# #Protocolo padrão entre o Cliente e o Servidor

#Versão 2.0

A nova versão do protocolo possibilita a criação de um profile para cada usuário. Neste profile são armazenados os login (id) e password. Além disso, cada jogo que o usuário jogar poderá ser adicionado ao seu profile e os itens marcados ou coletados em um jogo serão persistidos junto ao jogo no profile do usuário. Abordagens estruturas ou orientadas a objetos podem ser usadas para representar a ligação entre Profile, Game, Trophies, State e Media.

Requisição:

Arquivo JSON contendo a seguinte estrutura:

{

id: 'id do usuário, definido por cadastro no site',

game: 'id do jogo que ele está atualmente jogando, criado pelo desenvolvedor',

op: 'nome da operação que o servidor deverá executar',

data: 'objeto JSON que descreve os parâmetros da operação'

}

Resposta:

Arquivo JSON com a seguinte estrutura:

{

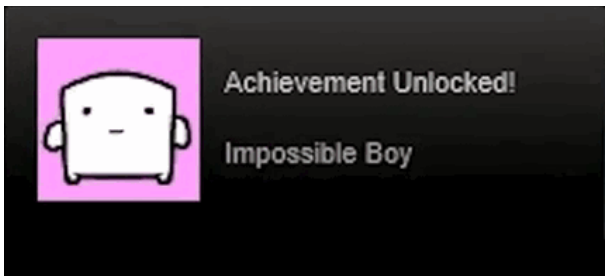
response: 'código da resposta de acordo com o processamento da operação solicitada',

data: 'dados, em JSON, gerado pelo servidor de acordo com o requisição solicitada. Se a resposta gerar um erro, aqui poderá ser retorna o motivo. Caso contrário, retornar o quê é esperado pela operação'

}

# #Operações da versão 2.0

## Conquistas (Achievements/Trophies) (+ GamerScore/XP)




#addProfile

Requisição:

id: 'id do usuário',

op: 'add-profile',

data: '{password: 'gvt12345', email: 'user@user.com'}'

Respostas:

response: 'ok', data: ''

response: 'error', data: 'Id do usuário já existe em nossos servidores'

response: 'error', data: 'Alguma exceção não esperada'

#queryProfile

Requisição:

id: 'id do usuário',
op: 'query-profile',
data: '{password: 'gv12345'}'

Respostas:

response: 'ok', data: '{lastLogin: '2017-06-3T18:25:43.511Z', email: 'user@user.com'}'

response: 'error', data: 'Usuário ou senha inválidos.'

#addGame

Requisição:

id: 'id do usuário',
game: 'id do jogo',
op: 'add-game',
data: '{name: 'Super Pitfal', description: 'Um jogo sensacional e que vai desafiar suas habilidades de sobrevivência na floresta'}'

Respostas:

response: 'ok1', data: 'Jogo adicionado com sucesso ao usuário.'
response: 'ok2', data: 'Jogo já existente para o usuároi, operação ignorada.'
response: 'error', data: 'Usuário não existente'

#addTrophy

Requisição:

id: 'id do usuário',
game: 'id do jogo',
op: 'add-trophy',
data: '{name: '10 coins', xp: 30, title: 'IT\'S A START', description: 'Collected first 10 coins on the game'}'

Resposta:

response: 'ok', data: ''

#getTrophy

Requisição:

id: 'id do usuário',
game: 'id do jogo',
op: 'get-trophy',
data: '10 coins'

Resposta:

response: 'ok', data: '{name: '10 coins', xp: 30, title: 'IT\'S A START', description: 'Collected first 10 coins on the game'}'

#listTrophy

Requisição:

id: 'id do usuário',
game: 'id do jogo',
op: 'list-trophy',
data: ''

Resposta:

response: 'ok', data: '[
{name: '10 coins', xp: 30, title: 'IT\'S A START', description: 'Collected first 10 coins on the game'},
{name: 'first death', xp: 10, title: 'KEEP CALM AND PLAY', description: 'First death on game'},
... ]'

#clearTrophy
Requisição:
id: 'id do usuário',
game: 'id do jogo',
op: 'clear-trophy',
data: ''
Resposta:
response: 'ok', data: ''

## Cloud-Save (Checkpoint)

#saveState
Requisição:
id: 'id do usuário',
game: 'id do jogo',
op: 'save-state',
data: '{x: 130, y: 30}'
Resposta:
response: 'ok', data: ''

#loadState
Requisição:
id: 'id do usuário',
game: 'id do jogo',
op: 'load-state',
data: ''
Resposta:
response: 'ok', data: '{x: 130, y: 30}'

## Media Share (Game DVR): foto (e video)

#saveMedia
Requisição:
id: 'id do usuário',
game: 'id do jogo',
op: 'save-media',
data: '{mimeType: 'image/png', src: 'dados da media serializado via método canvas.toDataURL()'}'
Resposta:
response: 'ok', data: ''
#listMedia
Requisição:
id: 'id do usuário',
game: 'id do jogo',

op: 'list-media',
data: '{mimeType: 'image/png', start: 0, count: 10}'

Resposta:

response: 'ok', data: '[
{mimeType: 'image/png', src: 'BDdje64_jD...'},
{mimeType: 'image/png', src: 'fDkmh62g22...'},
... ]'

# #Considerações sobre a comunicação server-to-server

Para simplificar a implementação, é importante observar que a comunicação entre os servidores não será autenticada e nem criptografada.

A descrição de um cenário básico é:

- O usuário entra no site do jogo
  - O jogo não vai conseguir obter nenhum dado no servidor, pois o usuário não está autenticado
  - O usuário poderá jogar o jogo de forma anônima (sem nada ser persistido)
  - O usuário poderá criar seu **profile** e partir de então ter sua evolução persistida
    - #addProfile
  - O usuário poderá se autenticar na página
    - #queryProfile
    - Pode-se usar os Cookies para gravar o id e password do usuário e possibilitar uma auto-autenticação durante o *reload* da página. Claro que esta abordagem não é segura, mas somente um exercício didático.
    - #addGame no profile recém autenticado
- O usuário interage com o jogo
  - Os dados do usuário (*trophies, state* e *media*) são enviados para o servidor
    - # (mensagens específicas)
- O servidor recebe as mensagens do cliente
  - O servidor deverá ter uma lista de **servidores amigos** com os quais estabelecerá comunicação constante a cada solicitação.
  - Ao receber uma mensagem #queryProfile ou #addProfile o servidor primeiramente solicitará a mesma mensagem aos servidores amigos. Se algum deles responder positivamente, o servidor atual vai manter um mapa ligando o **id** usuário ao **IP** do servidor que atendeu ao pedido. Desta forma, todas as demais requisições deste usuários serão despachadas para o servidor responsável, e o servidor atual atuará como um ***proxy***.
  - Ao receber outras mensagens, o servidor vai primeiramente consultar seu mapa para verificar se o usuário está sendo mantido em outro servidor.