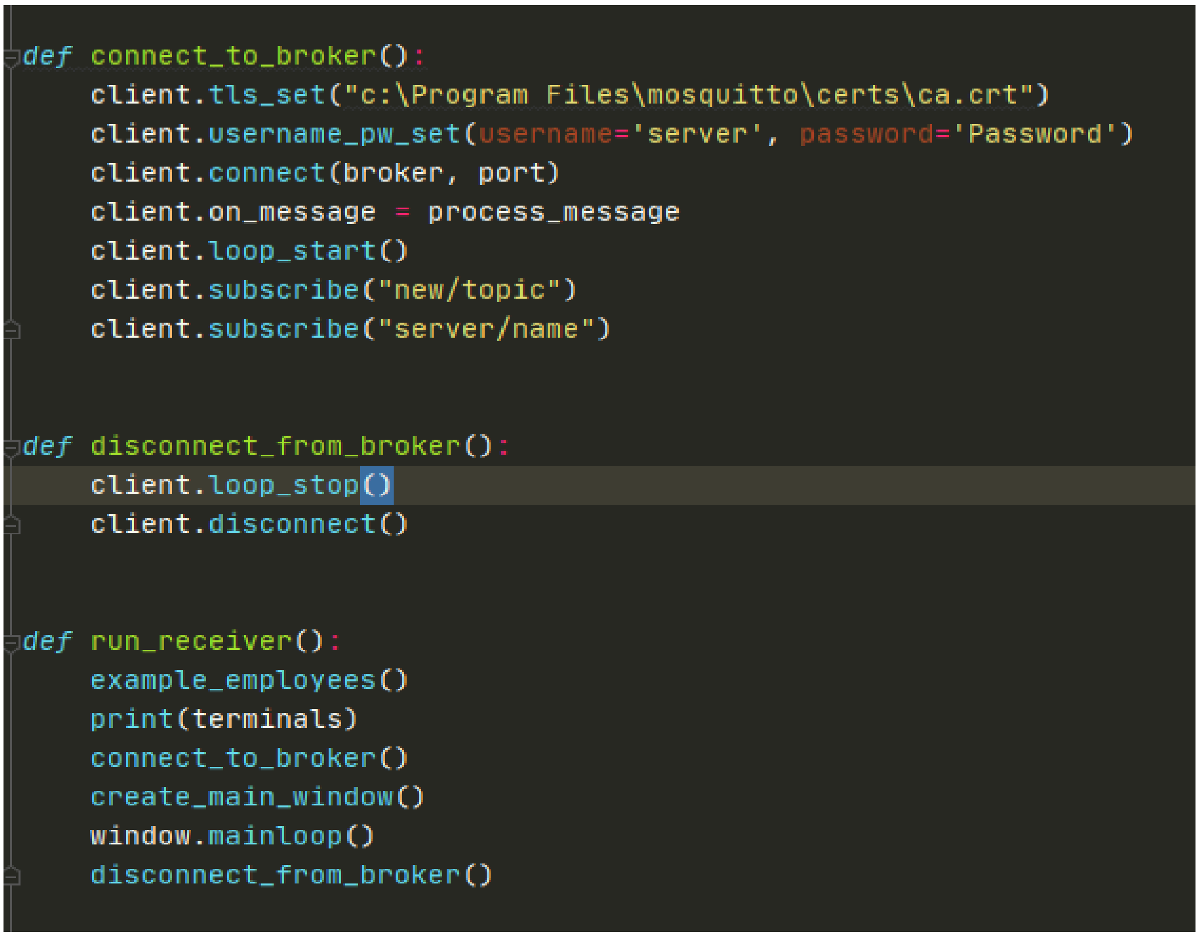
```
# CLIENT
terminal_id = "T#1"
broker = "DESKTOP-SNV1RBM"
port = 8883
client = mqtt.Client()
window = tkinter.Tk()
```



