Projeto final de Sistemas Operacionais

Para visualizar esse arquivo é recomendado usar o navegador Google Chrome com a extensão Markdown Viewer, será necessário ir nas configurações da extensão e na área de permissões para autorizar o acesso a arquivos

Definiçoes de Projeto

Tabela 1: vlores a serem passados para a API /set

Direção	Valor
frente	1
esquerda	2
direita	3

Servidor flask

Por ser baseado em python a framework flask foi escolhida devido a sua simplicidade de implementação. de modo que com poucas linhas foi possível implementar uma serie de APIs REST para o projeto

dependencias

- Python 3.7
- configurar as variáveis de ambiente para incluir os diretorios onde estão localizados os arquivos python.exe e pip.exe
- biblioteca Flask

```
pip install flask
```

- biblioteca python-dotenv

```
pip install python-dotenv
```

Notas

> - o projeto foi desenvolvido com a IDE:
> PyCharm 2019.1.3 (Professional Edition) Build #PY-191.7479.30, built on May 29, 2019 Licensed to Luís Henrique Beltrão Santana Subscription is active until January 30, 2020 For educational use only. JRE: 11.0.2+9-b159.60 amd64 JVM: OpenJDK 64-Bit Server VM by JetBrains s.r.o Windows 10 10.0

- caso a IDE esteja aberta durante o processo de configuração das variáveis de ambiente será necessário reinicia-la
- para executar o servidor basta executar o script start_api.py
- Por padrão o servidor usa a porta 5000
- o servidor flask foi configurado via o script de inicialização para operar em toda a rede, o script de inicialização irá fornecer o ip da maquina em que o servidor está sendo executado via um print no terminal

###Testes

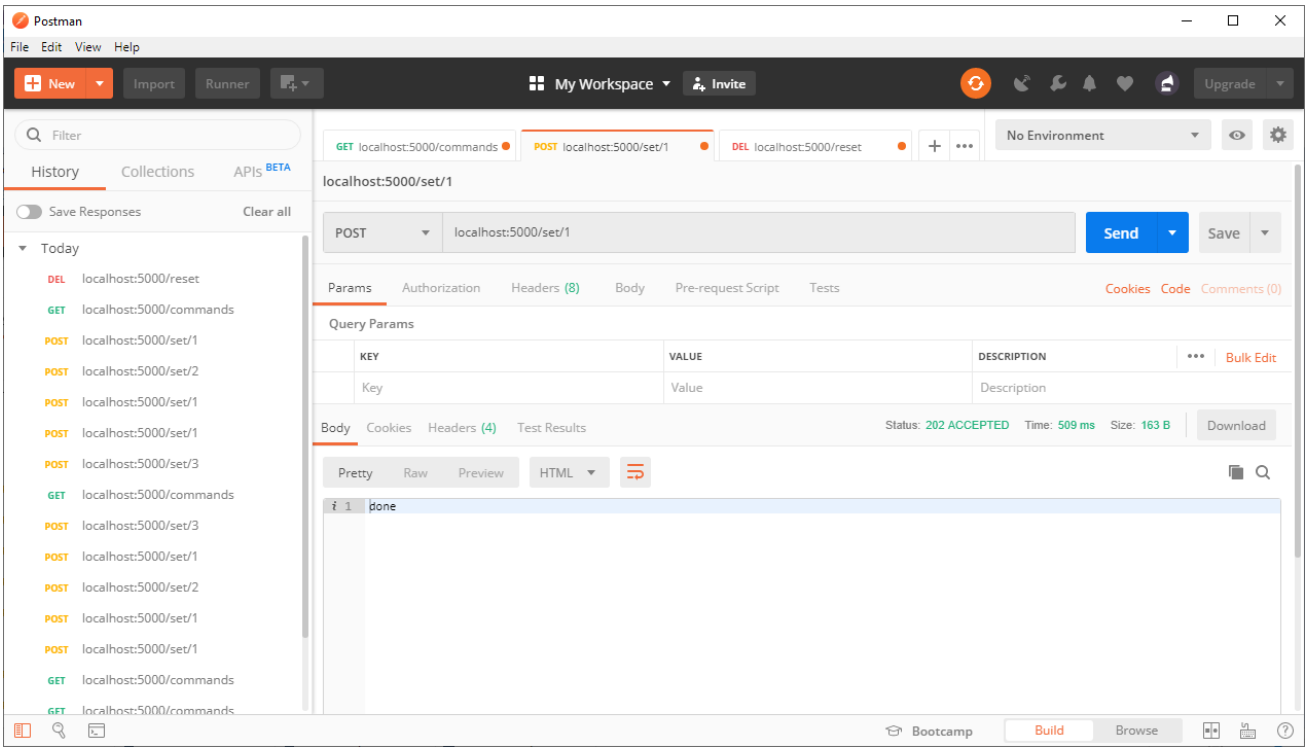

Figura 1: teste para a api de inserir dados

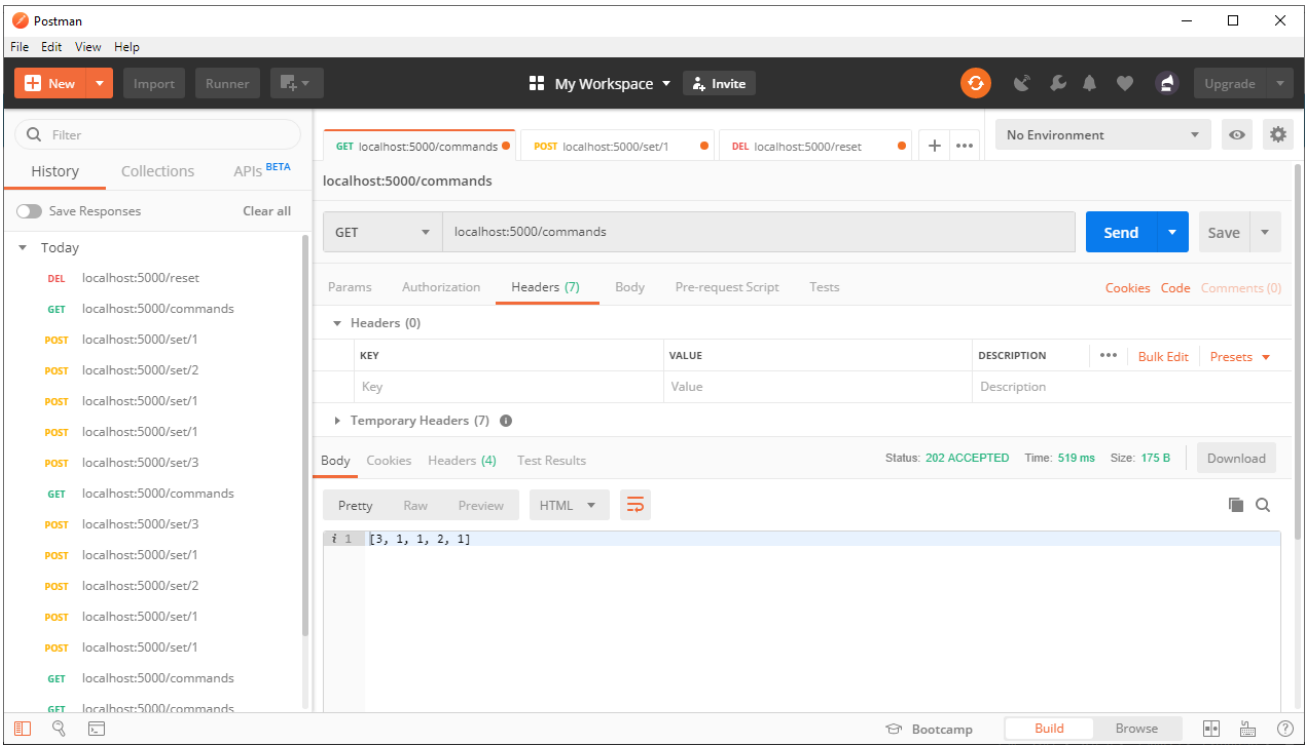

Figura 2: teste para a api de exibir a lista de comandos

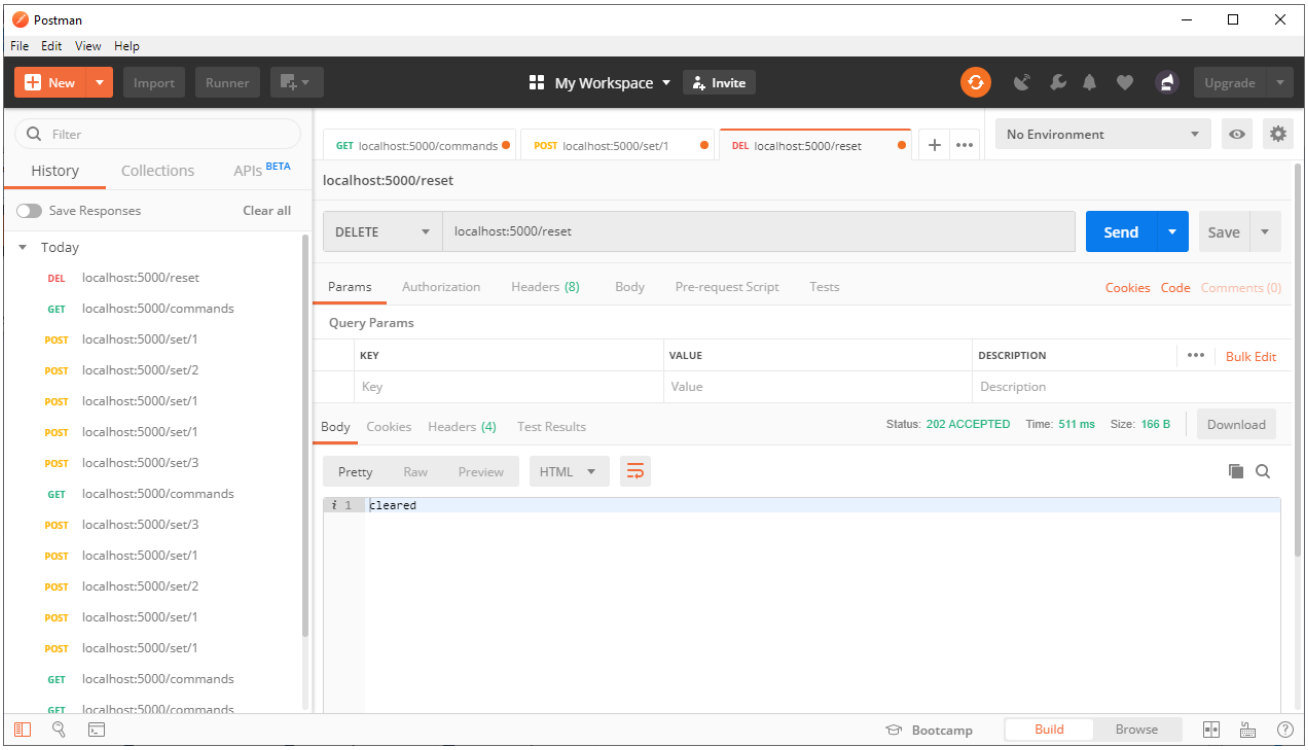

Figura 3: teste para a api de limpar os dados

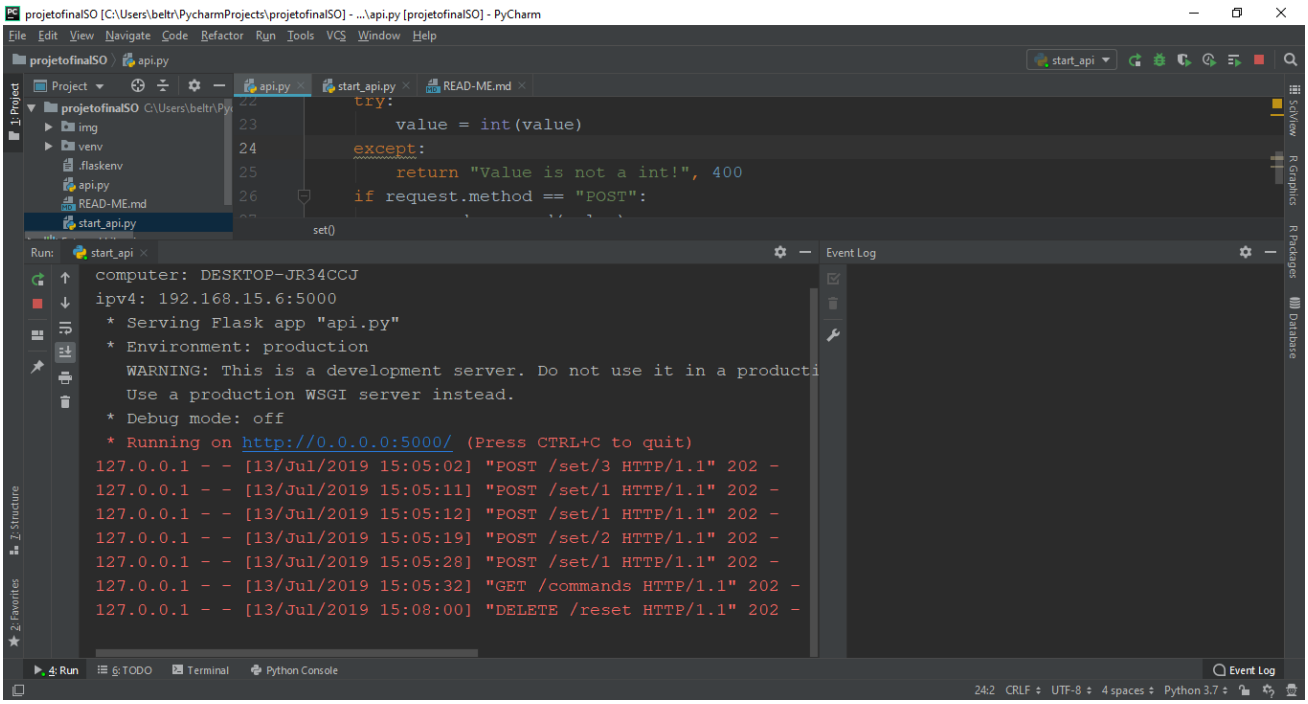

Figura 4: log do servidor

as figuras 1, 2 e 3 mostram as principais chamadas de API do projeto sendo testadas com o uso do postman, enquanto a Figura 4 mostra o servidor recebendo e tratando todas estas chamadas através do log de requisições do servidor Flask.
o último número, nas linhas de log, representa o código HTML de resposta que o servidor servidor atribuiu aquela requisição.

> ### Referencias
>
> set up
> flask: Hello World!
> methods
> REST codes

Angular

visando uma implementação resiliente e de boa performace foi excolhida a framework angular para a interação do usuário com o servidor, a codificação empregada é o typescript, que por ser uma linguagem fracamente tipada é bem resistente a

erros.
Outra vantagem do Angular é que a framework faz um bom uso da metodologia reativa que permite tratar evenjtos asincronos de maneira simplificada, uma vez que se entenda o paradigma de programação reativa

Dependencias

- NodeJS 10.16.0
- npm 6.9.0
- Angular 8.2.0
- Bootstrap 4.3.1

```
npm install bootstrap@4.3.1
```

Notas

> - O projeto foi desenvolvido com a IDE:
>   WebStorm 2019.1.3 Build #WS-191.7479.14, built on May 27, 2019 Licensed to Luís Henrique Beltrão Santana Subscription is active until January 30, 2020 For educational use only. JRE: 1.8.0_202-release-1483-b53 amd64 JVM: OpenJDK 64-Bit Server VM by JetBrains s.r.o Windows 10 10.0

- por padrão o servidor usa a porta 4200
- o servidor não estã configurado para operar em rede de modo que só irá funcionar no computador onde está sendo executado
- configurar o bootstrap no arquivo **angular.json** conforme demonstrado abaixo

```
(...): {
(...),
"styles": [
  "styles.scss",
  "node_modules/bootstrap/dist/css/bootstrap.min.css"
],
(...)
}
```

Testes

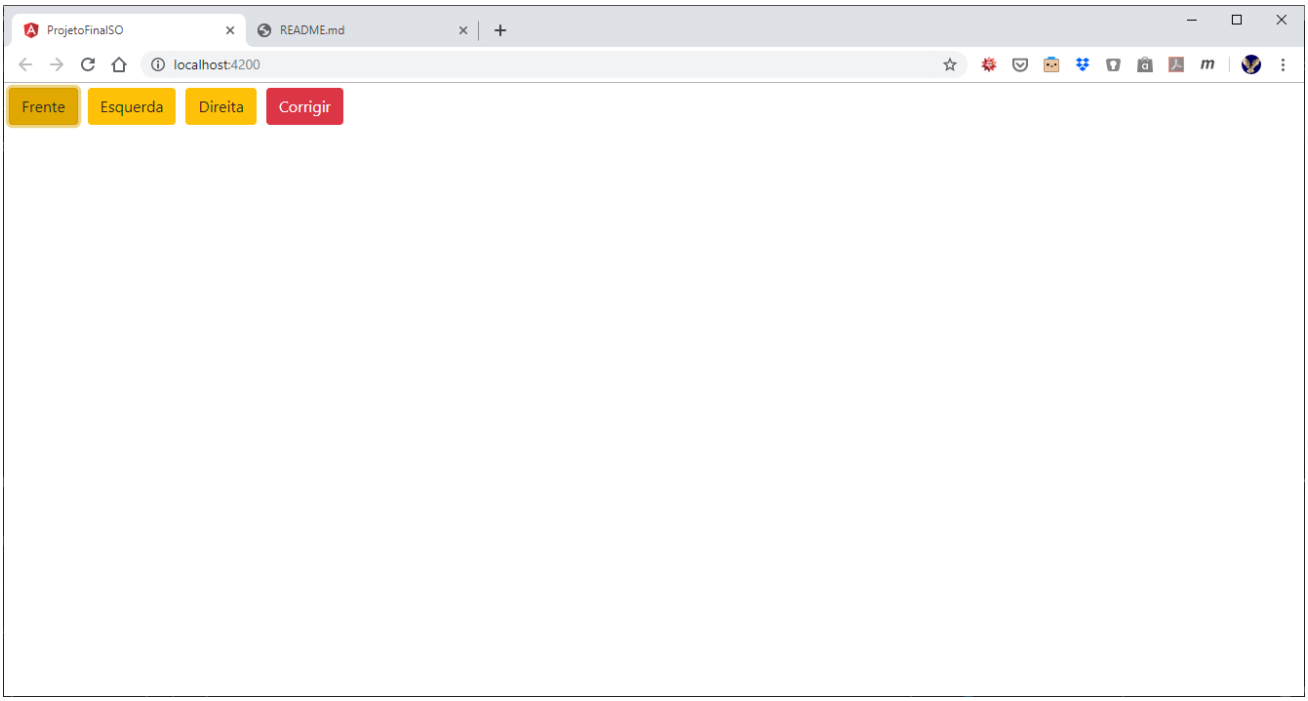

Figura 5 Página de controle

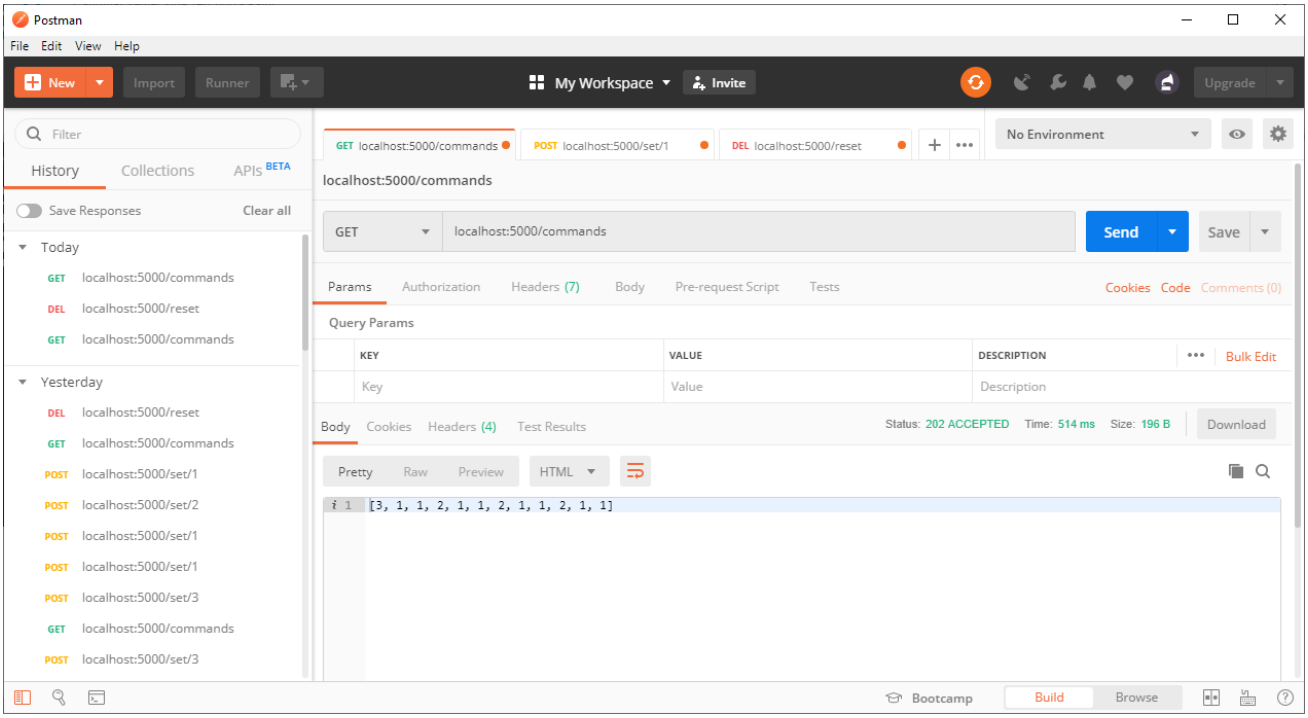

Figura 6 Postman simulando o teste da Conexão do Robô

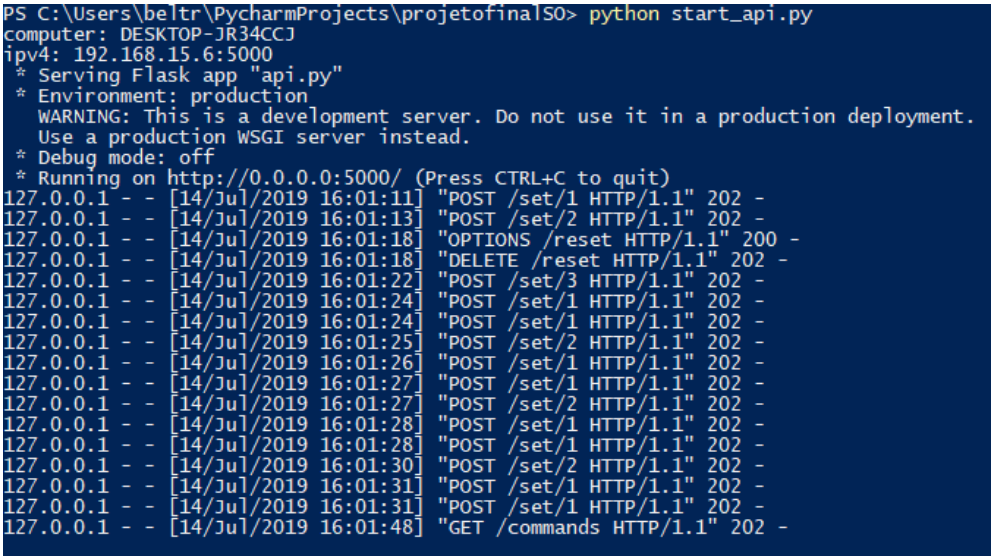

Figura 7 Log do servidor

A Figura 5 mostra a interface de comando do projeto implementada em angular enquanto a Figura 6 mostra uma chamada do postman que é exatamente igual a chamada que será realizada pelo robô.
A Figura 7 mostra o log do servidor sendo executado no powershelle serve de validação para os dados da Figura 6 que foram gerados empregando a interface da Figura 5.
Durante estes testes foi detectado que o navegador estava fazendo uma chamada de redirecionamento antes do delete por meio de um método REST "OPTIONS", que com a implementação basica original do servidor entrava em conflito com o CORS. Para corrigir esse problema foi necessário implementar uma regra de acesso especial para a API de delete conforme demonstrado no código a seguir:

```
from flask import Flask, request, Response
from flask_cors import CORS

app = Flask(__name__)
cors = CORS(app, resources={r"/reset": {"origins": "*"}})
(...)
```

Referencias

httpClientModule
bootstrap
Documentação bootstrap

TI-RTOS

Dependencias

- projeto exemplo HttpClient TIRTOS
- projeto exemplo Mutex TIRTOS
- TivaWare™ for C Series

Notas

> - O projeto foi desenvolvido com a IDE:
> Code Composer Studio
> Version: 9.1.0.201905231800
> Build id: N201905231800

- o binário que representa o id está agrupado de trás para frente de modo que cada Byte representa um inteiro do ip

