Sauron

System Alarmujący o Uczniach Robiących Oszustwa Naukowe

Patryk Mroczyński
Jakub Wiśniewski
Daniel Staśczak
Oskar Rutkowski

Podział prac

Patryk Mroczyński:

Aplikacja serwerowa, baza danych - Python, MongoDB

Oskar Rutkowski:

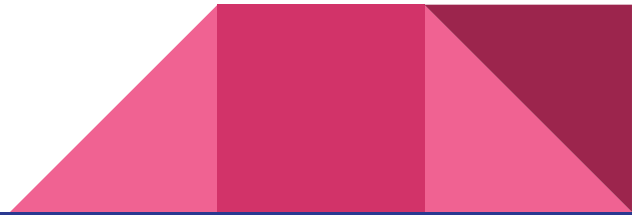
Aplikacja prowadzącego, interfejs - aplikacja webowa

Daniel Staśczak:

Aplikacja monitorująca procesy i odwiedzane strony - Python

Jakub Wiśniewski:

Aplikacja prowadzącego - aplikacja webowa



Szczegóły wymagań

Zostaną stworzone i wykorzystane trzy usługi:

- autoryzacja i tworzenie użytkowników oraz przechowywanie informacji użytkowników - ustawianie whitelisty,
- usługa zbierająca i wysyłające dane o procesach studentów,
- usługa oszustw (wysyłanie następuje po wykryciu oszustwa po stronie studenta - wysyłanie listy procesów, aktywnej w tym czasie whitelisty, screenshota).

Szczegóły wymagań

Aplikacja kliencka:

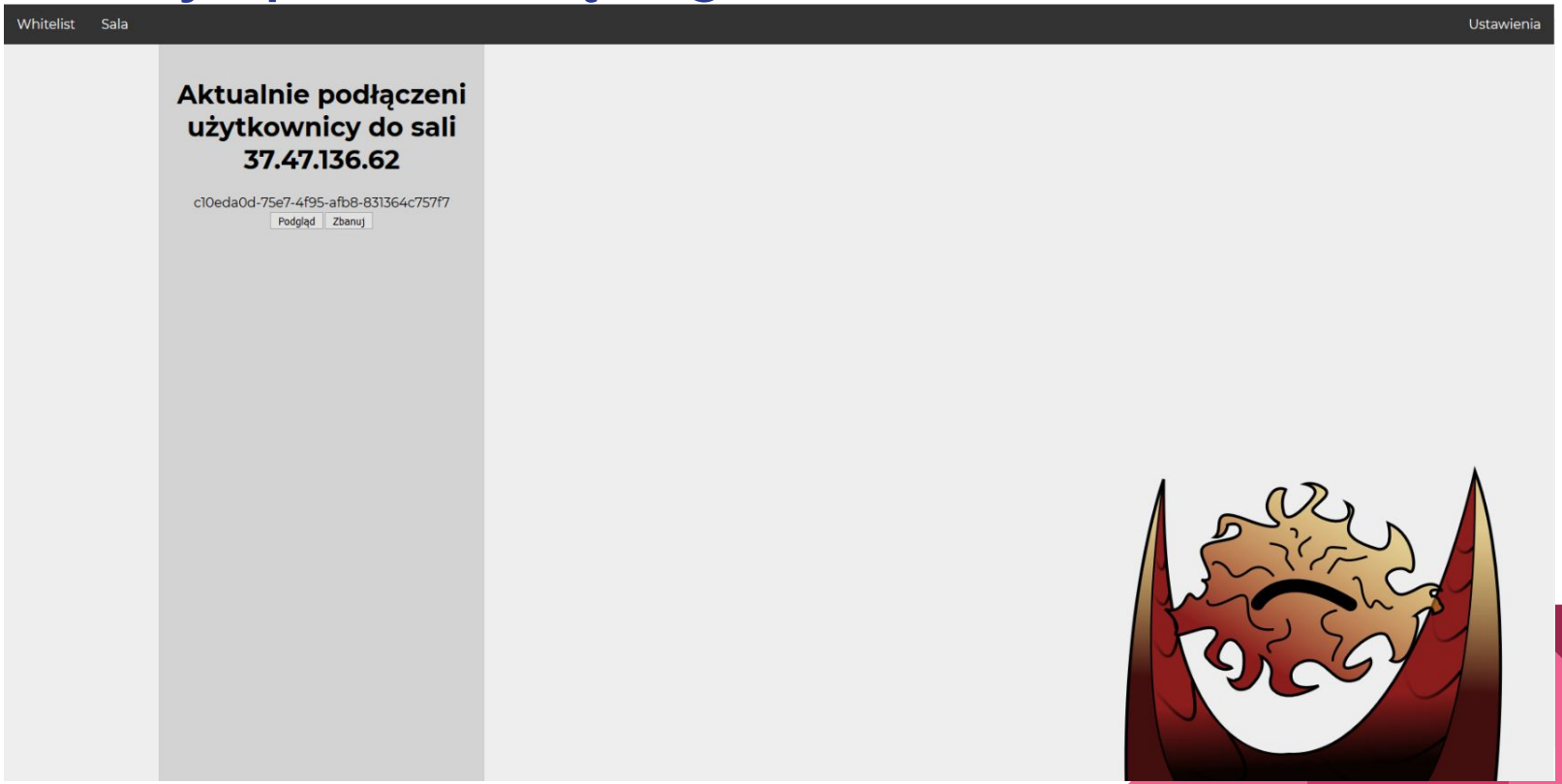
- zbiera:
  - procesy,
  - zakładki,
- wysyła na serwer (nazwę procesu, id procesu, czas utworzenia, nazwę użytkownika, parametry procesu) ,
- pobiera whitelisty z serwera,
- działa w tle.

