Automação em Tempo Real - 2017/01

Trabalho Final

1. INTRODUÇÃO

Nesse trabalho você irá desenvolver um sistema de rastreamento, semelhante ao extinto "Google Latitude". O sistema deve ser capaz de receber dados de posicionamento de celulares monitorados através da internet, armazenar esses dados e fornecer uma interface gráfica para visualização dessas informações.

O sistema a ser desenvolvido é composto pelos seguintes módulos:

- Interface móvel cliente;
- gateway;
- servidor de aplicação;
- historiador;
- interface gráfica.

Cada módulo do sistema deve ser desenvolvido como uma aplicação independente. A interface móvel cliente e a interface gráfica já estão prontas e serão fornecida no *moodle*.

Os módulos comunicam-se através mecanismos de comunicação entre processos (por exemplo, sockets, filas de mensagens, memória compartilhada, etc.).

O *gateway* é responsável por obter dados de posicionamento periódicos dos celulares monitorados. Para isso, esse módulo é definido como um servidor capaz de lidar com múltiplas conexões remotas concorrentemente. Além disso, o gateway deve possuir, em uma região de memória compartilhada, uma lista de clientes conectados, contendo informações da última posição e estado recebidos. O gateway utiliza o protocolo HTTP para comunicar com os clientes monitorados (mais detalhes na seção 2).

O *historiador* é responsável pela persistência das informações de posicionamento e estado recebidas. Todas as informações recebidas pelo gateway são repassadas ao historiador e devem ser armazenadas em disco, em um diretório de dados. O diretório de dados deve conter um arquivo para cada cliente monitorado. Cada arquivo deve conter os pacotes recebidos, ordenados pela ordem de recebimento. O arquivo de cada cliente deve ser identificado pela Identificação Internacional de Equipamento Móvel (do inglês *International Mobile Equipment Identity – IMEI*) (definido na seção 2). O arquivo de cada cliente deve ser definido em formato binário, e cada registro de posicionamento deve seguir o formato especificado na seção 3.

O *servidor de aplicação* disponibiliza informações sobre os clientes rastreados à uma interface gráfica. Essa interface é utilizada pelos usuários do sistema, para rastrear clientes e visualizar dados históricos. Para isso, o servidor de aplicação requisita dados históricos de posicionamento ao historiador. Além disso, o servidor de

aplicação fornece à interface cliente uma lista de clientes conectados ao sistema. Essa lista é lida a partir de uma região de memória compartilhada entre o gateway e o servidor de aplicação.

A *interface gráfica* permite que o usuário do sistema visualize informações de posicionamento de clientes conectados em tempo real. Além disso, essa interface permite a visualização de dados históricos, referentes a um cliente que já tenha conectado ao sistema previamente.

A comunicação entre o gateway e o historiador é realizada através de uma fila de mensagens definida em uma região de memória compartilhada e segue o protocolo definido na seção 3. O gateway é o produtor de mensagens e o historiador o consumidor.

A comunicação entre o servidor de aplicação e o historiador é também realizada via filas de mensagens seguindo o protocolo definido na seção 4. Além disso, o servidor de aplicação tem acesso à uma lista de clientes conectados ao sistema via uma região de memória compartilhada com o gateway, que é responsável pelo gerenciamento dessa lista (seção 6).

A comunicação entre o servidor de aplicação e a interface gráfica é realizada via rede, seguindo o protocolo definido na seção 5. Conforme mencionado, uma implementação da interface gráfica está disponível no *moodle*.

A figura 1 apresenta um diagrama esquemático da solução:

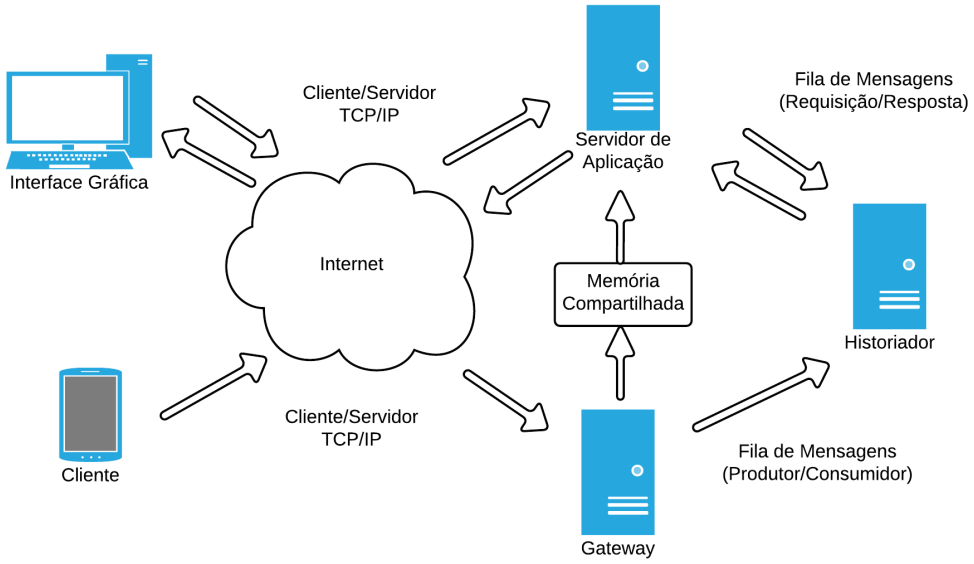


Figure 1 - Diagrama esquemático da solução

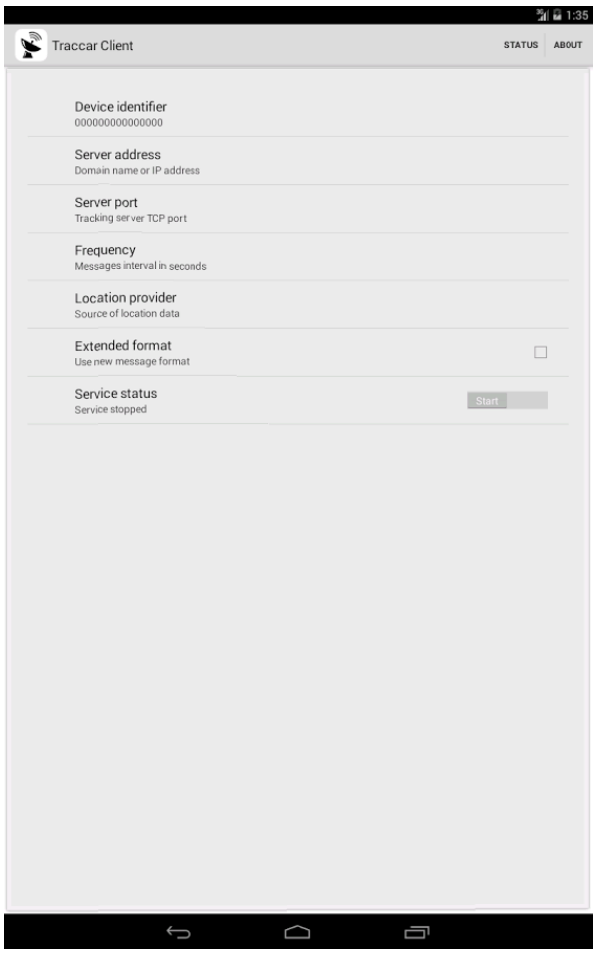
2. DESCRIÇÃO DO PROTOCOLO DE COMUNICAÇÃO ENTRE O GATEWAY E OS CLIENTES MONITORADOS

Os clientes monitorados do sistema devem instalar uma aplicação em seus smartphones responsável por enviar dados de posicionamento. A aplicação móvel utilizada nesse trabalho é gratuita, de código aberto, disponível para as plataformas Android e iOS e pode ser baixada através do seguinte link:

http://www.traccar.org/client/

Para utilizar essa aplicação com a sua solução, você deve configurá-la. Para isso, você deve alterar os seguintes campos detalhados na figura abaixo:

- *Server address* – ip do gateway
- *Server port* – porta do gateway
- *Frequency* – período em segundos que a aplicação enviará dados de posicionamento



Para iniciar/encerrar o envio de mensagens utilize o campo *Service status*.

Além da aplicação móvel, também será disponibilizado no *moodle* o código fonte de um simulador, capaz de simular conexões de clientes remotos. Esse simulador poderá ser utilizado para validar sua implementação e testar se a solução proposta suporta um grande número de clientes conectados concorrentemente (deve suportar no mínimo 100).

