**Cracker – Pcycys Project**

**specification  .**

Project location: <https://github.com/impact-oriented-programming/meat-is-murder>

The final project includes a [**client**](https://docs.google.com/document/d/1I7_SiKlTZmZDTZZvVg5YbOY_0N88Zw_btX1xWAm575E/edit#heading=h.k60mspk3w4qt), which streams Imaris CSVs output post segmentation to a [**server**](https://docs.google.com/document/d/1I7_SiKlTZmZDTZZvVg5YbOY_0N88Zw_btX1xWAm575E/edit#heading=h.9bf9r4r2y3fj), where multiple [**parsers**](https://docs.google.com/document/d/1I7_SiKlTZmZDTZZvVg5YbOY_0N88Zw_btX1xWAm575E/edit#heading=h.99ab55sjy191) read the CSVs, parse various parts of it, and publish the parsed results, which are then [saved](https://docs.google.com/document/d/1I7_SiKlTZmZDTZZvVg5YbOY_0N88Zw_btX1xWAm575E/edit#heading=h.g4spox6qlmn) to a [**database**](https://docs.google.com/document/d/1I7_SiKlTZmZDTZZvVg5YbOY_0N88Zw_btX1xWAm575E/edit#heading=h.y8lowwrwon84).

The results are then exposed via a [**RESTful API**](https://docs.google.com/document/d/1I7_SiKlTZmZDTZZvVg5YbOY_0N88Zw_btX1xWAm575E/edit#heading=h.d9vbnx6wkxzg), which is consumed by a [**CLI**](https://docs.google.com/document/d/1I7_SiKlTZmZDTZZvVg5YbOY_0N88Zw_btX1xWAm575E/edit#heading=h.noqofwn5full); there's also a [**GUI**](https://docs.google.com/document/d/1I7_SiKlTZmZDTZZvVg5YbOY_0N88Zw_btX1xWAm575E/edit#heading=h.eodaw5u0ddrv), which visualizes the results in various ways.

**the project**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| $ git clone git@github.com:[impact-oriented-programming/](https://github.com/impact-oriented-programming/meat-is-murder)ebay | |  |
|  |

Project package at **meat-is-murder/meatismurder**

Installation:

|  |  |  |
| --- | --- | --- |
|  |  |  |
| $ **cd** **meat-is-murder**  $ ./scripts/install.sh  …  $ **source** .env/bin/activate  [**meatismurder**] $ | |  |
|  |

We should be ready to roll.

**the client   .**

The client should be available as **minions.client** and support the following CLI:

**The commands must be in order**

1. Configuration of minions server uplaod:

|  |  |  |
| --- | --- | --- |
|  |  |  |
| $ **python** -m minions.client upload-config \       -h/--host '127.0.0.1'             \        -p/--port 8000                    \        -hp/--hashedpassword 0500000000   \        'db.ini' | |  |
|  |

Where db.ini file can contain multiple fields. For example:

|  |  |  |
| --- | --- | --- |
|  |  |  |
| [name\_of\_the\_minion]  state = already up or not  ip = ip address  host = host to be  port = port to be  user = username of the machine it going to run on  password = the respective password  [up\_local\_or\_not\_local]  state = up  host = 127.0.0.1  port = 5555  [down\_local]  ip = 127.0.0.1  user = sahar gavriely  password =  port = 5556  [down\_not\_local]  ip = 127.0.0.2  user = sahar gavriely  password =  port = 5557 | |  |
|  |

1. Resetting the last command due to crush of the master server :

|  |  |  |
| --- | --- | --- |
|  |  |  |
| $ **python** -m minions.client reset-server \       -h/--host '127.0.0.1'             \        -p/--port 8000 | |  |
|  |

**the master server  .**

The server should be available as **minion.server** and expose the following CLI:

|  |  |  |
| --- | --- | --- |
|  |  |  |
| $ **python** -m minions.server run-server \        -h/--host '127.0.0.1'          \        -p/--port 8000 | |  |
|  |

The server itself should receive the uploaded CSVs and decide if this CSV is relevant for the experiment type, all the CSV’s are saved in a cache folder which resets every time the server is initiated, when the relevant CSV’s are in the cache the server pass them to the parsers based on the experiment type and the parser\_type mentioned in the input file.

When the server is initialized it loads all the available parsers and is ready to receive files and locate them in a temporary folder inside the cache folder named as the postion\_id.

**the minion server  .**

The server should be available as **minion.minionserver** and expose the following CLI:

|  |  |  |
| --- | --- | --- |
|  |  |  |
| $ **python** -m minions.minionserver run-server \        -h/--host '127.0.0.1'          \        -p/--port 8000 | |  |
|  |

The server itself should receive the uploaded CSVs and decide if this CSV is relevant for the experiment type, all the CSV’s are saved in a cache folder which resets every time the server is initiated, when the relevant CSV’s are in the cache the server pass them to the parsers based on the experiment type and the parser\_type mentioned in the input file.

When the server is initialized it loads all the available parsers and is ready to receive files and locate them in a temporary folder inside the cache folder named as the postion\_id.