INSTITUTO FEDERAL DE SANTA CATARINA

AMELIZA SOUZA CORREA
CAMILLA BARRETO DE SOUSA

# PROJETO DE PROTOCOLOS
# MANUAL - PROTOCOLO DE APLICAÇÃO

São José - SC

abril/2021

# SUMÁRIO

# 1 DESCRIÇÃO

O protocolo de aplicação desenvolvido é baseado em TCP e se enquadra no modelo cliente-servidor, com interações do tipo pedido-resposta. O serviço oferecido pelo protocolo é o intercâmbio de provas completas, bem como as respostas e resultados, onde o usuário deve ser autenticado no início da comunicação. Foi desenvolvida uma API em *python* para o uso do protocolo, além de um programa de demonstração.

No Capítulo 2 há o esclarecimento da utilização da API para que a mesma possa ser usada para outros projetos e no Capítulo 3 há a explicação de como utilizar a aplicação de demonstração implementada para auxiliar no uso. O Apêndice A detêm as definições e sintaxes das mensagens do protocolo. Por fim, o Apêndice B trás os links importantes, direcionando para o repositório onde o código está disponível.

# 2 MANUAL API

A API desenvolvida realiza uma comunicação entre cliente e servidor por meio da conexão de *sockets* TCP e as mensagens trocadas são serializadas e estruturadas via *Protocol Buffers*. Para construir um objeto da API é necessário passar os parâmetros para a conexão via *socket*:

- `API_APP(ip_connect, port_connect, ip_bind, port_bind)`

. Os métodos oferecidos pela API para o desenvolvimento de aplicações são descritos a seguir.

## 2.1 Métodos

Os métodos da API sempre retornam a tupla (`ack, msg`), onde `ack` é do tipo `bool` e indica `True` para uma requisição bem sucedida e `False` se houve falha e o tipo do dado `msg` depende do método chamado.

### 2.1.1 Login

Antes do intercâmbio de provas, o usuário deve se autenticar com o servidor através do método `login(usuario, senha)`, onde deve informar usuário e senha. Dependendo do sucesso ou falha da requisição, o dado `msg` representa:

- `ack = True` $\longrightarrow$ `msg` é um dado vazio.
- `ack = False` $\longrightarrow$ `msg` é um dado do tipo `string` com os detalhes da falha (opcionalmente: vazio).

### 2.1.2 Requisição de prova

Para fazer a requisição de prova deve ser chamado o método `reqprova(idprova)`, onde deve ser informado o identificador da prova de interesse. Dependendo do sucesso ou falha da requisição, o dado `msg` representa:

- `ack = True` $\longrightarrow$ `msg` é uma lista com dados do tipo `QUESTAO`.
- `ack = False` $\longrightarrow$ `msg` é um dado do tipo `string` com os detalhes da falha (opcionalmente: vazio).

### 2.1.3 Envio das respostas

O envio de respostas deve ser feito através do método `reqresp(idprova, respostas)`, onde deve ser passado como parâmetro uma lista de respostas e o identificador da prova correspondente. Independente do sucesso ou falha da requisição, o dado `msg` representa:

- `ack = True / False` $\longrightarrow$ `msg` é um dado do tipo `string` com os detalhes da comunicação (opcionalmente: vazio).

### 2.1.4 Requisição do resultado

O método `reqresultado(idprova)` é usado para solicitar o resultado da prova especificada no parâmetro identificador. Dependendo do sucesso ou falha da requisição, o dado `msg` representa:

- `ack = True` ⟶ `msg` é uma lista com dados do tipo `RESULTADO`.
- `ack = False` ⟶ `msg` é um dado do tipo `string` com os detalhes da falha (opcionalmente: vazio).

### 2.1.5 Logout

Para desconectar do servidor o método `logout()` é usado. Independente do sucesso ou falha da requisição, o dado `msg` é:

- `ack = True / False` ⟶ `msg` é um dado do tipo `string` com os detalhes da comunicação (opcionalmente: vazio).

# 3 APLICAÇÃO DE DEMONSTRAÇÃO

A aplicação de demonstração implementada consiste em um menu, onde o aluno pode decidir as ações que deseja executar, de acordo com sua necessidade. Essas funcionalidades estão descritas abaixo e são executadas através da interface de aplicação.

## 3.1 Execução

Para a utilização do protocolo na mesma máquina que a aplicação, é necessário executar primeiro o servidor depois a interface do cliente. Assim como demonstrado abaixo:

```
usuario@pc$ python3 servidor.py
    Esperando conexão...

usuario@pc$ python3 interface.py
    ------ MENU ------
    1 - Login
    2 - Requisição de prova
    3 - Responder prova
    4 - Resultado da prova
    5 - Logout
```

## 3.2 Utilização

A sequência em que as opções aparecem neste menu deve ser seguida para o bom funcionamento da aplicação e para cada uma, há a necessidade do aluno informar parâmetros úteis para a validação das requisições perante o servidor, assim como pode ser visto abaixo:

- **Login**: Usuário e senha do aluno
- **Requisição de prova**: ID da prova para visualizar as questões
- **Responder prova**: ID da prova que está sendo respondida
- **Resultado da prova**: ID da prova que deseja o resultado
- **Logout**: Nenhum parâmetro precisa ser informado

## 3.3 Valores estáticos

Alguns registros tanto no servidor quanto na interface do cliente estão estáticos para facilitar a demonstração da API. Esses dados e seus respectivos valores são:

- **SERVIDOR**

```
ip_bind = "0.0.0.0"
port_bind = 8888
```

```
usuario = "aluno"
senha = "aluno"
token = "378rbf9sd"
id_prova = "1"
nota = 10
questoes = [
                id: 654
                enunciado: "Qual a capital de Santa Catarina?"
                pontos: 5
                alternativas {
                  descricao: "(a) Sao Paulo"
                  codigo: "111"
                }
                alternativas {
                  descricao: "(b) Florianopolis"
                  codigo: "222"
                }
                alternativas {
                  descricao: "(c) Sao Paulo"
                  codigo: "333"
                }
                , id: 987
                enunciado: "Qual a capital de Pernambuco?"
                pontos: 5
                alternativas {
                  descricao: "(a) Recife"
                  codigo: "111"
                }
                alternativas {
                  descricao: "(b) Aracaju"
                  codigo: "222"
                }
                alternativas {
                  descricao: "(c) Goiania"
                  codigo: "333"
                }
                , id: 321
                enunciado: "Qual a utilidade da Rosa dos Ventos?"
                pontos: 4
            ]
resultados =
            id_prova: "1"
            nota: 10
            questoes {
              questao: 987
              pontos: 3
            }
```

```
    questoes {
      questao: 654
      pontos: 3
    }
    questoes {
      questao: 321
      pontos: 4
    }
```

- **INTERFACE**

```
ip_connect = "127.0.0.1"
port_connect = 8888
ip_bind = "0.0.0.0"
port_bind = 0
respostas =
          token: "378rbf9sd"
          id_prova: "1"
          respostas {
            id: 654
            codigos {
              codigos: "222"
            }
          }
          respostas {
            id: 987
            codigos {
              codigos: "111"
            }
          }
          respostas {
            id: 321
            texto: "A Rosa dos Ventos eh um instrumento antigo
                    utilizado para auxiliar na localizacao relativa."
          }
```

# A MENSAGENS DO PROTOCOLO

Nesse apêndice foram reunidas as definições das mensagens do protocolo de aplicação, bem como a sintaxe e os tipos de dados que as compõem.

## A.1 Definição

A Tabela 1 mostra os tipos de mensagens definidas pelo protocolo e qual a origem da comunicação.

| **Mensagem** | **Descrição** | **Origem** |
|---|---|---|
| LOGIN | Pedido de autenticação | Cliente |
| ACKLOGIN | Confirmação da autenticação | Servidor |
| REQPROVA | Solicitação de uma prova | Cliente |
| ACKREQPROVA | Transmissão da prova solicitada | Servidor |
| REQRESP | Envio de respostas da prova | Cliente |
| REQRESULTADO | Solicitação do resultado da prova | Cliente |
| ACKREQRESULTADO | Resultado da prova | Servidor |
| STATUS | Status do pedido | Servidor |
| LOGOUT | Pedido de desconexão | Cliente |

Tabela 1 – Descrição das mensagens do protocolo de aplicação

## A.2 Sintaxe

As mensagens da API foram estruturadas via *Protocol Buffers*. Segue abaixo a sintaxe de cada uma das mensagens:

- MENSAGEM

```
message MENSAGEM {
  oneof msg {
    LOGIN login = 1;
    LOGOUT logout = 2;
    ACK_LOGIN acklogin = 3;
    STATUS status = 4;
    REQ_PROVA reqprova = 5;
    ACK_REQ_PROVA ackreqprova = 6;
    REQ_RESULTADO reqresultado = 7;
    ACK_REQ_RESULTADO ackreqresultado = 8;
    REQ_RESP reqresp = 9;
  }
}
```

- LOGIN

```
message LOGIN {
    required string login = 1;
    required string senha = 2;
}
```

- ACK_LOGIN

```
message ACK_LOGIN {
  optional string token = 1;
  required STATUS status = 2;
}
```

- STATUS

```
message STATUS {
  required int32 codigo = 1;
  optional string descricao = 2;
}
```

- REQ_PROVA

```
message REQ_PROVA {
  required string token = 1;
  optional string id_prova = 2;
}
```

- ACK_REQ_PROVA

```
message ALTERNATIVA {
  required string descricao = 1;
  required string codigo = 2;
}

message QUESTAO {
  required int32 id = 1;
  required string enunciado = 2;
  optional int32 pontos = 3;
  repeated Alternativa alternativas = 4;
}

message ACK_REQ_PROVA {
  required string id_prova = 1;
  repeated QUESTAO questoes = 2;
}
```

- REQ_RESP

```
message CODIGOS {
    repeated string codigos = 1;
}

message RESPOSTA {
  required int32 id = 1;
  oneof resp {
    string texto = 2;
    CODIGOS codigos = 3;
  }
}

message REQ_RESP {
  required string token = 1;
  required string id_prova = 2;
  repeated RESPOSTA respostas = 3;
}
```

- REQ_RESULTADO

```
message REQ_RESULTADO {
  required string token = 1;
  required string id_prova = 2;
}
```

- ACK_REQ_RESULTADO

```
message RESULTADO {
  required int32 questao = 1;
  required int32 pontos = 2;
}

message ACK_REQ_RESULTADO {
  required string id_prova = 1;
  required int32 nota = 2;
  repeated RESULTADO questoes = 3;
}
```

- LOGOUT

```
message LOGOUT {
  required string token = 1;
}
```

# B CÓDIGOS

Link para o repositório contendo os códigos citados no manual:

- Repositório

- API do protocolo