O gateway é um servidor TCP/IP que espera por conexões remotas em um determinado *ip* e porta conhecidos pela aplicação móvel dos clientes monitorados. Os clientes conectam no gateway e mantem a conexão aberta durante sua execução.

A comunicação entre o gateway e os clientes remotos é realizada através do protocolo HTTP (*https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol*).

Segue a sequência típica de comunicação entre o gateway e o cliente:

1. O gateway inicializa e espera por conexões em um determinado ip/porta.
2. Cada cliente que deseja ser monitorado conecta ao gateway.
3. Uma vez conectado, o cliente envia periodicamente uma mensagem de posicionamento, contendo:
   - data e hora
   - posicionamento do cliente (latitude e longitude)
   - velocidade
   - estado de funcionamento da aplicação móvel cliente.
4. O gateway recebe a mensagem de posicionamento, decodifica-a, atualiza a entrada correspondente ao cliente na lista de clientes ativos em memória compartilhada e repassa a informação ao historiador.

Caso a conexão com o cliente remoto seja interrompida, devido à falta de conectividade ou problemas na rede, a lista em memória compartilhada de clientes ativos deve ser atualizada, removendo o cliente em questão.

2.1 DESCRIÇÃO DAS MENSAGENS DO PROTOCOLO

Essa seção detalha as mensagens utilizadas na comunicação entre os clientes e o gateway.

2.1.1 - Mensagem de Posicionamento

Essa mensagem é enviada periodicamente pela aplicação cliente para reportar informações referente ao posicionamento.

A mensagem possui o seguinte formato (as partes da mensagem delimitadas por <>, por exemplo <POS>, são variáveis, ou seja, variam para cada mensagem recebida):

```
GET /?<POS> HTTP/1.1\r\n
User-Agent: <USER_AGENT>\r\n
Host: <IP>:<PORT>\r\n
Connection: Keep-Alive\r\n
Accept-Encoding: gzip\r\n\r\n
```

A mensagem recebida é dividida nas seguintes partes:

- Linha da requisição – a linha que começa com GET;
- Campos de cabeçalho da requisição – linhas de 2 a 4;
- Uma linha em branco (“\r\n”)

Você deverá apenas analisar a **Linha da Requisição,** as outras partes da mensagem podem ser ignoradas. Além disso, note que as linhas da mensagem são separadas pelos caracteres "\r\n".

Você deverá analisar o campo POS da Linha de Requisição, esse campo é denominado *Query String (https://en.wikipedia.org/wiki/Query_string*) no protocolo HTTP.

O formato do campo POS é definido como:

```
id=<IMEI>&timestamp=<DT>&lat=<LAT>4&lon=<LONG>&speed=<VEL>&bearing=<ROL>&al
titude=<ALT>&batt=<BATT>
```

Sendo que:

- IMEI – identificador do cliente (int)
- DT – *string* contendo data e hora no formato *Unix Timestamp*. Em C/C++ equivale ao tipo *time_t* (https://en.wikipedia.org/wiki/C_date_and_time_functions#time_t)
- LAT – latitude (double)
- LONG– longidude (double)
- VEL - velocidade em km/h (int)
- ROL – rolamento (int)
- ALT – Altitude (int)
- BATT – estado da bateria em percentual (double)

Os campos ROL, ALT e BATT não serão utilizados.

Exemplo de mensagem:

```
GET /?id=189740&timestamp=1447347829&lat=-19.86979471&lon=-
43.96226144&speed=0.0&bearing=0.0&altitude=819.0&batt=91.0 HTTP/1.1\r\n
User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0; Nexus 5 Build/MRA58N)\r\n
Host: 150.164.35.70:9001\r\n
Connection: Keep-Alive\r\n
Accept-Encoding: gzip\r\n\r\n
```

A mensagem acima corresponde a uma mensagem de posicionamento criada às 15:03:49 do dia 12/11/2015 e os seguintes campos devem ser extraídos

- Identificador do Cliente (IMEI): 189740;
- Localização (lat/long): =-19.86979471=-43.96226144;
- Velocidade: 0 km/h;

2.1.2 – Resposta da Mensagem de Posicionamento

O gateway deve enviar uma resposta para cada mensagem de posicionamento recebida no seguinte formato:

```
HTTP/1.1 200 OK\r\n
Content-Length: 0\r\n
Keep-Alive: timeout=15,max=100\r\n\r\n
```

3. DESCRIÇÃO DO PROTOCOLO DE COMUNICAÇÃO ENTRE O GATEWAY E HISTORIADOR

A comunicação entre o gateway e o historiador é realizada através de uma fila de mensagens entre os dois processos. O nome da fila de mensagens é "gateway_historiador".