Szczegóły wymagań

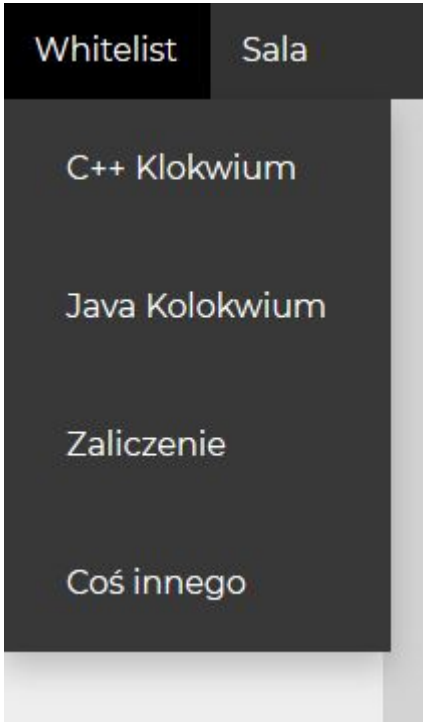
Aplikacja prowadzącego:

- nauczyciel akademicki musi się zarejestrować i zalogować,
- na górze strony, możliwość wyboru sali oraz aktywnej whitelisty,
- pobiera informacje o podłączonych użytkownikach w danej sali (uruchomione procesy, listę otwartych kart przeglądarki) z serwera,
- po lewej stronie lista studentów podłączonych do sali ,
- po kliknięciu na studenta, po prawej stronie wyświetlą się jego szczegóły,
- gdy użytkownik używa niedozwolonych procesów/kart podświetla się na czerwono,
- sortowanie po statusie użytkowników,
- zbieranie informacji z serwisu oszustw.

Aplikacja prowadzącego



Aplikacja prowadzącego



Dokumenty


```
{
    "login": "frodo",
    "password": "cfeef46dcc39a17d9f4d9fa2e1acd9d9aaae3bcc55bd63d710f35f2a7aadd146"
}
```



```
{
    "create time": 1524075964.679969,
    "nazgul": "jan.kowalski@student.put.poznan.pl",
    "processes": [
        {
            "name": "systemd",
            "pid": 1,
            "create_time": 1524066222.03,
            "cmdline": [
                "/sbin/init",
                "splash"
            ],
            "username": "root"
        },
        {
            "name": "kthreadd",
            "pid": 2,
            "create_time": 1524066222.03,
            "cmdline": [
                ""
            ],
            "username": "root"
        },
        {
            "name": "python3",
            "pid": 4958,
            "create_time": 1524083164.02,
            "cmdline": [
                "python3",
                "processes.py"
            ],
            "username": "daniel"
        }
    ]
}
```

Context Manager dla bazy MongoDB

```python
from pymongo import MongoClient

class MongoAPI:

    def __init__(self, database, collection):
        self.connection = MongoClient()
        self.collection = self.connection[database][collection]

    def __enter__(self):
        return self

    def __exit__(self, *args):
        self.connection.close()

    def insert(self, value, many=False):
        self.collection.insert_one(value) if not many else self.collection.insert_many(value)

    def get(self, conditions, many=False):
        try:
            return dict(self.collection.find_one(conditions, {'_id': False})) if not many else list(self.collection.find(conditions, {'_id': False}))
        except TypeError:
            return None
```

Prototyp pierwszego serwisu

```python
import cherrypy
from database import MongoAPI
import json


@cherrypy.expose
class UserService:

    def GET(self):
        with MongoAPI(database='sauron', collection='users') as col:
            return json.dumps(col.get(None, True))

    @cherrypy.tools.json_in()
    def POST(self):
        request = cherrypy.request.json
        try:
            with MongoAPI(database='sauron', collection='users') as col:
                col.insert({'login': request['login'], 'password': request['password']})
        except KeyError:
            raise cherrypy.HTTPError(400, 'BAD REQUEST')


if __name__ == '__main__':
    cherrypy.config.update({
            'server.socket_host': '10.0.2.15',
            'server.socket_port': 8080,
        })

    cherrypy.quickstart(UserService(), '/users', {'/': {'request.dispatch': cherrypy.dispatch.MethodDispatcher()}})
```

Odczyt listy procesów poprzez psutil

```
import datetime
import json

import psutil

NAZGUL_NAME = [u.name for u in psutil.users() if u.host == 'localhost'][0]
PROCESSES = [p.as_dict(attrs=['pid',
                              'name',
                              'cmdline',
                              'username',
                              'create_time']) for p in psutil.process_iter()]

OUTPUT = {'nazgul': NAZGUL_NAME,
          'processes': PROCESSES,
          'create_time': datetime.datetime.now().utcnow().timestamp()}
print(json.dumps(OUTPUT, indent=4))
```