Butler

CPT Trevano
Sezione informatica
Anno 2020/21

Filippo Zinetti
17.03.2022

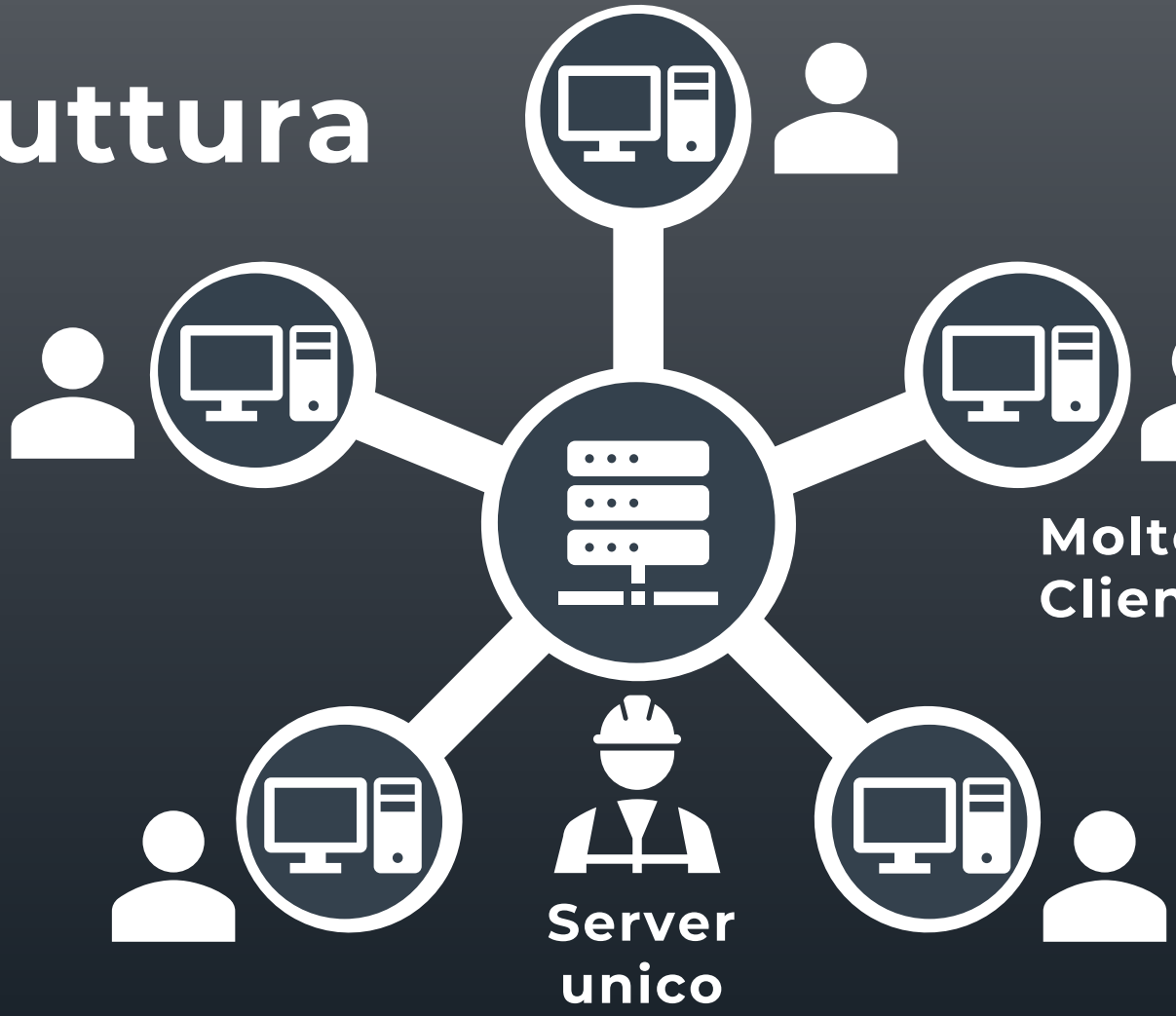
Indice

- Contesto
- Struttura
- Risultato
- Conclusioni

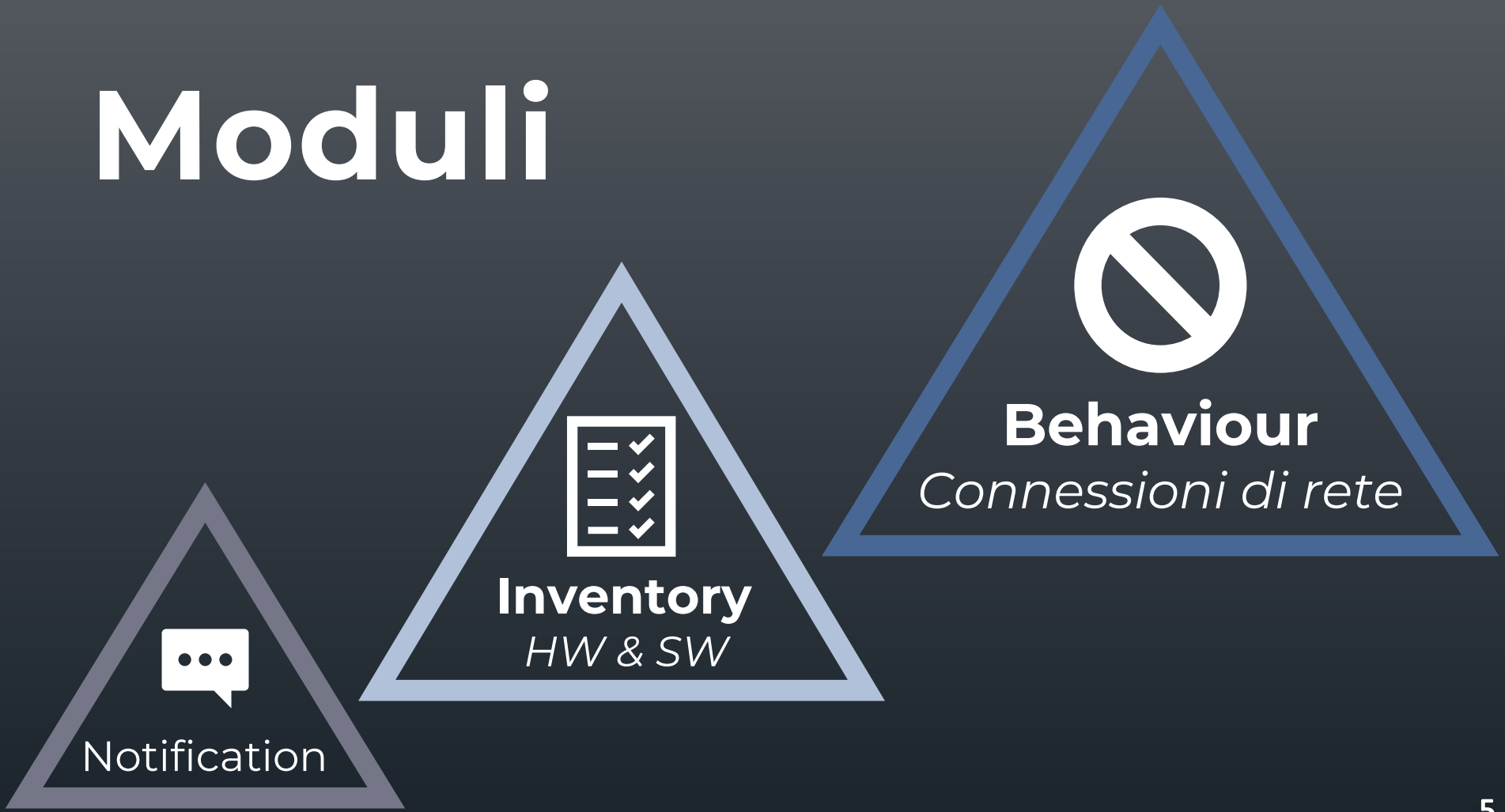
Contesto

- Progetto 2° semestre → **LPI**
- Sistema di invio notifiche «Butler»
- Messaggi di allerta + 2 moduli