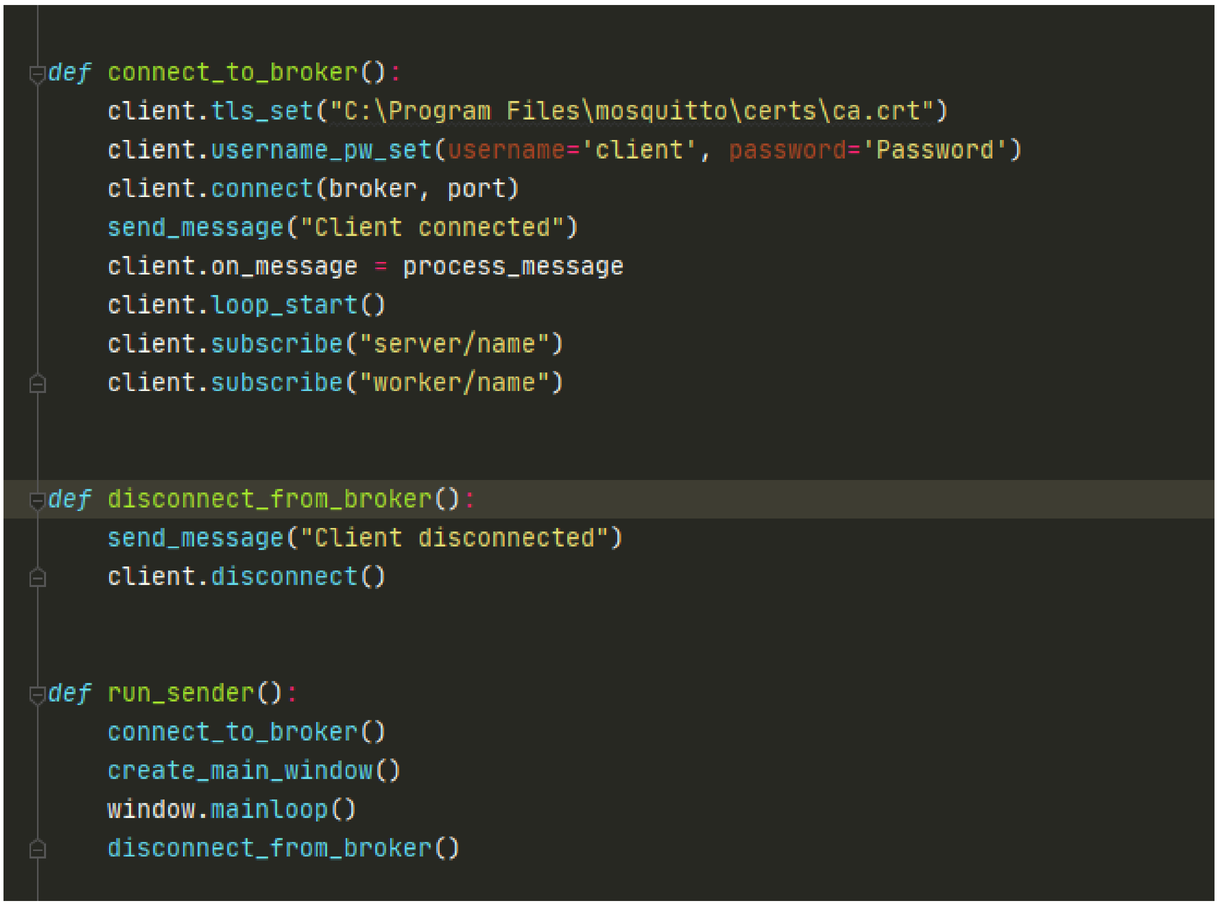
```
import paho.mqtt.client as mqtt
import tkinter
import sqlite3
import time
import random
from datetime import date

broker = "DESKTOP-SNV1RBM"
port = 8883
client = mqtt.Client()
window = tkinter.Tk()
```

We need to setup our broker for mosquitto settings. So in here we are creating our server and client.

