Документация по работе с JSON ответами от сервера Гигахруща

Базовая структура всех объектов

Любой ответ от сервера хранит JSON объект:

```
{
    "type" : "ANSWER/SERVER/MAP",
    "content" : {}
}
```

type хранит один из трех типов, SERVER используется для оповещений, MAP нужен для обновления карты на клиенте и тебе для бота не потребуется, ANSWER является ответом от сервера на команду от клиента

content хранит объект, который вернул сервер, и держит в себе весь основной ответ от сервера.

Content

У всех контентов базовая структура такая:

```
{
    "type" : "Команда",

    // Поля, соотвествующее типу команды
}
```

type хранит тип команды, всего команд 16 + 2 дополнительных (вложенных объектов в другом объекте):

1. Me
2. Look
3. Move
4. Craft
5. Drop
6. Pickup
7. Inventory
8. ChangeFloor
9. LookItem
10. Recipes

11. UseItem
12. Attack
13. Battle
14. Equip
15. RepairExit
16. PlayersInRoom

Для каждого из типов команд свой набор полей и объектов

!Все ниже перечисленные объекты являются именно вложенным content в основном объекте!

Me

```
{
    "type" : "Me",

    "username" : string,
    "level" : int,
    "exp" : int,
    "expToLU" : int,
    "maxInventory" : int,
    "health" : int,
    "armor" : int,
    "weaponSkill" : int,
    "currentWeapon" : string
}
```

Команда "я", хранящая все данные об игроке, все в принципе и так интуитивно понятно, но на всякий expToLU показывает кол-во опыта, необходимое до след уровня.

Look

```
{
    "type" : "Look",
    "coordinates": {
        "x" : int,
        "y" : int,
        "floor" : int,
        "canGoUp" : bool,
        "camGoDown" : bool,
        "isExitBroken" : bool,
    }

    "locationName" : string,
    "locationDescription" : string,
```

```
    "items" : [
        {
            "name" : string,
            "description" : string
        },
        {
            "name" : string,
            "description" : string
        }
    ],
    "enemies" : [
        {
            "name" : string,
            "description" : string
        },
        {
            "name" : string,
            "description" : string
        }
    ],

    "sides" : [string,string...]
}
```

Команда "осмотреться" возвращает всю информацию о комнате.

coordinates это объект который хранит инфу о текущих координатах а так же дополнительных свойствах, которые могут использовать для перемещения.

items и enemy хранят одинаковые структуры и возвращают объекты которые есть в этой комнате, name хранит техническое имя врага или предмета
description хранит описание этого элемента в комнате (Его положение), тебе description не понадобиться, он нужен только для клиента.

sides хранит массив строк с возможными выходами в другие комнаты (север, юг, запад, восток)

Move

```
{
    "type" : "Move",

    "canMove" : bool,
    "moved" : bool,
    "unknownSide" : bool,
```

```
    "look" : {Объект Look}
}
```

canMove показывает может ли клиент идти в этом направлении, moved показывает переместился ли игрок (на всякий), unknownSide тебе не понадобиться, это больше мне для дебага.

look хранит в себе объект команды Look, на клиенте у меня был небольшой рекурсивный парсинг где я после команды move отправлял в этот же парс обьект look из этой команды для доп вывода инфы, синтаксис look такой же поэтому дополнительно парса делать не нужно и можно вызвать уже готовую функцию, если ты не переместился то look будет пустым.

Craft

```
{
    "type" : "Craft",

    "noItemFound" : bool,
    "noResources" : bool,
    "noCraftFound" : bool,

    "lostedItems" : [string,string...],
    "craftedItem" : string
}
```

noItemFound показывает что данный предмет не найден
noCraftFound показывает что предмет существует но крафта для него нет
noResources хранит true если у тебя недостаточно ресурсов для крафта,

lostedItems хранит строковый массив потерянных предметов
cratedItem хранит имя скрафченого предмета в случае удачной попытки крафта.

