BOT

Общие положения

1. При открытии бота, устанавливаются соединения с DGTX сервером и ядром

**{login: { 'name':name, 'psw':psw, 'apikey':ak, 'status':'wait'} }**

1. Входящее соединение websocket добавляется в словарь CONNECTIONS

**{ websocket: {‘id’:id, ‘ts’:time, ‘type’:type } }**

**‘id’ –** генерируемый идентификатор соединения

**‘ts’ –** время добавления

**‘type’ –** тип соединения, при добавлении в словарь, устанавливается в **‘wait’**

Соединившийся клиент должен в течении 10 сек зарегистрироваться, иначе соединение закрывается.

1. При закрытии соединения, оно удаляется из словаря CONNECTIONS
2. Входящие и исходящие сообщения должны соответствовать формату

**{ ‘id’:id, ‘message\_type’:mt, ‘data’:{ } }**

* **id**– идентификатор сообщения
* **message\_type** – тип сообщения, может быть:

**registration –** сообщение от клиента ядру, для регистрации в качестве бота или менеджера

**mc** – сообщение от менеджера к ядру

**cm** – сообщение от ядра менеджеру

**cb** – сообщение от ядра боту

**bc** – сообщение от бота ядру

* **data**– словарь данных, зависящих от **message\_type (**типа сообщения)

Типы сообщений. **message\_type**

1. **‘registration’ –** регистрация клиента

‘**data**: { ‘**typereg’:typereg**, […] }

‘**typereg’** может принимать значения: ‘**rocket’** или ‘**manager’**.

Это сообщение присылают клиенты для регистрации их как бота или центра управления.

* **‘typereg’:‘rocket’: -** регистрация в качестве бота

**‘data:{ ‘typereg’:’rocket’, ‘version’:version, ‘psw’:psw }**

**‘version’ –** версия программы бота

Порядок действий:

1. В словарь ROCKETS заносится запись следующего вида:

**websocket:{'ts':time, 'version':version, 'status':status}**

**websocket –** соединение

**ts –** время регистрации

**version –** версия программы бота

**status –** при регистрации устанавливается в **‘wait’**

1. В словаре CONNECTIONS: **type** устанавливается в **‘rocket’**
2. Отправляется ответ:

**{ ‘id’:id, ‘status’:’ok’, ‘data’:{} }**

* **‘typereg’:‘manager’: -** регистрация в качестве центра управления

**‘data:{ ‘typereg’:’manager’, ‘user’:user, ‘psw’:psw }**

**‘user’ –** имя пользователя

**‘psw’ -** пароль

Порядок действий:

1. Проверяются **user** и **psw.** Если проверка не пройдена, сервер отправляет сообщение:

**{ ‘id’:id, ‘status’:’error’, ‘data’:{‘message’:’Bad user or psw’} }**

иначе дальнейшие пункты.

1. В словарь MANAGERS заносится запись следующего вида:

**websocket:{'ts':time, 'user':user}**

**websocket –** соединение

**ts –** время регистрации

**user –** имя пользователя

1. В словаре CONNECTIONS: **type** устанавливается в **‘manager’**
2. Отправляется ответ:

**{ ‘id’:id, ‘status’:’ok’, ‘data’:{} }**

1. **‘mc’ –** сообщение от менеджера ядру

**data**: { ‘**command’:command**, […] }

* **‘command’:’getrockets’ –** получить данные о ботах

Сервер отправляет приславшему сообщение центру управления ответ в виде:

**{ ‘id’:id, ‘status’:’ok’, ‘data’:{**

**websocket:{'ts':time, 'version':version, 'status':status}**

**} },**

где **websocket –** идентификатор соединения из словаря CONNECTIONS, остальные данные из словаря ROCKETS

* **‘command’:’getpilots’ –** получить данные о пилотах

Сервер отправляет приславшему сообщение центру управления ответ в виде:

**{ ‘id’:id, ‘status’:’ok’, ‘data’:{**

**login:{ 'name':name, 'status':status}**

**} },**

из словаря PILOTS

* **‘command’:’getmanagers’ –** получить данные о центрах управления

Сервер отправляет приславшему сообщение центру управления ответ в виде:

**{ ‘id’:id, ‘status’:’ok’, ‘data’:{**

**websocket:{'ts':time, 'user':user}**

**} },**

где **websocket –** идентификатор соединения из словаря CONNECTIONS, остальные данные из словаря MANAGERS

* **‘command’:‘start\_race’ –** авторизовать пилота на ракете

Дополнительные параметры:

‘login’:login – логин пилота

‘rocket\_id’:rid – идентификатор ракеты

Порядок действий:

1. Соединению из словаря ROCKETS, соответствующему rid, отправляется сообщение: