

## Manual - LUIZA LABS – Test

### Requisitos:

- IDE – Visual Studio 2017 (Community Version);
- Net Framework 4.6;
- Postman.

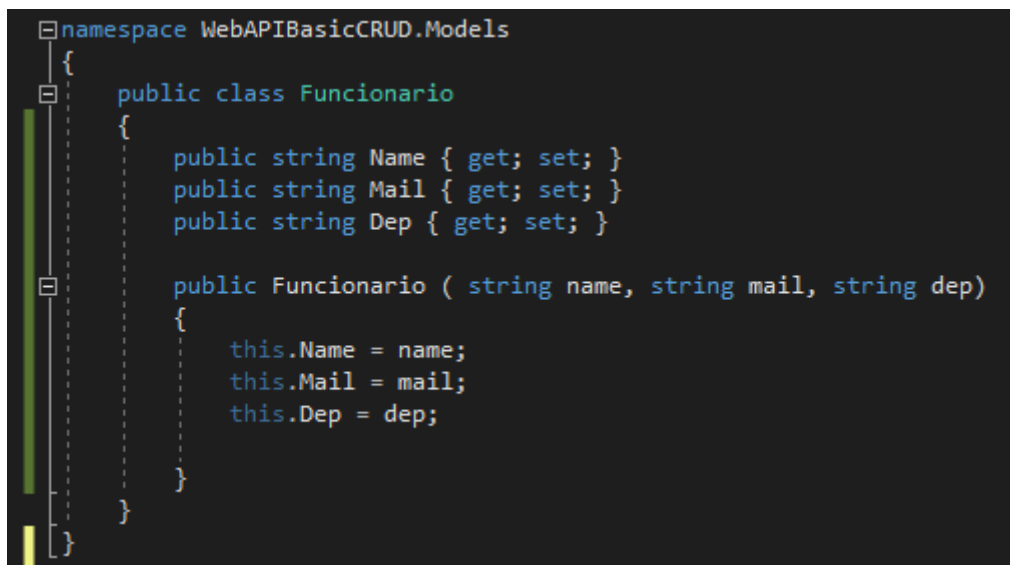
### API:

- A API a ser consumida no POSTMAN é: **localhost:49283/api/Funcionarios**;
- Antes de consumir a API no POSTMAN, inicie a aplicação para ver qual porta ela irá oferecer, após iniciada o navegador padrão irá inicializar a url no localhost com a porta. Pegue essa porta e substitua no valor da URL acima para o correto funcionamento.

### Passos para execução:

- Importar o código fonte abaixado do GITHUB e importar no Visual Studio;
- Clique no botão F5 para começar o start debug. O IIS Express irá subir a aplicação

### Processo de criação da API:



```
namespace WebAPIBasicCRUD.Models
{
    public class Funcionario
    {
        public string Name { get; set; }
        public string Mail { get; set; }
        public string Dep { get; set; }

        public Funcionario ( string name, string mail, string dep)
        {
            this.Name = name;
            this.Mail = mail;
            this.Dep = dep;
        }
    }
}
```

- Criou-se a classe Funcionario dentro do pacote MODEL, classe responsável por ter a estrutura do objeto funcionário.
- Criou-se uma classe controller denominada Funcionarios dentro do pacote Controllers, responsável pelas ações (chamadas da API) dentre elas:
  - Get : para listar todos os funcionários;
  - Post: para criar objetos funcionário com os seguintes atributos: Name, Email, Department;
  - Delete: para deletar na lista de funcionários um funcionário específico, pelo seu respectivo nome;

```

public class FuncionariosController : ApiController
{
    //static list
    private static List<Funcionario> fun = new List<Funcionario>();

    //get all
    public List<Funcionario> Get()
    {
        return fun;
    }

    //create
    public HttpResponseMessage Post ( string name, string email, string department)
    {
        //Any blank field = err
        if ((string.IsNullOrEmpty(name)) || (string.IsNullOrEmpty(email)) || (string.IsNullOrEmpty(department)))
        {
            return Request.CreateResponse(HttpStatusCode.BadRequest, "Check empty fields!");
        }
        else
        {
            fun.Add(new Funcionario( name, email, department));
            return Request.CreateResponse(HttpStatusCode.Created, "Employee Created.");
        }
    }

    //delete by name
    public HttpResponseMessage Delete(string name)
    {
        //blank field = err
        if (string.IsNullOrEmpty(name))
        {
            return Request.CreateResponse(HttpStatusCode.BadRequest, "Could not find a employee to delete");
        }
        else
        {
            fun.RemoveAt(fun.IndexOf(fun.First(x => x.Name.Equals(name))));
            return Request.CreateResponse(HttpStatusCode.OK, "Employee Deleted");
        }
    }
}

```

### Testes básicos da utilização da API:

Após a criação das respectivas classes, iniciasse o programa POSTMAN para testarmos a aplicação.