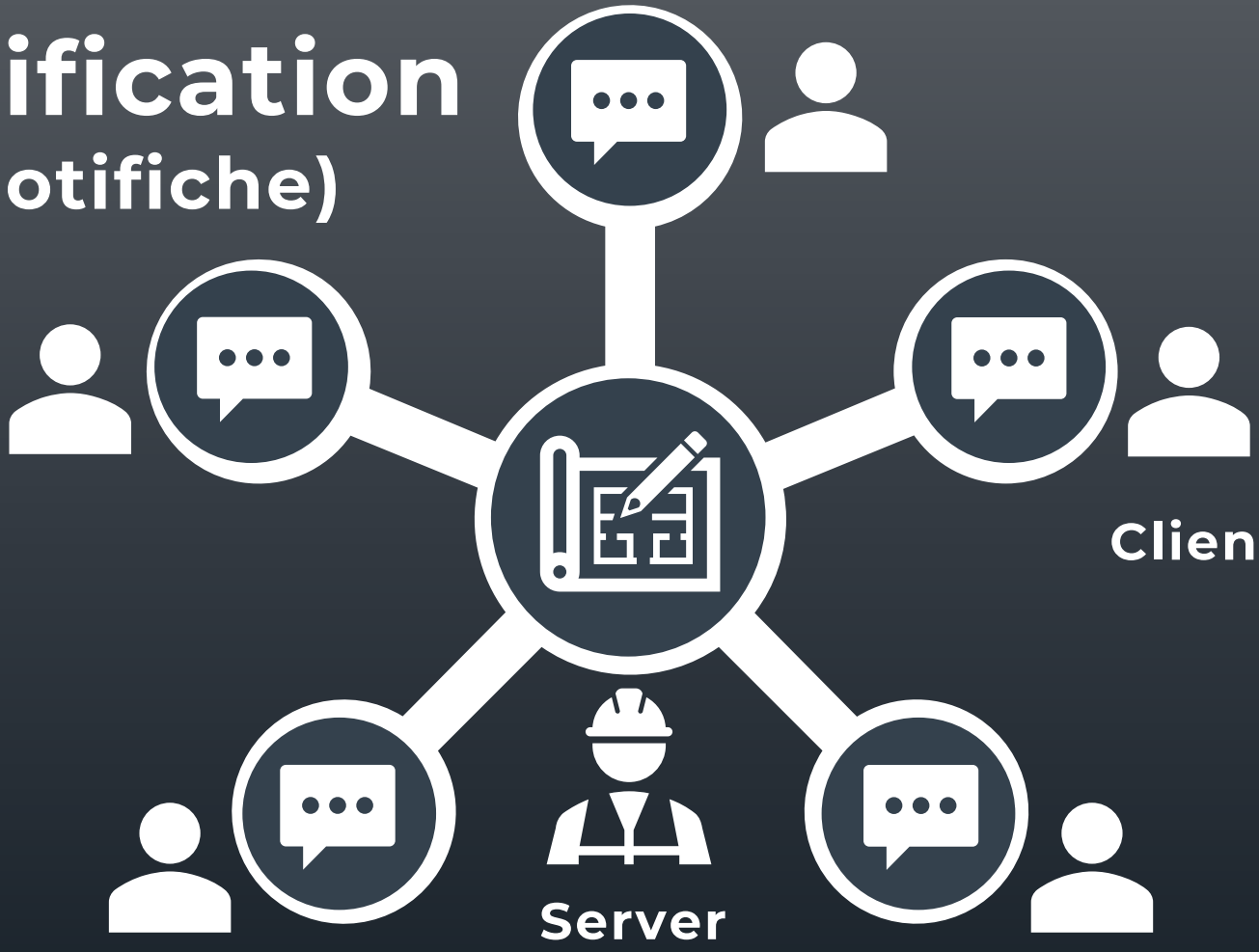
Struttura



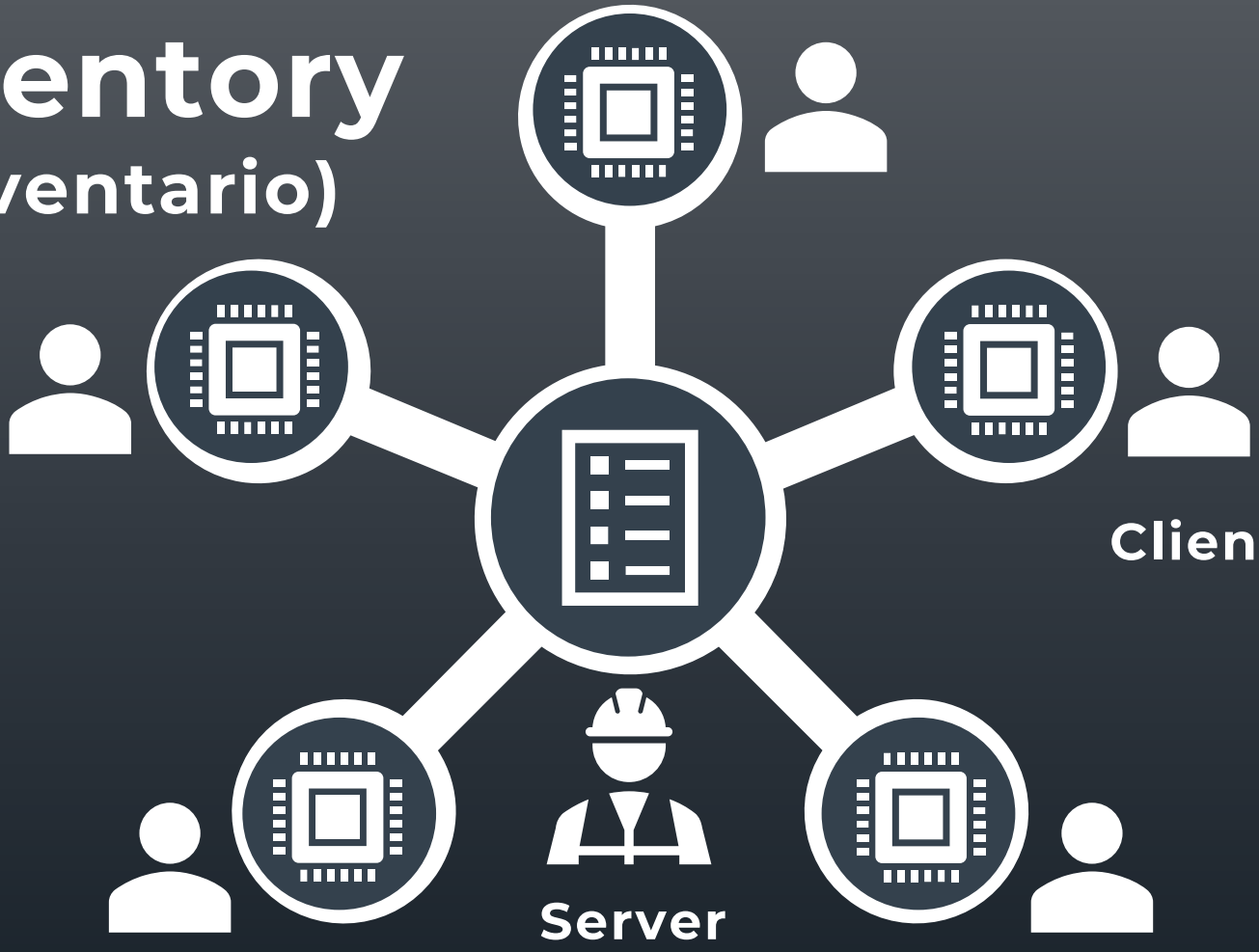
Moduli


Behaviour
Connessioni di rete
Inventory
HW & SW
Notification