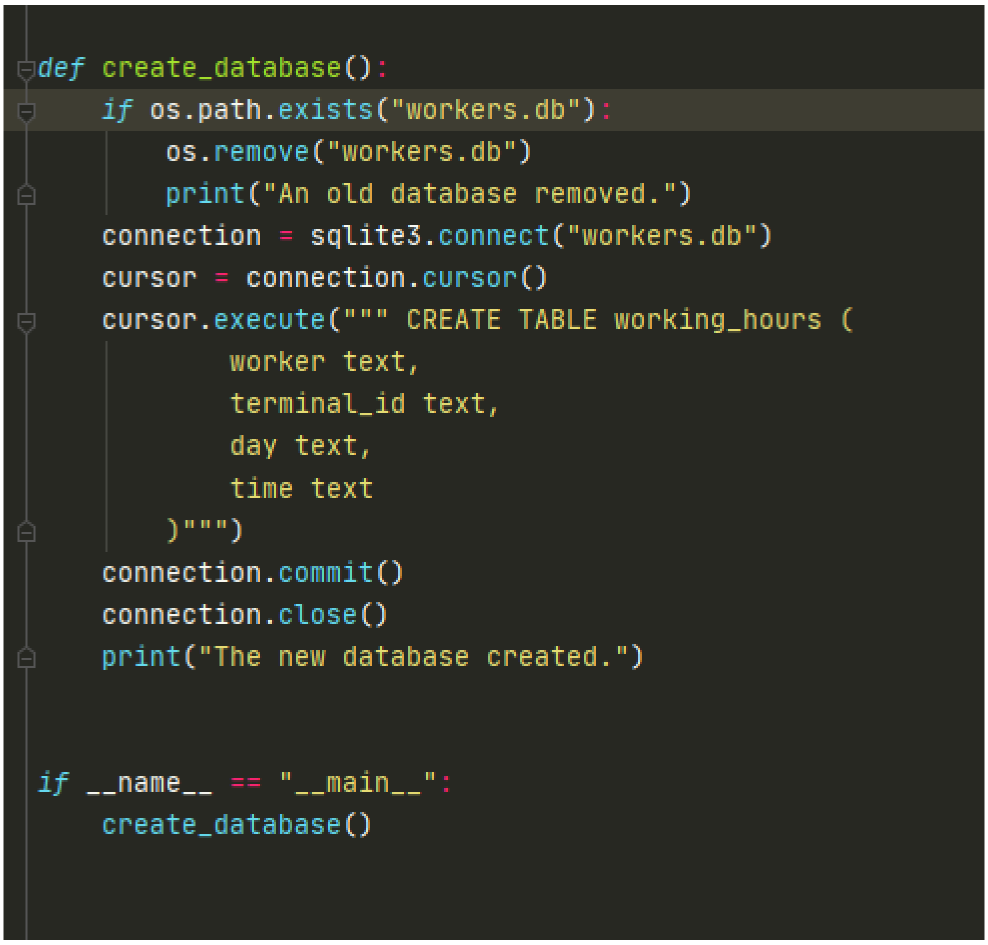
So in this part, we are connect and disconnect to our broker. And as its total part for our labs. There is also encryption and authentication part. Which is in line 3, we are set our password and username for SERVER and in line 2, we are giving path to the SSL certificate for security. And I added next picture for CLIENT and also we are setting pass and username and also path for SSL certificate. Because we want end-to-end security here. And also for authentication part we have subscribe part for SERVER/CLIENT. Both sides subscribe TOPICS in acl.config file.



```
def connect_to_broker():
    client.tls_set("c:\Program Files\mosquitto\certs\ca.crt")
    client.username_pw_set(username='server', password='Password')
    client.connect(broker, port)
    client.on_message = process_message
    client.loop_start()
    client.subscribe("new/topic")
    client.subscribe("server/name")


def disconnect_from_broker():
    client.loop_stop()
    client.disconnect()


def run_receiver():
    example_employees()
    print(terminals)
    connect_to_broker()
    create_main_window()
    window.mainloop()
    disconnect_from_broker()
```



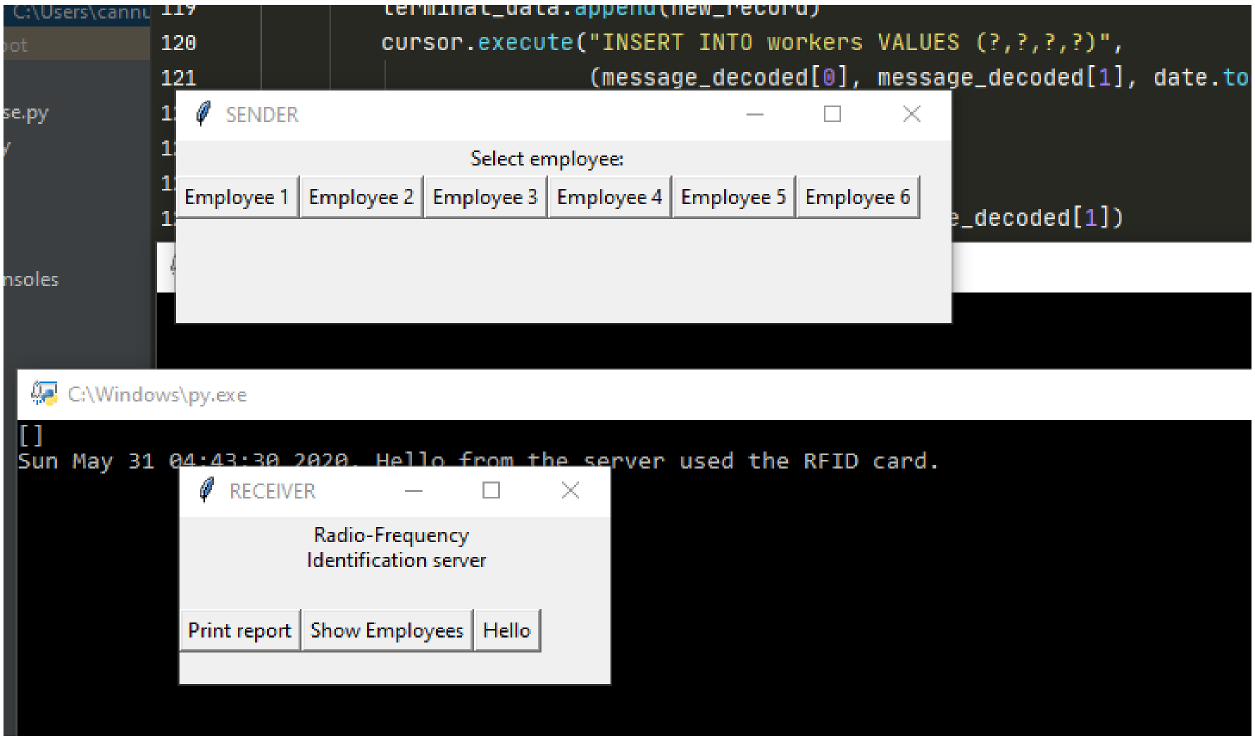
```
def connect_to_broker():
    client.tls_set("C:\Program Files\mosquitto\certs\ca.crt")
    client.username_pw_set(username='client', password='Password')
    client.connect(broker, port)
    send_message("Client connected")
    client.on_message = process_message
    client.loop_start()
    client.subscribe("server/name")
    client.subscribe("worker/name")


def disconnect_from_broker():
    send_message("Client disconnected")
    client.disconnect()


def run_sender():
    connect_to_broker()
    create_main_window()
    window.mainloop()
    disconnect_from_broker()
```



```python
def create_database():
    if os.path.exists("workers.db"):
        os.remove("workers.db")
        print("An old database removed.")
    connection = sqlite3.connect("workers.db")
    cursor = connection.cursor()
    cursor.execute(""" CREATE TABLE working_hours (
            worker text,
            terminal_id text,
            day text,
            time text
        )""")
    connection.commit()
    connection.close()
    print("The new database created.")


if __name__ == "__main__":
    create_database()
```

And another important element is Database file which is we are using SQLite3 for creating Database for our Laboratory.

6. Description of operations and presentation of the interface



```
        terminal_data.append(new_record)
        cursor.execute("INSERT INTO workers VALUES (?,?,?,?)",
                       (message_decoded[0], message_decoded[1], date.to
                                                         e_decoded[1])
```

```
C:\Windows\py.exe
[]
Sun May 31 04:43:30 2020, Hello from the server used the RFID card.
```

Its easy to install our application and run it. We just need to change broker = “localhost” part.

And this is how it looks like.

7. Summary

In this Laboratory’s we learned how RIFD Card System looks like how it suppose to be done and how we are implementing and creating a database. We have seen how safe systems are and what precautions are taken. How to setup MQTT Protocol and how to encrypt passwords.

In laboratory’s the most difficult part setup to broker, SSL certificates and create acl file for TOPICS. In our computer’s its hard to see informations about network.