Drop

```
{
    "type" : "Drop",

    "dropped" : string,
    "itemFound" : bool
}
```

dropped хранит имя выброшенного предмета
itemFound хранит true если такой предмет есть в инвентаре, иначе false

Pickup

```
{
    "type" : "Pickup",

    "canPickup" : bool,
    "item" : string,
    "pickuped" : bool,
    "isInventoryFull" : bool
}
```

canPickup показывает можешь ли ты поднять предмет в этой комнате, вернет false если там есть враги
pickuped хранит false если ты не смог поднять этот предмет, то есть он просто не найден
item хранит имя предмета которое ты подобрал
isInventoryFull вернет true если инвентарь полон и ты уже не можешь подбирать предметы

Inventory

```
{
    "type" : "Inventory",
    "items" : [string,string..]
}
```

Просто хранит массив текущих предметов в инвентаре

ChangeFloor

```
{
    "type" : "ChangeFloor",

    "isElevatorBroken" : bool,
    "isPlayerBrokeElevator" : bool,
    "canGoUp" : bool,
    "canGoDown" : bool,
    "moveType" : int,
    "changed" : bool,
    "canChange" : bool,
    "changedFloor" : int
}
```

Тут слегка заёбная структура, всё потому-что это нужно для правильного вывода ошибки на клиенте

isElevatorBroken показывает сломан ли лифт на этаже
isPlayerBrokeElevator показывает сломал ли этот лифт игрок сейчас при попытке сменить этаж
canGoUp и canGoDown показывает че куда он может переместиться
moveType хранит int значение 1 - вверх 0 - вниз, это нужно просто для клиента чтобы вывести правильно, тебе не нужно
changed хранит true если ты переместился
canChange хранит true если это комната для выхода на этаж, иначе false, тоже нужно для вывода ошибки на клиенте
changedFloor хранит номер этажа на который ты переместился

LookItem

```
{
    "type" : "LookItem",
    "found" : bool,

    "object" : {Вложенный обьект}
}
```

Несмотря на то что команда LookItem, она так же осматривает и врагов

found хранит найден ли предмет или враг

object хранит вложенный объект, и внутрянка зависит от типа объекта или же врага

Базовый синтаксис object (Вложенный объект в LookItem)

Enemy:

```
{
    "type" : "Enemy",
    "name" : string,
    "description" : string,
    "health" : int,
    "damage" : int
}
```

Возвращает имя и описание врага, а так же его текущее здоровье и дамаг

Item:

Тут есть 4 типа объектов для предметов, Component, Weapon, Armor и Heal

Component:

```
{
    "type" : "Item",
    "ItemType" : "Component",
    "name" : string,
    "description" : string
}
```

Возвращает имя и описание, использовать компонент нельзя поэтому дополнительных свойств у него нет

Weapon:

```
{
    "type" : "Item",
    "ItemType" : "Weapon",
    "name" : string,
    "description" : string
    "damage" : int
}
```

Возвращает имя, описание и урон

Armor:

```
{
    "type" : "Item",
    "ItemType" : "Armor",
    "name" : string,
    "description" : string
    "armor" : int
}
```

Возвращает имя, описание и реген брони

Heal:

```
{
    "type" : "Item",
    "ItemType" : "Heal",
    "name" : string,
    "description" : string
    "heal" : int
}
```

Возвращает имя, описание и реген здоровья

Recipes

```
{
    "type" : "Recipes",

    "enableCrafts" : [string,string...]
}
```

Возвращает список доступных крафтов.