- **Create:**
  - Clique no método POST;
  - Insira a URL;
  - Clique em BODY;
  - Selecione x-www-form-urlencoded;
  - Clique em Params;
  - Insira os parâmetros (insira quantas vezes desejar);
  - Exibição da mensagem de retorno do objeto na lista criado;
  - **OBS:** No exemplo foram criados 3 objetos.

localhost:49283/api/F + ... No Environment ▼ 👁 ⚙

**POST** ▼ localhost:49283/api/Funcionarios?name=Fabiano Júnior&email=fabiano@gmail.com&department=Jr. Dev Params Send Save ▼

Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/> name	Fabiano Júnior			
<input checked="" type="checkbox"/> email	fabiano@gmail.com			
<input checked="" type="checkbox"/> department	Jr. Dev			
New key	Value	Description		

Authorization Headers (1) **Body** Pre-request Script Tests Cookies Code

☐ form-data ☒ x-www-form-urlencoded ☐ raw ☐ binary

Key	Value	Description	...	Bulk Edit
New key	Value	Description		

Body Cookies Headers (10) Test Results Status: 201 Created Time: 380 ms Size: 428 B

Pretty Raw Preview JSON ▼ 🔍

```
1 "Employee Created."
```

- **Get: Retornando todos os objetos da lista**
  - Clique no método GET
  - Insira a URL: localhost:49283/api/Funcionarios ;
  - Clique em SEND;
  - Será retornado a LISTA com os objetos criados.

localhost:49283/api/F + ... No Environment ▼ 👁 ⚙

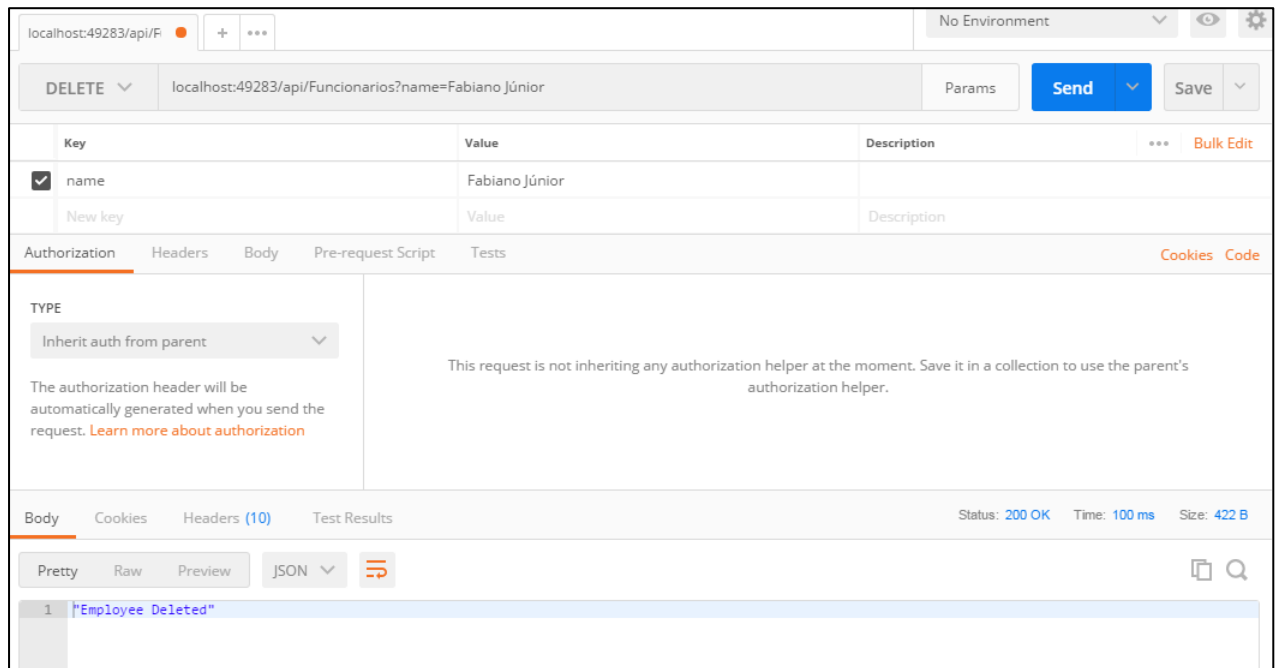
**GET** ▼ localhost:49283/api/Funcionarios Params Send Save ▼

Body Cookies Headers (10) Test Results Status: 200 OK Time: 183 ms Size: 610 B

Pretty Raw Preview JSON ▼ 🔍

```
1 [
2   {
3     "Name": "Fabiano Júnior",
4     "Mail": "fabiano@gmail.com",
5     "Dep": "Jr. Dev"
6   },
7   {
8     "Name": "Benedito Souza",
9     "Mail": "benedito@gmail.com",
10    "Dep": "Pl. Dev"
11  },
12  {
13    "Name": "João Silva",
14    "Mail": "joão@gmail.com",
15    "Dep": "Sn. Dev"
16  }
17 ]
```

- Delete: Removendo um usuário pelo nome
  - Selecione o método DELETE;
  - Clique em Params;
  - Passe o parâmetro 'name', com um nome da lista, no caso, Fabiano Júnior;
  - Clique em SEND;
  - Mensagem do objeto deletado exibida.



Execute novamente o método GET, com a URL: localhost:49283/api/Funcionarios, para verificar que o objeto já não existe mais