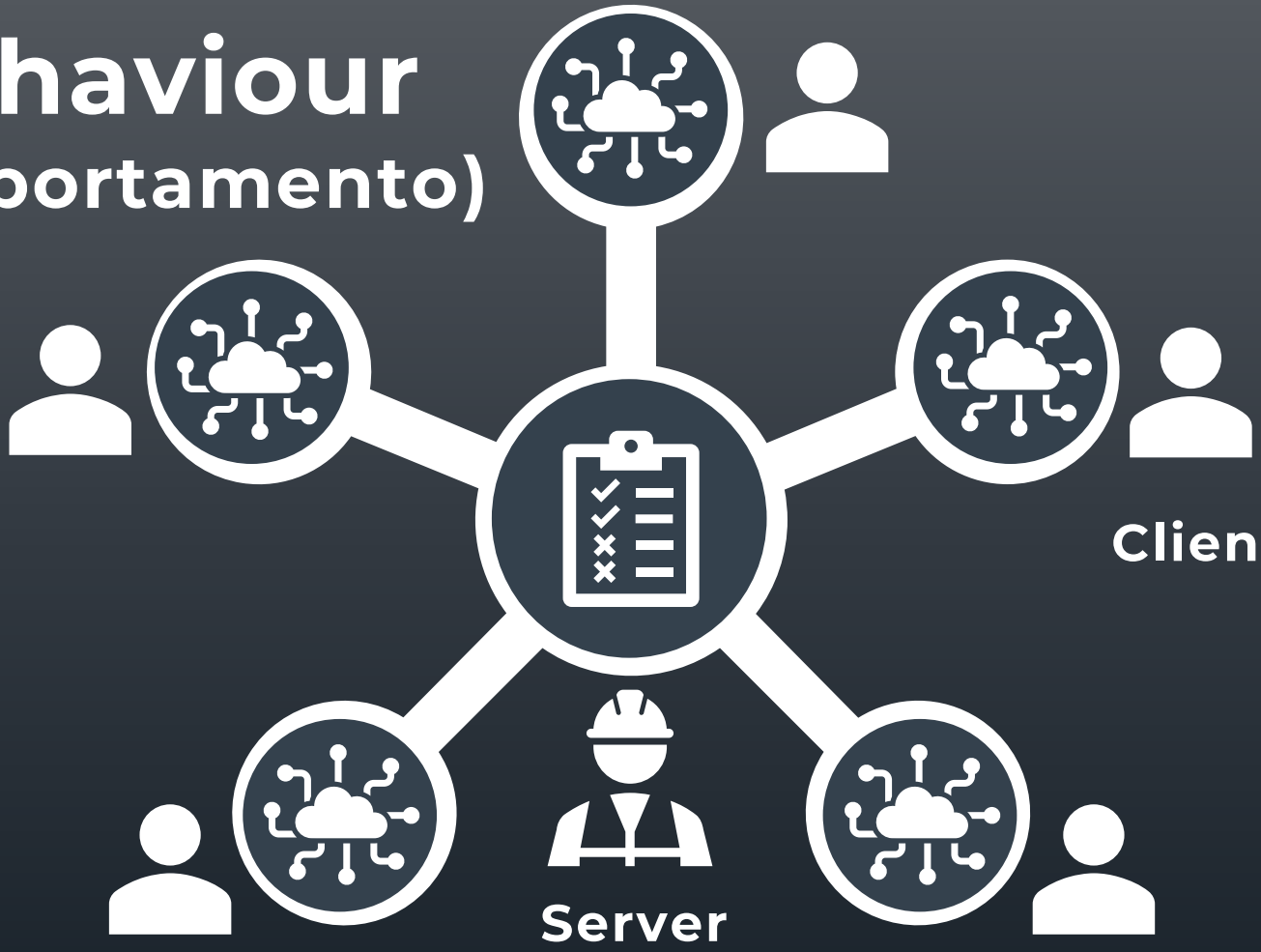
Notification
(notifiche)



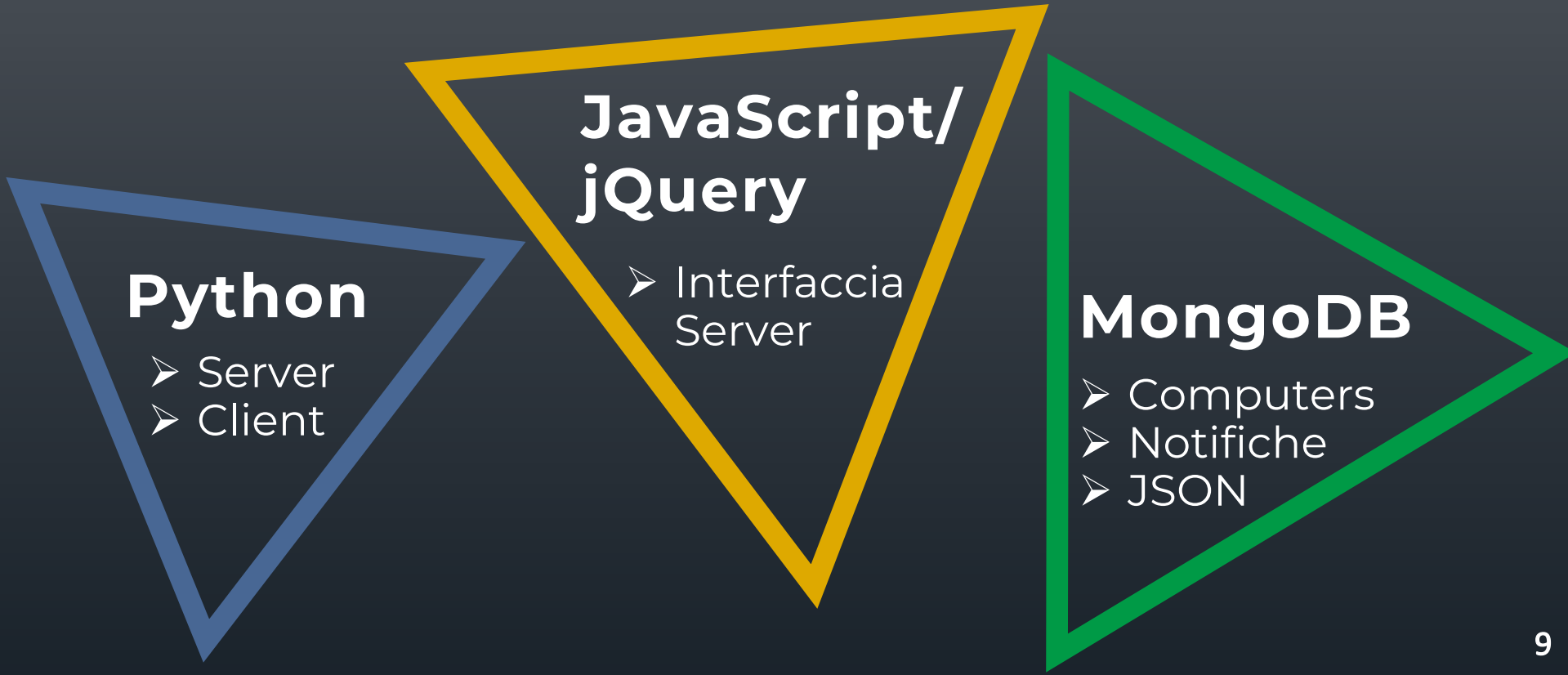
Inventory
(inventario)



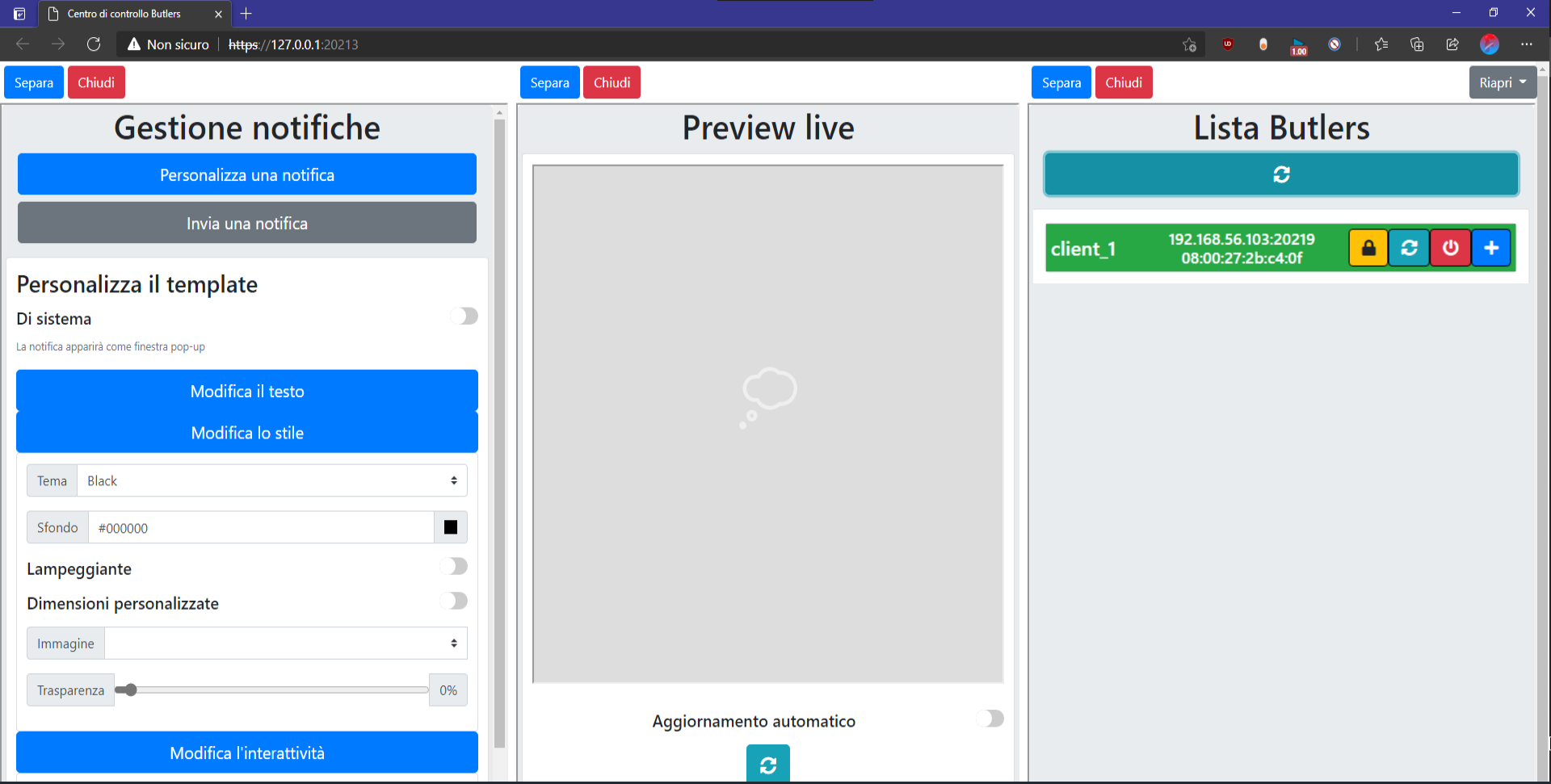
Behaviour
(comportamento)



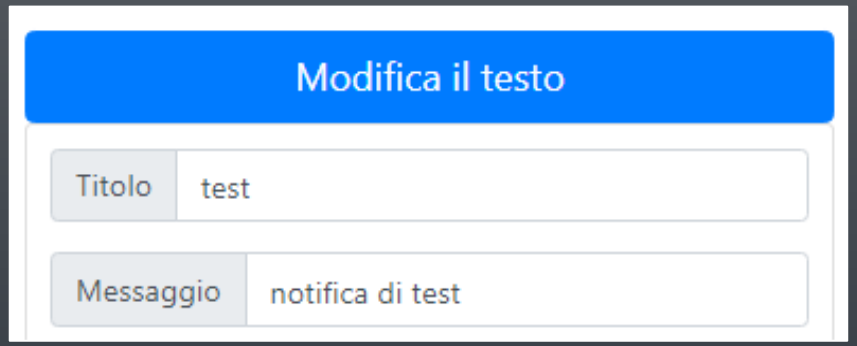
Strumenti principali

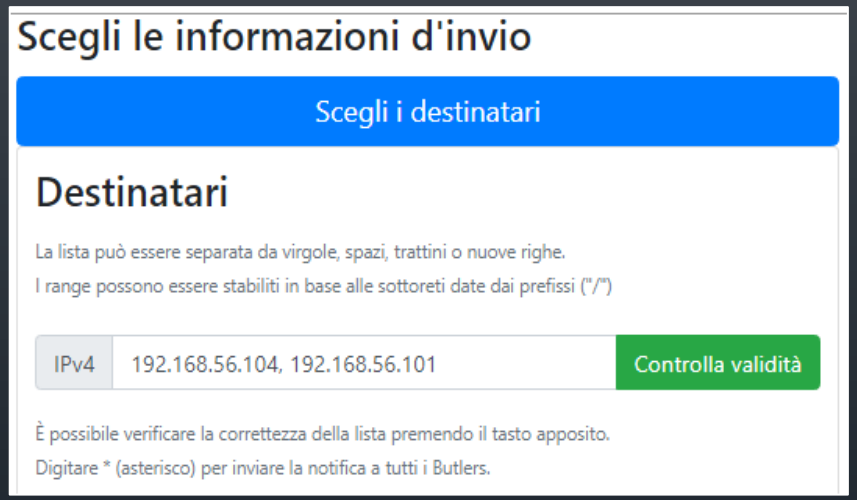


Risultato



Interfaccia server



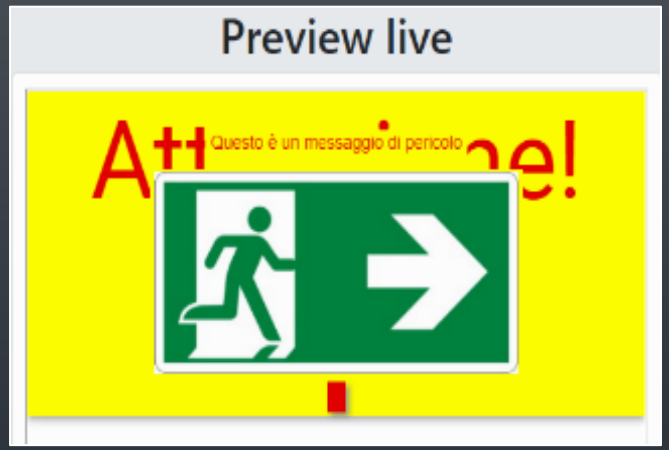


Gestione notifiche

Parametri notifiche

- Nome
- **Pop-up/di sistema**
- Titolo, messaggio*
- Colore, dimensione font
- Colore, velocità lampeggiamento testo
- **Programma/comando script**
- **File script**
- **Immagine**

- Possibilità di spostamento
- Possibilità di chiusura
- Precedenza totale su altre finestre
- Pulsante con testo
- Timer di chiusura automatica
- Colore di sfondo
- Colore, velocità lampeggiamento sfondo
- Altezza+larghezza / automatico



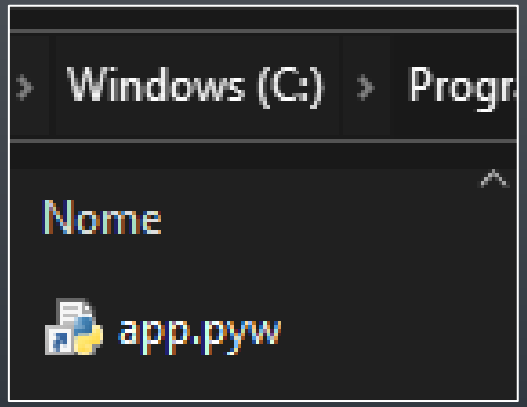
Preview web

Test sul server

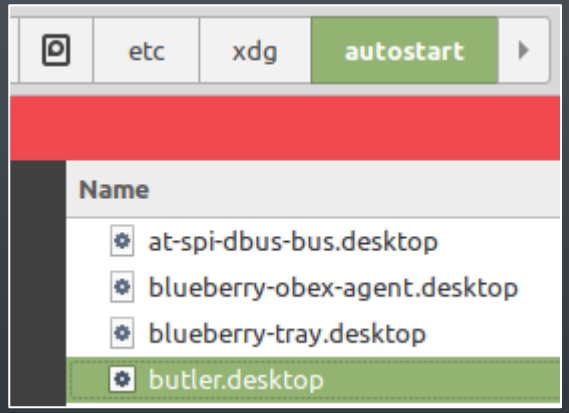
Pop up Butler

Coerenza notifiche

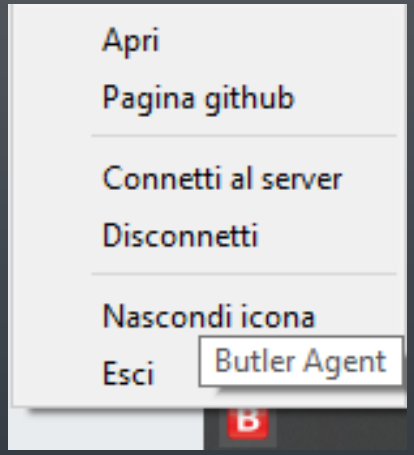
Windows

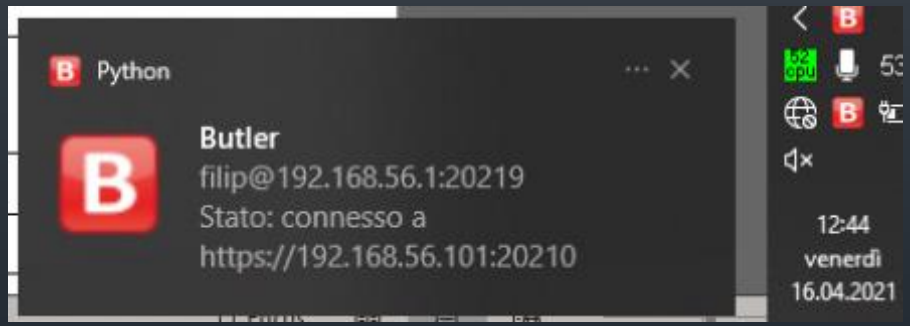


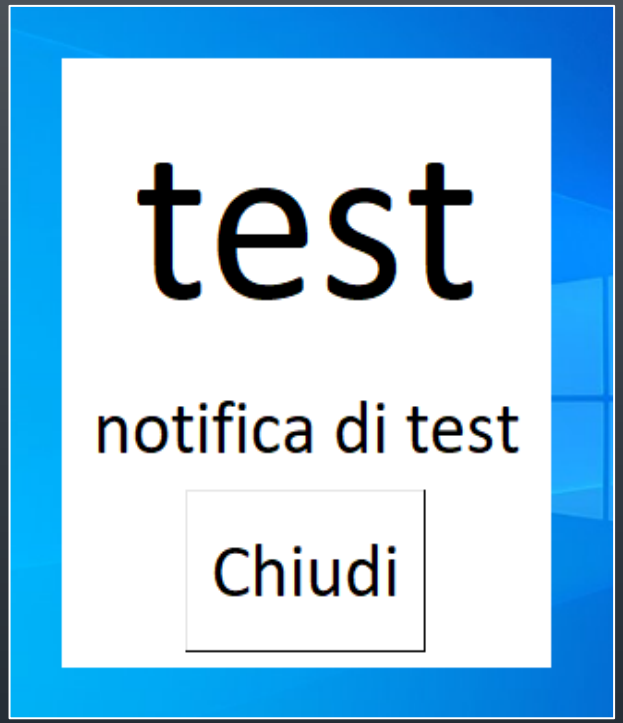
Linux



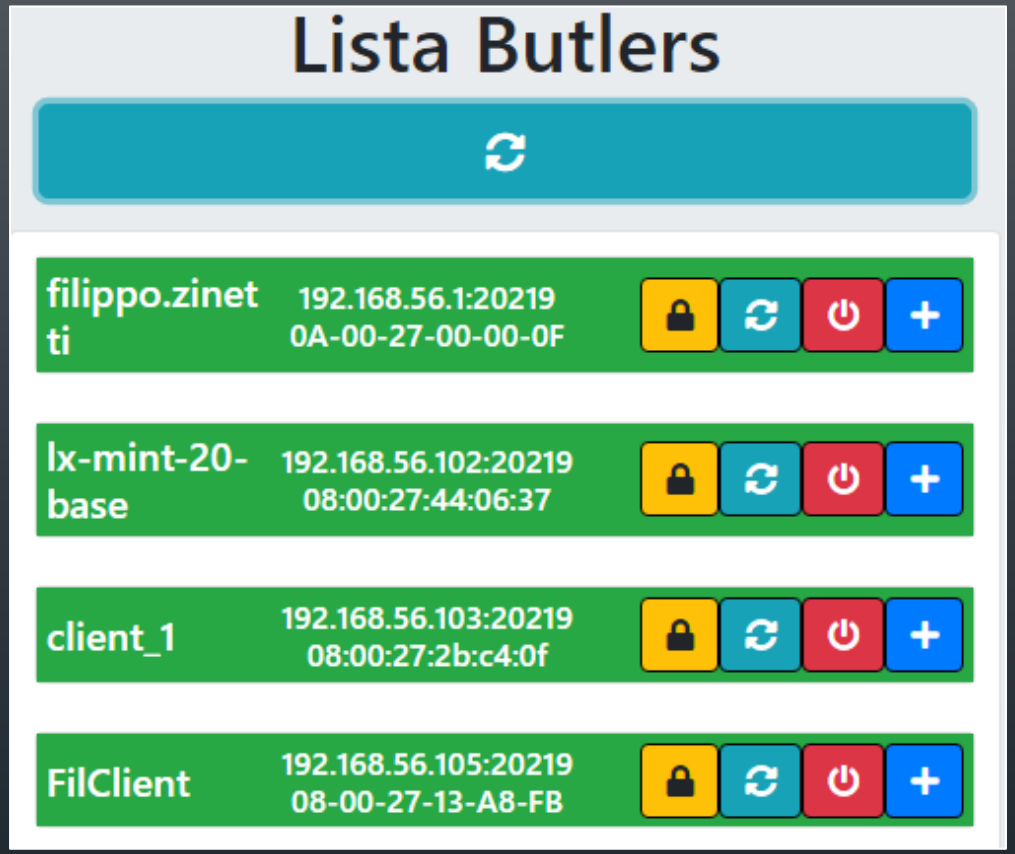
Multipiattaforma



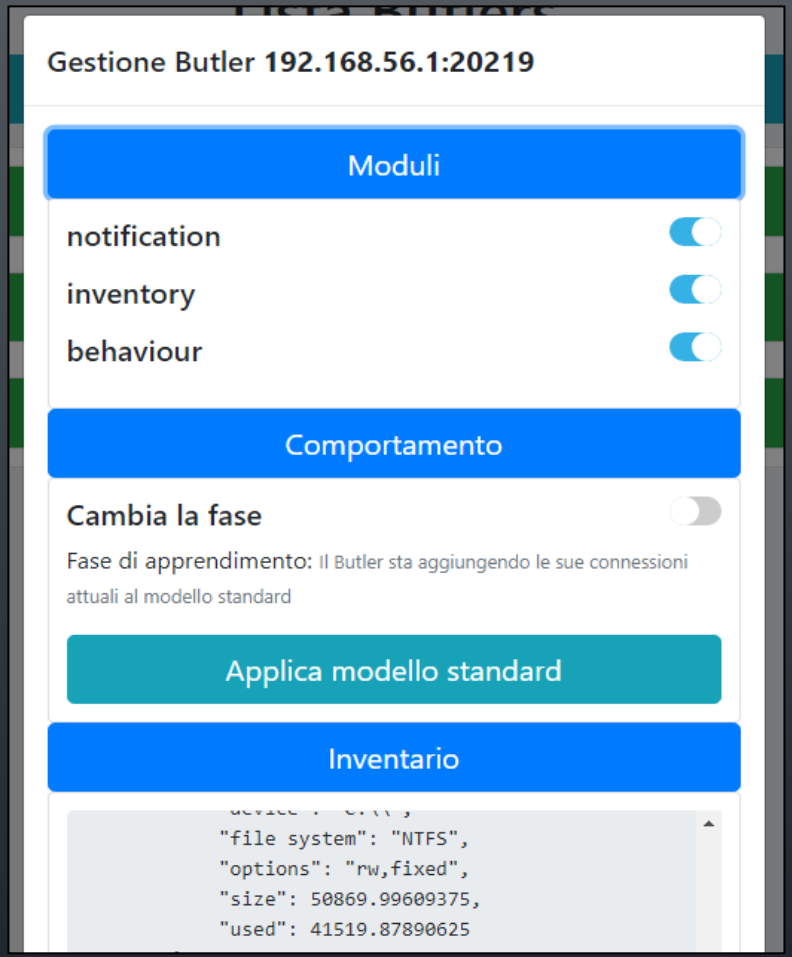


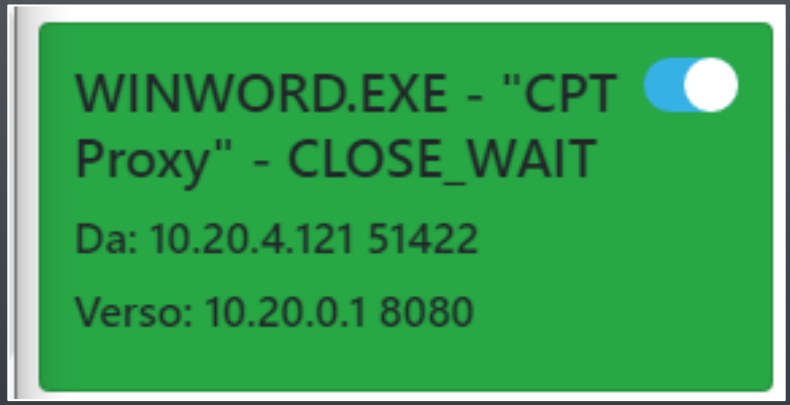


Interfaccia e notifiche Butler



Gestione clients





Aggiunta interfaccia server

Conclusioni

- ❑ Librerie in corso di sviluppo
- ✓ Programma stabile
- ✓ Modulare
- ✓ Uso pratico