```
00001001_00001111_10101000_11000000 = 9_15_168_192 = 192.168.15.9
```

- As requisições precisam ser enviadas com um intervalo entre o recebimento do ip por DHCP e o envio efetivo requisição de pelo menos 1s, caso cotrário a mensagem não chegará ao servidor e a Launchpad entra em um loop infinito devido ao erro ocorrido
- qualquer erro levará a Launchpad em um estado de loop infinito de NOP
- A biblioteca oficial não reconhece o código de sucesso 202, de modo que trata o mesmo como um erro, o código genérico 200 é devidamente reconhecido, desse modo as APIs que se comunicam com a Launchpad foram modificadas para retornar 200
- foi observado que a resposta da requisição gera um preenchimento disperso do vetor que recebe a resposta que teve a sua capacidade dobrada para suportar esse comportamento com um número razoável de comandos
- o projeto foi desenvolvido sobre o modelo HttpClient que gerou o projeto httpget
- é importante que a requisição só ocorra após a inicialização do sistema visto que cada requisição se baseia em uma thread que demandará recursos do sistema para ser devidamente executado, por padrão a biblioteca trata a chamada de requisição http antes da inicialização do sistema como um erro
- o sistema inicia efetivamente após a chamada da seguinte função:

```
BIOS_start();
```

- caso tenha sido realizado o processo de desbloqueio do ICDI devido a escrita indevida em um dos bits reservados da porta C o endereço MAC da placa precisará ser regravado nos endereços descritos no datasheet com a ferramenta LM Flash Programmer