UseItem

```
{
    "type" : "UseItem",
    "haveItem" : bool,

    "enemyStep" : {Вложенный объект атаки врага},
    "checkPlayerDeath" {Вложенный объект CheckPlayerDeath}:,

    "playerInBattle" : bool,

    "item" : {Вложенный объект}
}
```

haveItem показывает есть ли этот предмет в инвентаре

playerInBattle показывает в битве игрок или нет, нужно для последующего парса enemyStep и checkPlayerDeath

item хранит объект, который тоже уникальный для каждого из типов

enemyStep (Вложенный объект в UseItem)

```
{
    "enemyName" : string,
    "hasArmor" : bool,

    "replic" : string,

    "loseHealth" : int,
    "loseArmor" : int,

    "currentHealth" : int,
    "currentArmor" : int
}
```

Объект хранит инфу о том на сколько враг задамажил игрока
hasArmor показывает имел ли игрок броню в момент атаки, нужно в частности для клиента для вывода правильного

replic хранит реплику врага, тоже чисто клиенту, тебе нахуй не нужно

loseHeath и loseArmor показывает сколько ты потерял здоровья и брони после атаки врага
current и так понятно, показывает текущее здоровье и броню

checkPlayerDeath (Вложенный объект в UseItem)

```
{
    "isDead" : bool,
    "x" : int,
    "y" : int,
    "Floor" : int
}
```

хранит переменную мертв ли игрок и координаты его смерти если он сдох

item (Вложенный объект в UseItem)

Тут тоже 4 типа как и в LookItem, но другие

Component:

```
{
    "type" : "Component"
}
```

Возвращает только тип, его нельзя использовать поэтому никаких доп полей не требуется

Weapon:

```
{
    "type" : "Weapon"
}
```

Возвращает только тип, его нельзя использовать поэтому никаких доп полей не требуется

Armor:

```
{
    "type" : "Armor",
    "name" : string,
    "description" : string
    "armor" : int,
    "used" : bool
}
```

Возвращает имя, описание при использовании и реген брони
used показывает использовался ли этот предмет, если нет то значит у тебя полная броня

Heal:

```
{
    "type" : "Heal",
    "name" : string,
    "description" : string
    "heal" : int,
    "used" : bool
}
```

Возвращает имя, описание при использовании и реген здоровья
used показывает использовался ли этот предмет, если нет то значит у тебя полное здоровье

Attack

```
{
    "type" : "Attack",

    "inBattle" : bool,
    "wepEquiped" : bool,

    "enemyName" : string,

    "makeDamage" : int,
    "skillDamage" : int,
    "enemyRemainHealth" : int,

    "isEnemyDead" : bool,
    "winExp" : int,
    "itemFromEnemy" : string,

    "enemyStep" : {Вложенный объект атаки врага},
    "checkPlayerDeath" : {Вложенный объект CheckPlayerDeath},
```

```
    "levelUp" : {Вложенный объект CheckLevelUp}
}
```

Ебнутая хуйня

inBattle показывает в битве ли игрок, если нет то атаковать нет смысла там дальше нихуя нет
wepEquiped показывает экипировано оружие, если нет то он скипает ход и делает атаку врага

enemyName хранит имя врага которого ты атакуешь

makeDamage показывает сколько ты дамага нанёс врагу
skillDamage это бонусный дамаг к основному
enemyRemainHealth показывает сколько у врага осталось хп

isEnemyDead показывает сдох ли враг, если он сдох получишь winExp - опыт, и itemFromEnemy - предмет который тебе выпал с него

enemyStep и checkPlayerDeath и есть тоже самое что и UseItem, атака врага и проверка смерти игрока, выведи этот парс в отдельную функцию

levelUp это вложенный объект на проверку нового уровня

levelUp (Вложенный объект в Attack)

```
{
    "isLevelUp": bool,

    "newLevel" : int,
    "nextLevelExp" : int,
    "currentExp" : int
}
```

isLevelUp хранит true если ты получил новый уровень, вообще вряд ли тебе для бота это понадобиться, так что забей хуй