Spazio domande

Contatti

- Filippo Zinetti (allievo): fzinetti@bluewin.ch
- Fabio Piccioni (formatore): fabio.piccioni@edu.ti.ch
- Antonio Fontana (perito 1): antonio.fontana@rsi.ch
- Claudio Bortoluzzi (perito 2): claudio.bortoluzzi@rsi.ch
- Guido Montalbetti (responsabile progetti 2021): guido.montalbetti@edu.ti.ch
- CPT Trevano: decs-cpt.trevano@edu.ti.ch

Materiale aggiuntivo

Allievo:	Filippo Zinetti
Formatore:	Fabio Piccioni
Primo perito:	Antonio Fontana
Secondo perito:	Claudio Bortoluzzi
Responsabile:	Guido Montalbetti
Luogo:	Scuola Arti e Mestieri Trevano

Modifiche al server

- Comunicazione con i Butlers
- GUI Centro di controllo
- Gestione comandi
- Database

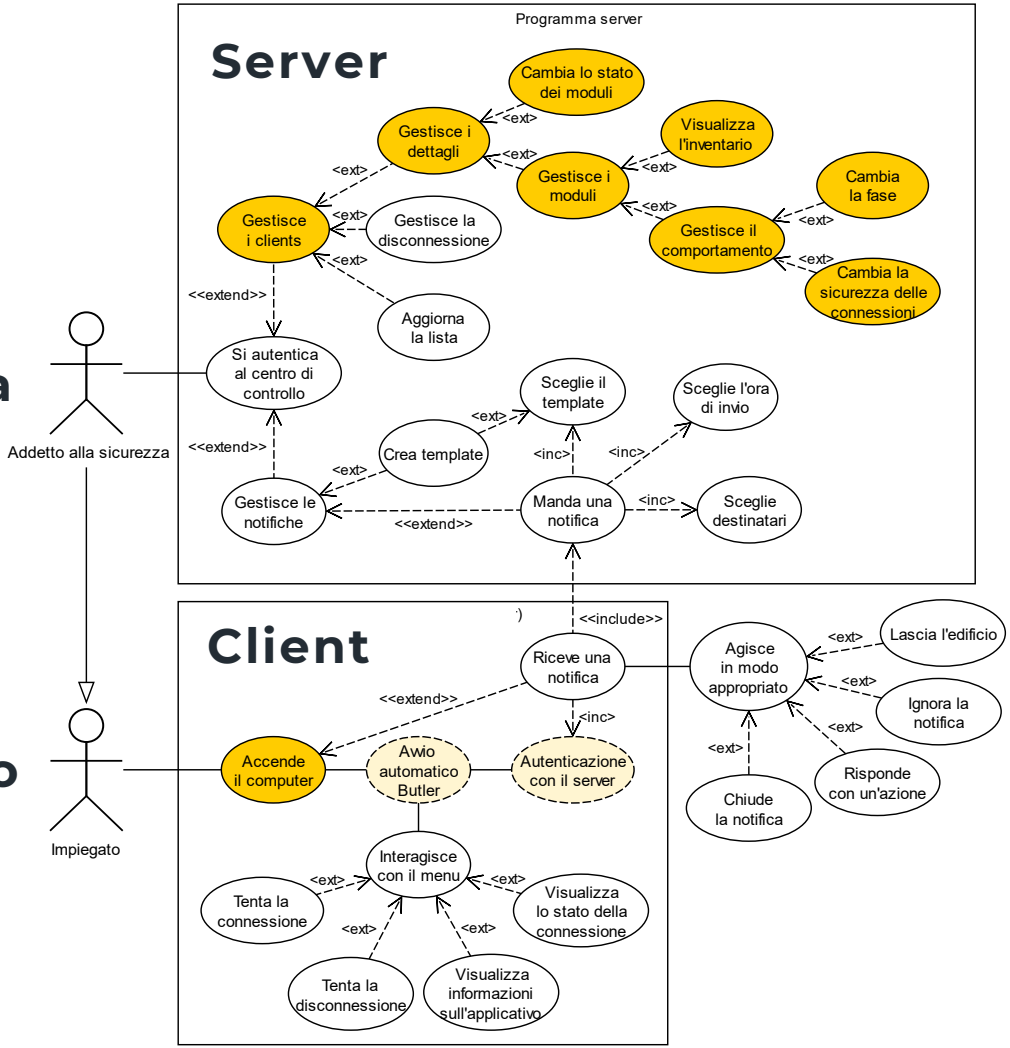
- Classe «Inventory»
- Classe «Behaviour»
- Comunicazione con il server

Modifiche al client

Casi d'uso

Addetto alla sicurezza

Impiegato



Server

- Creazione stile notifiche
- Invio dati notifiche

- Ricezione dati notifiche
- Comparsa pop-up

Agent client “Butler”

Librerie principali

Design notifiche personalizzabile

PySimpleGUI

Richieste HTTPS

requests

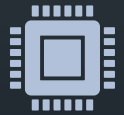
Database NoSQL

pymongo

Comunicazione stateless

flask

Inventario e connessioni



psutil

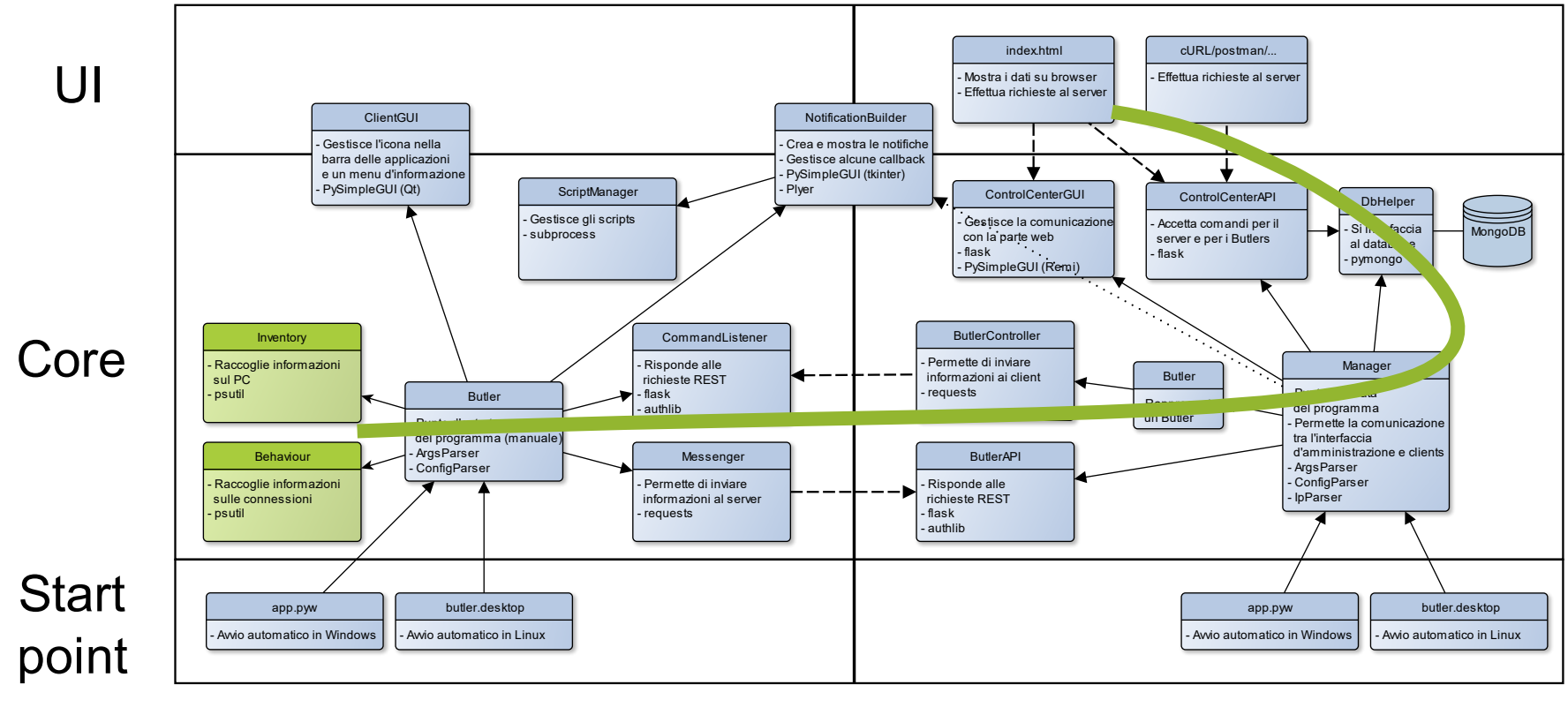
Sicurezza con JWT

authlib

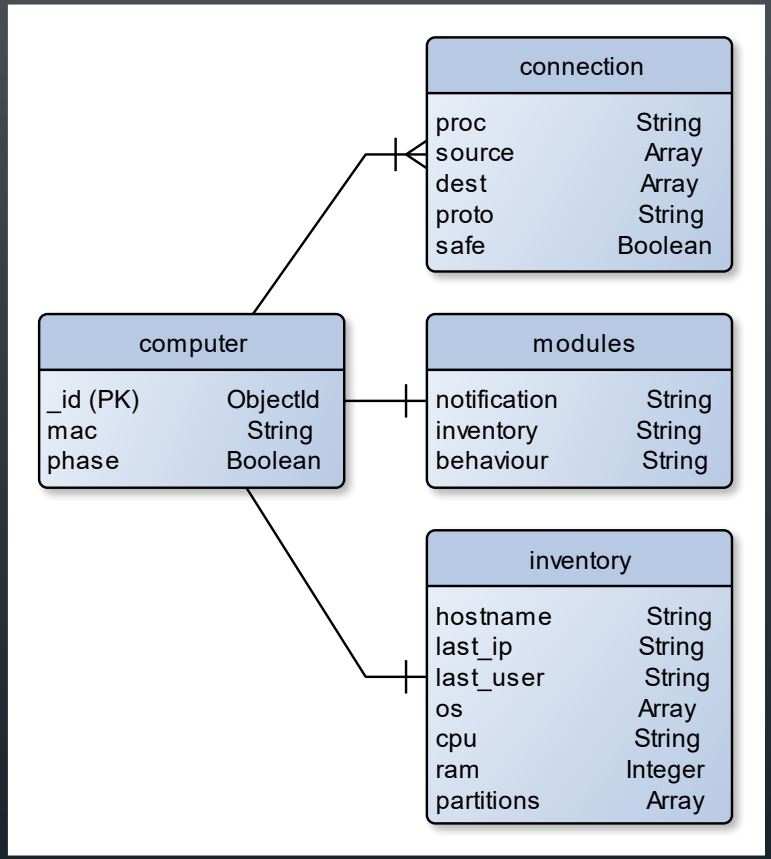
Behaviour

- Raccolta di informazioni
  - Fase di **apprendimento**
  - Fase di **analisi**
- Modello standard generale

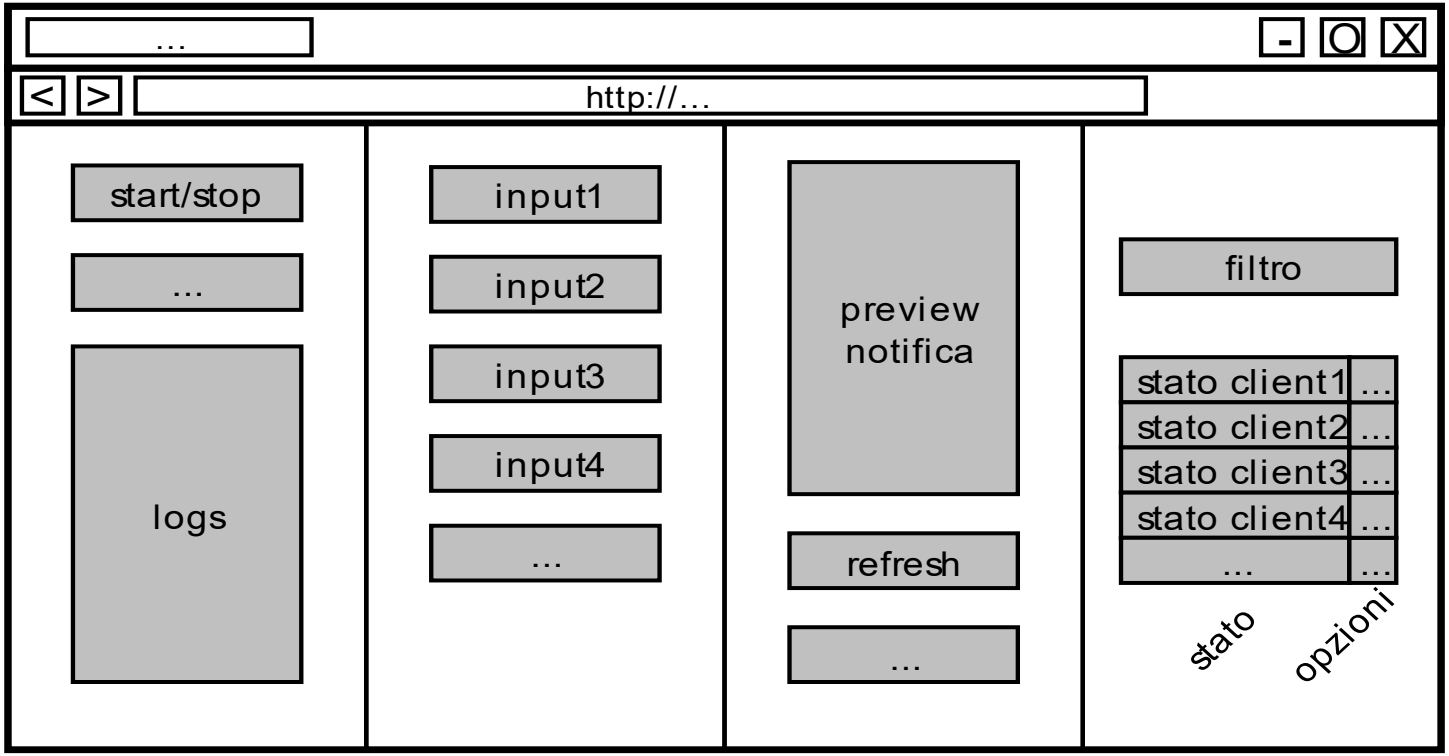
Componenti


UI
Core
Start point
ClientGUI
- Gestisce l'icona nella barra delle applicazioni e un menu d'informazione
- PySimpleGUI (Qt)
ScriptManager
- Gestisce gli scripts
- subprocess
NotificationBuilder
- Crea e mostra le notifiche
- Gestisce alcune callback
- PySimpleGUI (tkinter)
- Plyer
Inventory
- Raccoglie informazioni sul PC
- psutil
Behaviour
- Raccoglie informazioni sulle connessioni
- psutil
Butler
del programma (manuale)
- ArgsParser
- ConfigParser
CommandListener
- Risponde alle richieste REST
- flask
- authlib
Messenger
- Permette di inviare informazioni al server
- requests
app.pyw
- Avvio automatico in Windows
butler.desktop
- Avvio automatico in Linux
Butler
index.html
- Mostra i dati su browser
- Effettua richieste al server
cURL/postman/...
- Effettua richieste al server
ControlCenterGUI
- Gestisce la comunicazione con la parte web
- flask
- PySimpleGUI (Remi)
ControlCenterAPI
- Accetta comandi per il server e per i Butlers
- flask
DbHelper
- pymongo
MongoDB
ButlerController
- Permette di inviare informazioni ai client
- requests
Butler
Manager
del programma
- Permette la comunicazione tra l'interfaccia d'amministrazione e clients
- ArgsParser
- ConfigParser
- IpParser
ButlerAPI
- Risponde alle richieste REST
- flask
- authlib
app.pyw
- Avvio automatico in Windows
butler.desktop
- Avvio automatico in Linux
Server

Modello dei dati

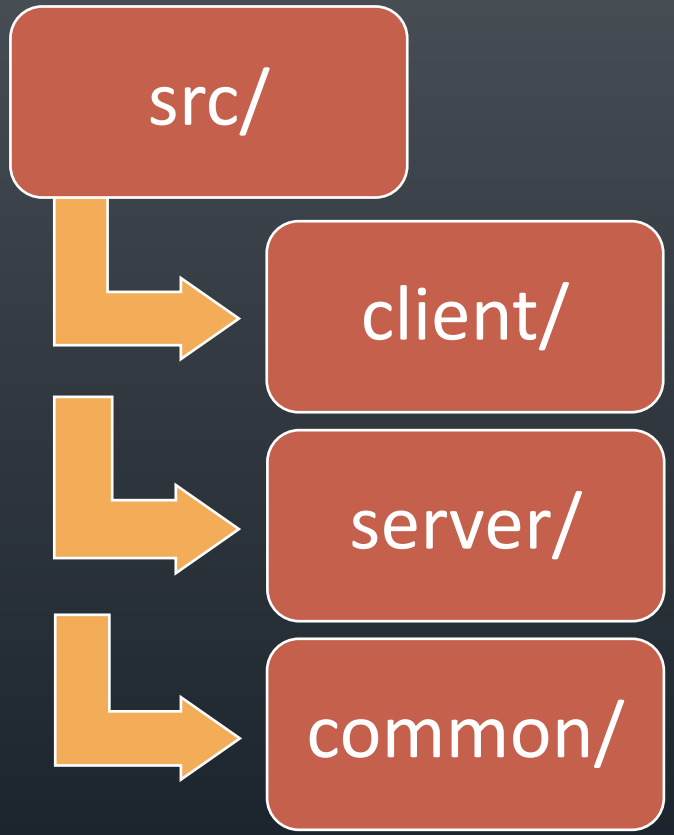



```
▾ 39: Object
    proc: "svchost.exe"
    status: "SYN_SENT"
  ▸ source: Array
  ▸ dest: Array
    proto: "https"
    safe: false
▸ 40: Object
▸ 41: Object
▸ 42: Object
▾ 43: Object
    proc: "ssh.exe"
    status: "SYN_SENT"
  ▸ source: Array
  ▾ dest: Array
      0: "8.8.8.8"
      1: 22
    proto: "ssh"
    safe: false
```



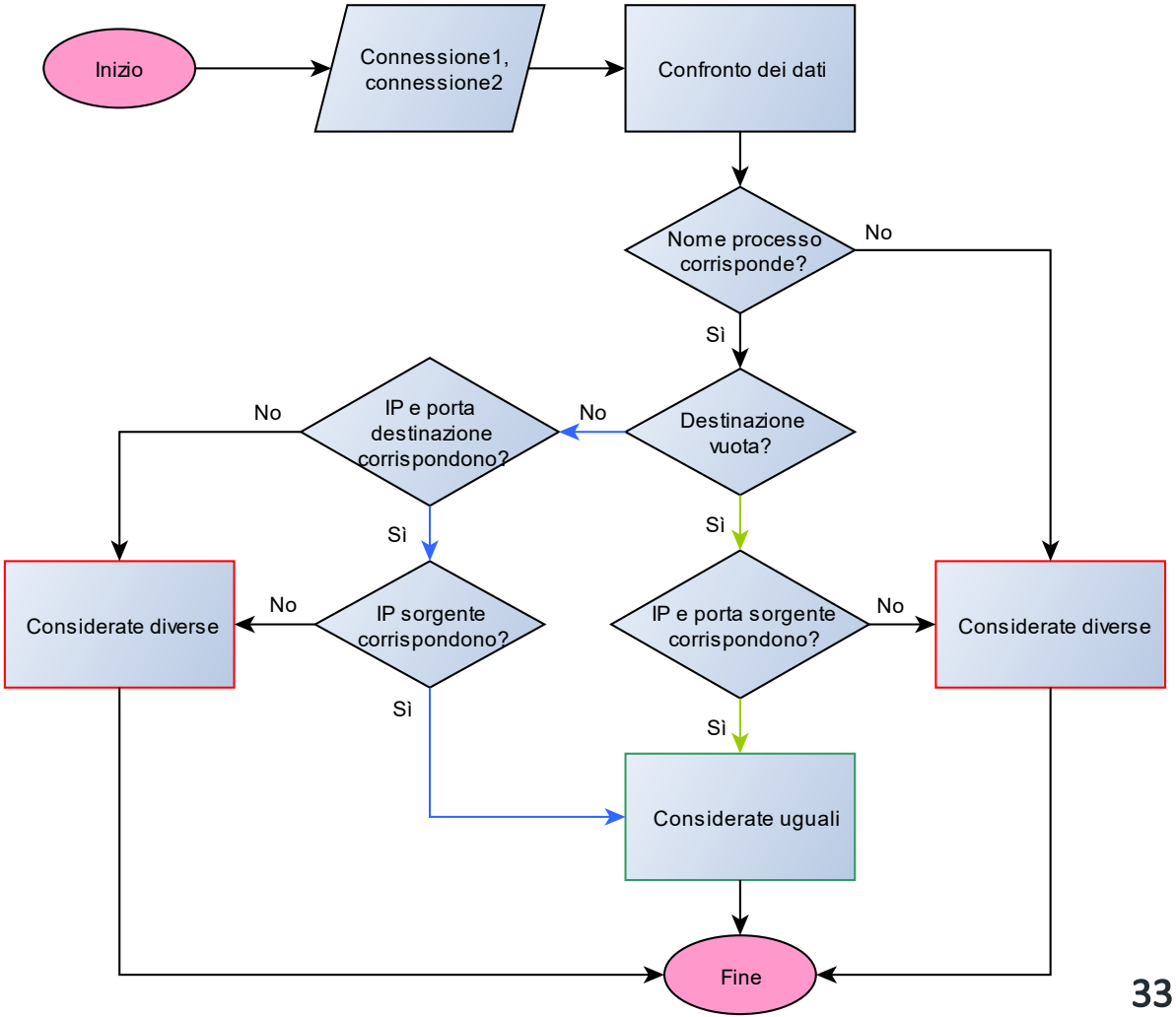
Mockup server

Installazione



- Sviluppati in parallelo
- Molte classi comuni → **common/**
- No .exe

Schema di confronto



Esempi di connessioni di rete

Sorgente		Destinazione	Stato
192.168.1.154:52673	→	192.168.1.107:8008	ESTABLISHED
192.168.1.154:56882	→	192.168.1.107:8009	ESTABLISHED
192.168.1.154:23385	→	52.84.150.34:443	ESTABLISHED
192.168.1.154:15100	→	52.84.150.34:443	ESTABLISHED
0.0.0.0:135	→	0:0	LISTENING
0.0.0.0:902	→	0:0	LISTENING
127.0.0.1:49845	→	0:0	LISTENING
127.0.0.1:65001	→	0:0	LISTENING
192.168.1.154:15096	→	52.84.150.34:443	TIME_WAIT
192.168.1.154:19208	→	52.84.150.34:443	TIME_WAIT
192.168.1.154:23387	→	52.84.150.34:443	TIME_WAIT
192.168.1.154:26807	→	52.84.150.34:443	TIME_WAIT

Query MongoDB

```
result = self.computerColl.find_one_and_update(
    {'mac': mac, 'model': {'$elemMatch': {
        'proc': conn['proc'],
        '$or': [
            {'$and': [
                {'dest.{}'.format(self.IP): ''},
                {'dest.{}'.format(self.PORT): ''},
                {'dest.{}'.format(self.IP): conn['dest'][self.IP]},
                {'dest.{}'.format(self.PORT): conn['dest'][self.PORT]},
                {'source.{}'.format(self.IP): conn['source'][self.IP]},
                {'source.{}'.format(self.PORT): conn['source'][self.PORT]},
            ]},
            {'$and': [
                {'dest.{}'.format(self.IP): {'$ne': ''}},
                {'dest.{}'.format(self.PORT): {'$ne': ''}},
                {'dest.{}'.format(self.IP): conn['dest'][self.IP]},
                {'dest.{}'.format(self.PORT): conn['dest'][self.PORT]},
                {'source.{}'.format(self.IP): conn['source'][self.IP]}
            ]}
        ]
    }}},
    {'$set': {'model.$': conn} }
```

```
    conn1['proc'] == conn2['proc'] and (
    (
        conn1['dest'] == ['',''] and
        conn2['dest'] == ['','']
        and conn1['source'] == conn2['source']
    ) or (
        conn1['dest'] != ['','']
        and (
            conn1['dest'] == conn2['dest']
            and conn1['source'][0] == conn2['source'][0]
        ) or (
            conn1['dest'][0] == conn2['dest'][0]
            and conn1['source'] == conn2['source'])
    ))
```

Condizione modulo Behaviour