A comunicação segue o padrão produtor/consumidor, em que o gateway age como produtor de mensagens e o historiador como consumidor. Cabe ao historiador criar a fila de mensagens. O gateway deve apenas abri-la.

O gateway deve repassar ao historiador todas as informações e posicionamento recebidas dos clientes remotos.

3.1 DESCRIÇÃO DAS MENSAGENS DO PROTOCOLO

Essa seção detalha o formato da mensagem utilizada na comunicação entre o gateway e o historiador. Essa mensagem é definida como um *struct* que contem as informações de posicionamento e estado de operação de um determinado cliente. Cabe ao gateway converter as mensagens textuais, recebidas dos clientes remotos, para o *struct* e então enviá-las ao historiador via fila de mensagens.

O código abaixo detalha a mensagem:

```
struct position_t
{
   int id;
   time_t timestamp;
   double latitude;
   double longitude;
   int speed;
};
```

4. DESCRIÇÃO DO PROTOCOLO DE COMUNICAÇÃO ENTRE O HISTORIADOR E O SERVIDOR DE APLICAÇÃO

A comunicação entre o historiador e o servidor de aplicação também é realizada via filas de mensagens. Porém, são utilizadas duas filas de mensagens. A primeira, denominada "servapp_historiador", é utilizada para que o servidor de aplicação envie pedidos de dados históricos ao historiador. A segunda, denominada "historiador_servapp", é utilizada para que o historiador envie os resultados dos pedidos.

O historiador é responsável por criar as duas filas. O servidor de aplicação deve apenas abri-las na sua inicialização.

A comunicação entre esses dois módulos é definida seguindo o padrão requisição/resposta, ou seja, o servidor de aplicação envia um pedido de dados históricos ao historiador utilizando a fila de mensagens "servapp_historiador". O historiador, espera por mensagens nessa fila e, ao receber uma mensagem, lê os dados históricos correspondentes e os envia ao servidor de aplicação, utilizando a fila "historiador_servapp".

4.1 DESCRIÇÃO DAS MENSAGENS DO PROTOCOLO

Essa seção detalha as mensagens utilizadas na comunicação entre o servidor de aplicação e o historiador. Duas mensagens são definidas: uma mensagem enviada pelo servidor de aplicação ao historiador para requisitar dados históricos e uma mensagem com o resultado da requisição, enviada pelo historiador ao servidor de aplicação.

As duas mensagens são definidas como *structs*, conforme o código abaixo:

```
static const unsigned MAX_POSITION_SAMPLES = 30;

struct historical_data_request_t
{
  int id;
  int num_samples;
};

struct historical_data_reply_t
{
  int num_samples_available;
  position_t data[MAX_POSITION_SAMPLES];
};
```

O *struct historical_data_request_t* é utilizado pelo servidor de aplicação para enviar pedidos de dados históricos ao historiador. O servidor de aplicação deve preencher esse *struct* com o identificador do cliente desejado, assim como número de posições desejadas (campo *num_samples*). Note que existe um limite de no máximo MAX_POSITION_SAMPLES=30 que podem ser requisitadas.

O *struct historical_data_reply_t* contêm a resposta do historiador, o campo *num_samples_available* contêm o número de amostras retornadas (que pode ser menor ou igual ao valor requisitado). As informações de posicionamento do cliente estão presentes no vetor *data*.

Note que a fila de mensagens "servapp_historiador" contêm mensagens do tipo *historical_data_request_t* e a fila "historiador_servapp" mensagens do tipo *historical_data_reply_t*.

5. DESCRIÇÃO DO PROTOCOLO DE COMUNICAÇÃO ENTRE O SERVIDOR DE APLICAÇÃO E A INTERFACE GRÁFICA

A comunicação entre o servidor de aplicação e a interface cliente é realizada via rede (protocolo TCP/IP). O servidor de aplicação age como um servidor TCP/IP que espera por conexões remotas das interfaces clientes.