###Testes

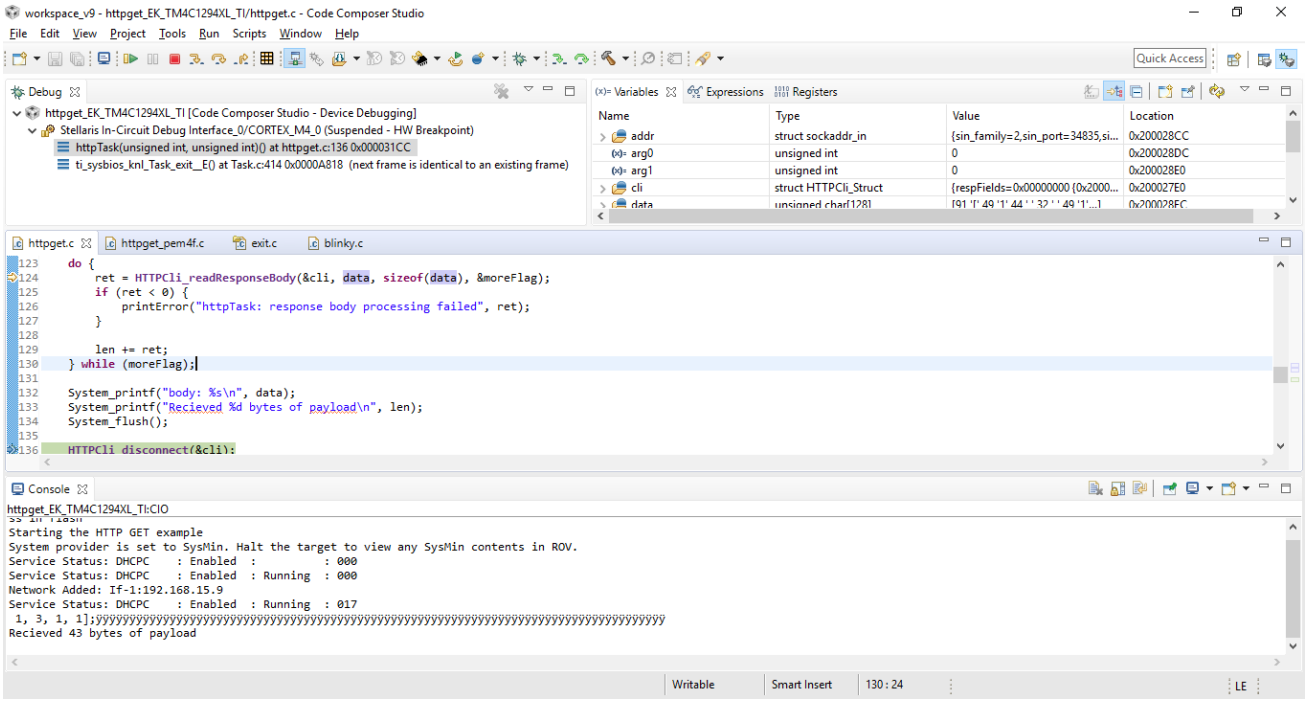

Figura 8 momento de recebimento da resposta

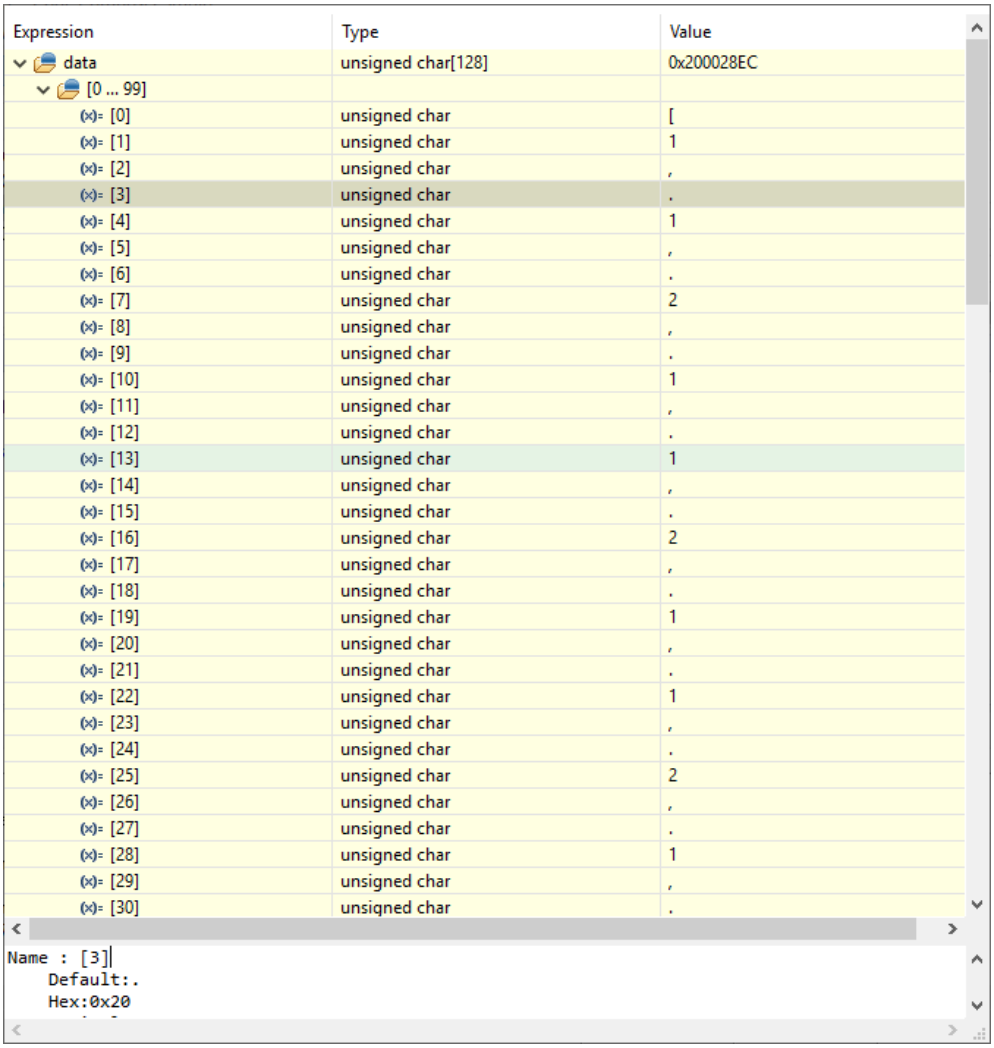

Figura 9 inspeção da resposta no debugger

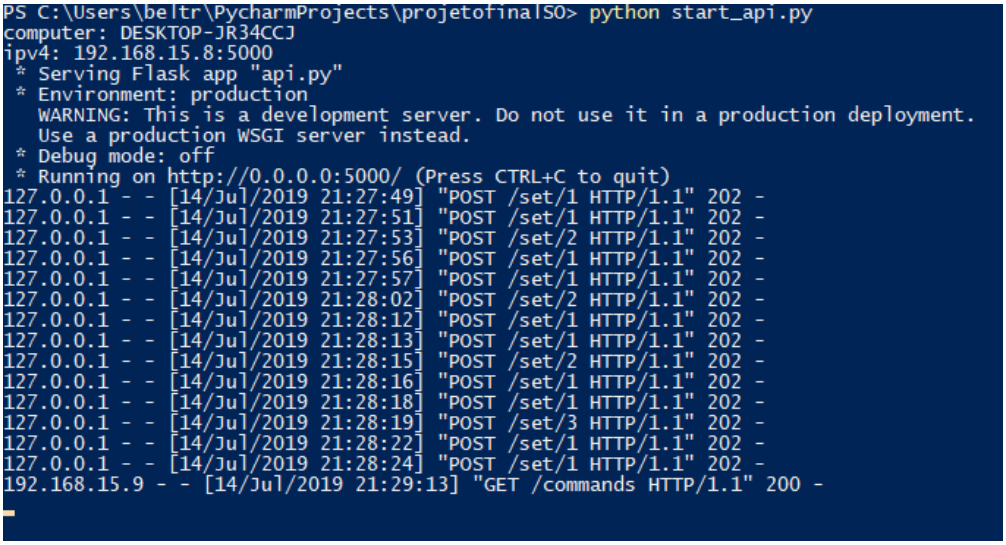

Figura 11 Log do servidor

Na figura 8 é possível observar que a mensagem foi efetivamente recebida, porém a mensagem está estranha com diversos caracteres desconhecidos após o final da mensagem que é marcada pelo ";".
Uma inspeção mais detalhada é bem visível na figura 9, onde esta sendo mostrado o conteúdo da resposta que está correto, de modo que a questão da exibição é puramente algum bug da biblioteca, por se tratar de um bag puramente estético não perceptível fora da IDE de desenvolvimento, o mesmo não interfere diretamente no desenvolvimento do projeto.
A figura 11 mostra o log do servidor onde é possível observar que os passos foram programados usando uma conexão local, no caso a página web do projeto, demarcada pelo ip 127.0.0.1, e por fim uma consulta foi realizada pelo ip 192.168.15.9 que conforme é possível confirmar na figura 8 se trata da requisição feita pela Launchpad