Battle

```
{
    "type" : "Battle",

    "enemyInBattle" : bool,
    "enemyInBattlePlayerName" : string,

    "startedBattle" : bool,
    "startedBattleWith" : string,
```

```
    "enemyStep" : {Вложенный объект атаки врага},
    "checkPlayerDeath" : {Вложенный объект CheckPlayerDeath}
}
```

enemyInBattle хранит true если другой игрок уже дерётся с этим врагом, его имя хранит в enemyInBattlePlayerName

startedBattle хранит true если игрок начал битву с врагом, имя врага храниться в startedBattleWith

enemyStep и checkPlayerDeath ебать тоже самое, только тут учти, при входе в битву враг может и не атаковать, поэтому эти объекты могут быть пустыми, делай проверку на empty

Equip

```
{
    "type" : "Equip",

    "equiped" : bool,
    "canEquip" : bool,
    "wep" : string
}
```

equiped хранит true если игрок смог экипировать это оружие, иначе false а значит что это не оружие или такого предмета нет

canEquip вернет false если ты пытался экипировать не оружие

wep хранит имя экипированного оружия

RepairExit

```
{
    "type" : "RepairExit",

    "isExitBroken" : bool,
    "canFixThisRoom" : bool,
    "haveResources" : bool,

    "needRes" : [
        {
            "name" : string,
            "count" : int
        },
        {
```

```
            "name" : string,
            "count" : int
        }
    ],
    "lostedItems" : [string,string...],

    "repaired" : bool
}
```

canFixThisRoom вернет true если эта комната является выходом и её фактически возможно починить
isExitBroken вернет false если выход не сломан
haveResources вернет false если недостаточно ресурсов для починки

needRes хранит массив объектов, каждый объект хранит имя предмета, необходимого для починки, а так же его количество, которое нужно чтоб починить

lostedItems хранит массив потерянных предметов, просто строки с именами предметов

repaired вернет true если игрок починил выход

PlayersInRoom

```
{
    "type" : "PlayersInRoom",
    "players" : [string,string..,]
}
```

Возвращает строковый массив игроков в текущей комнате вместе с игроком

+ Skip

Бонусная команда которая нахуй никому не нужна, просто скип хода в битве с врагом, ты ничего не будешь делать а он тебя будет пиздить

```
{
    "type" : "Skip",
    "inBattle" : bool,

    "enemyStep" : {Вложенный объект атаки врага},
    "checkPlayerDeath" : {Вложенный объект CheckPlayerDeath}

}
```

inBattle вернет false если ты попытаешься скипнуть когда ты в битве, если же в битве то тебе враг даст пизды и надо парсить enemyStep и checkPlayerDeath

3 дополнительных ответа от сервера

В случае если ты еблан и неправильно написал команду то сервер может вернёт еще 3 ответа в зависимости от типа ошибки

```
{
    "type" : "inBattle/unknown/badSyntax"
}
```

inBattle показывает что ты пытаешься юзануть эту команду в бою, а там её не место блять
unknown показывает что сервер в душе не ебёт че это такое ты написал ему
badSyntax означает что сервер знает команду но тебе не хватает аргументов для правильного синтаксиса.

P.S.

Весь парсинг выглядит примерно так

```
nlohmann::json js = nlohmann::json::parse(client.recv_buffer_server);

if (js["type"] == "ANSWER") {
    addLog(logs, js);
}
else if (js["type"] == "MAP") {
    map = js["content"];
}
else if (js["type"] == "SERVER") {
    serverMessages.push_back(js["content"]);
}
```

В парсере просто идёт перебор типо в обьекте content

```
if (obj["content"]["type"] == "Move") {
    //Типо парсинг
}
else if (obj["content"]["type"] == "Look") {
    //Типо парсинг
}
else if (obj["content"]["type"] == "Equip") {
    //Типо парсинг
}
```

Для нормальной отладки ответов от сервера, а именно переборов всех bool и выбора какого-то действия относительно например ошибки, чекай парсер на клиенте, там есть база для отладки ответов