A interface cliente conecta ao servidor de aplicação e pode requisitar:

- lista de clientes ativos;
- dados históricos de um cliente em particular.

Toda vez que o servidor de aplicação recebe um pedido de lista de clientes ativos, esse deve acessar esses dados disponíveis na região de memória compartilhada com o gateway e formatá-los seguindo o protocolo textual definido na seção 5.1 para que possam ser repassados à interface cliente.

Caso o servidor de aplicação receba pedidos de dados históricos de um determinado cliente, esse pedido deve ser repassado ao historiador, utilizando o protocolo de comunicação definido na seção 4. Ao receber a resposta do historiador, as informações em formato binário devem ser convertidas para o formato descrito na seção 5.1, antes de serem repassadas à interface cliente.

5.1 DESCRIÇÃO DAS MENSAGENS DO PROTOCOLO

Essa seção detalha as mensagens utilizadas na comunicação entre a interface gráfica e o servidor de aplicação.

*Para facilitar a comunicação, todas as mensagens enviadas e recebidas dos utilizam o caractere **'\n'** como delimitador de final de mensagem, ou seja, todas as mensagens transmitidas devem terminar com um **'\n'**.*

5.1.1 REQ_ATIVOS - Mensagem de Requisição de lista de clientes ativos

A interface gráfica envia essa mensagem ao servidor para requisitar a lista de clientes monitorados conectados ao sistema.

Formato:

```
REQ_ATIVOS
```

5.1.2 ATIVOS - Mensagem de Listagem de Clientes Ativos

O servidor envia essa mensagem para a interface gráfica, como resposta à mensagem REQ_ATIVOS, contendo a lista de clientes ativos. A lista de clientes ativos deve ser obtida a partir da região de memória compartilhada entre o servidor de aplicação e o gateway (ver seção 6 para mais detalhes).

```
ATIVOS;<NUM>;<ID1>;<ID1>;<ID2>;<ID2>;...;<IDN>
```

- NUM – número de cliente ativos
- IDi – Identificador do cliente i

Dado um bug no código da interface cliente, o identificador de cada cliente deve ser enviado duas vezes!

Exemplo:

```
ATIVOS;2;015225;015225;115389;115389
```

Dois clientes ativos, o primeiro com identificador `015225` e o segundo com identificador `115389.`

Caso não existam clientes ativos, o servidor deve enviar a seguinte mensagem:

```
ATIVOS;0
```

5.1.3 REQ_HIST - Mensagem de Requisição de Dados Históricos do Cliente

A interface gráfica envia essa mensagem para o servidor de aplicação, requisitando dados históricos de um cliente monitorado.

Caso a interface gráfica requisite apenas uma amostra, o servidor de aplicação deve buscar essa informação na lista de clientes ativos, presente na memória compartilhada entre o servidor de aplicação e o gateway (ver seção 6 para mais detalhes). Caso contrário, ele deve requisitar essa informação do historiador (seção 4).

Formato:

```
REQ_HIST;<ID>;<NUM_AMOSTRAS>
```

- ID – identificador do cliente
- NUM_AMOSTRAS – número de amostras de posicionamento a serem enviadas. O servidor deve enviar as NUM_AMOSTRAS mais recentes.

Exemplo:

```
REQ_HIST;2;5
```

Requisita as últimas 5 amostras de posicionamento do cliente com id 2.

5.1.4 HIST - Mensagem de Envio de Dados Históricos

O servidor envia essa mensagem para a interface gráfica, como resposta à mensagem REQ_HIST, contendo os dados históricos requisitados.

Formato:

```
HIST;<NUM>;<ID>;<POS1>;<POS2>;...;<POSN>
```

- <NUM> - número de amostras de posicionamento (note que o número de amostras enviadas não necessariamente deve ser igual ao número requisitado. Por exemplo, a interface gráfica pode requisitar 100 amostras, mas pode haver apenas 50 no arquivo de dados).
- ID – identificador do cliente
- <POSi> - mensagem com o formato da mensagem de posicionamento POS, contendo a amostra de posicionamento i. As mensagens de posicionamento recebidas devem estar na sequência da mais recente à mais antiga.

A mensagem de posicionamento possui a seguinte estrutura:

POS;<DT>;<LAT>;<LON>;<VEL>;<ESTADO>

- DT – data e hora, string no formato DDMMAAAAHHMMSS
- LAT – latitude (double)
- LONG– longitude (double)
- VEL - velocidade (em km/h)
- ESTADO – 0 = Cliente Offline / 1= Cliente Online

Exemplo de mensagem do tipo POS:

POS;03072013145532;-49.1232;-19.3434;100;1

Mensagem de posicionamento registrada às 14:55:32 do dia 03/07/2013 com os seguintes campos:

- Localização (lat/long): -49.123232/-19.343434;
- Velocidade: 100km/h;
- Estado: Online

O campo ESTADO poderá ser sempre enviado com o valor igual a 1.

Caso não existam dados históricos para o cliente selecionado, o servidor deve enviar a seguinte mensagem:

```
HIST;0
```

5. DESCRIÇÃO DA COMUNICAÇÃO ENTRE O E O GATEWAY E O SERVIDOR DE APLICAÇÃO

Conforme discutido, o gateway e o servidor de aplicação compartilham uma região de memória que contém informações referentes aos clientes conectados ao sistema.

Essa região de memória compartilhada, denominada “UsuariosAtivos”, deve conter uma lista de clientes atualmente conectados ao sistema, assim como sua última informação de posicionamento e estado de operação.

A seguinte estrutura de dados será utilizada para representar a lista de clientes ativos:

```
struct active_users_t
{
  active_users_t() : num_active_users(0)
  {
    for (unsigned i=0; i< LIST_SIZE; ++i)
    {
      list[i].id = -1;
    }
  }
  int num_active_users;
  enum { LIST_SIZE = 1000000};
  position_t list[LIST_SIZE];
  boost::interprocess::interprocess_mutex mutex;
};
```

A estrutura de dados acima é composta por:

- um contador de clientes ativos (campo *num_active_users*);
- uma lista do tipo *position_t*, com no máximo LIST_SIZE entradas;
- um *mutex* utilizado para sincronização de acessos a lista.

A lista de clientes ativos é indexada pelo identificador do cliente, ou seja, por exemplo, o cliente com o identificador 13 estará na posição 13 da lista. Para identificar se um dado cliente está conectado ou não, basta verificar o valor do campo *id* referente a sua posição na lista. Caso o valor seja -1, o cliente não está conectado. Caso contrário, o valor desse campo deve corresponder ao identificador do cliente.

As seguintes operações de atualização dessa estrutura devem ser realizadas pelo gateway:

- alterar o valor do campo *id* para o id do cliente recém conectado toda vez que um novo cliente conectar, ou seja, após receber a mensagem de posicionamento (seção 2.1.1);
- alterar o valor do campo para -1 toda vez que um cliente desconectar;
- alterar o valor do *struct position_t* toda vez que um cliente enviar dados de posicionamento, ou seja, após receber a mensagem de posição (seção 2.1.2).

O servidor de aplicação acessa esta estrutura para obter a lista de clientes ativos (mensagem REQ_ATIVOS – seção 5.1.1) e para obter o último registro de posicionamento de um dado cliente (mensagem REQ_HIST com NUM_AMOSTRAS igual a 1 – seção 5.1.3)

Para que o servidor de aplicação possa obter a lista de cliente ativos, este deve percorrer essa estrutura de dados e, para cada posição, verificar o valor de *id*.

7. INSTRUÇÕES

O trabalho deve ser desenvolvido em C++ utilizando as bibliotecas de comunicação via rede e comunicação entre processos vistas em sala de aula. O trabalho envolve os

conceitos de gerenciamento de threads, sincronização, entrada e saída e comunicação entre processos.

Cada grupo será composto por *seis* integrantes, divididos em *três* duplas. Cada dupla é responsável por uma aplicação. A avaliação do trabalho será realizada por cada dupla (80% dos pontos) e por todo o grupo (20% dos pontos).

A avaliação do trabalho consistirá em uma apresentação. A apresentação será realizada no último dia de aula e cada grupo deverá apresentar a solução desenvolvida e serão arguidos pelo professor.

No dia da apresentação, você deve entregar (via *moodle*) o código fonte e os arquivos de projeto,

Os executáveis da interface gráfica e o simulador de cliente estarão disponíveis no *moodle*.