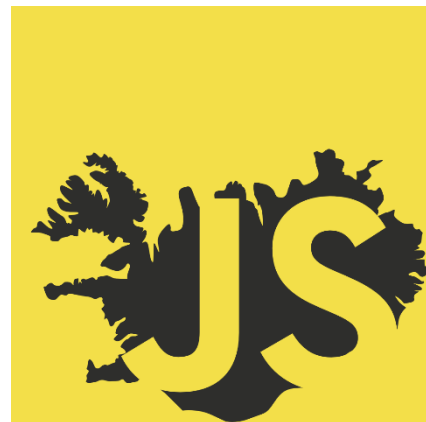


Creación de Apis con Mongo, Python, Html5 y Javascript (jquery)

Autor: Jesús Alfonso Contreras Rdz.



Contenido

| | |
|---|---|
| Definiciones | 2 |
| Ajax: | 2 |
| API (Application Programming Interface):..... | 2 |
| REST:..... | 2 |
| MongoDB:..... | 2 |
| Flask: | 3 |
| Backend: | 3 |
| Frontend: | 3 |
| Instalación del ambiente | 4 |
| Python: | 4 |
| MongoDB:..... | 5 |
| Programación | 6 |

Definiciones

Ajax:

Javascript y XML Asincrono, Técnica de desarrollo web que permite establecer comunicación asíncrona y en segundo plano con servicios externos al sitio web, permitiendo realizar cambios en la estructura y diseño de la página sin necesidad de recargar la misma, utilizando Javascript como código para manejo de eventos y Xml como formato de recepción y envío de datos.

API (Application Programming Interface):

Es un conjunto de funciones y procedimientos que cumplen una o muchas funciones con el fin de ser utilizadas por otro software.

REST:

Representational State Transfer, Estilo de arquitectura de software que utiliza el protocolo HTTP para establecer comunicación con componentes web. Permite el envío y recepción de contenido a través del protocolo mencionado anteriormente, sobre procedimientos bien definidos (GET, POST, PUT, DELETE) para establecimiento de comunicación Cliente/Servidor.

MongoDB:

Es una base de datos orientada a documentos. Esto quiere decir que en lugar de guardar los datos en registros, guarda los datos en documentos. Estos documentos son almacenados en BSON, que es una representación binaria de JSON.

Flask:

Es un “micro” Framework escrito en Python y concebido para facilitar el desarrollo de Aplicaciones Web bajo el patrón MVC.

Backend:

Está enfocado en hacer que todo lo que está detrás de un sitio web funcione correctamente. Toma los datos, los procesa y los envía al usuario, además de encargarse de las consultas o peticiones a la Base de Datos, la conexión con el servidor, entre otras tareas que debe realizar en su día a día. Cuenta con una serie de lenguajes y herramientas que le ayudan a cumplir con su trabajo como PHP, Ruby, Python, JavaScript, SQL, MongoDB, MySQL, etc, estos son usados para crear sitios dinámicos.

Frontend:

Se enfoca en el usuario, en todo con lo que podemos interactuar y lo que vemos mientras navegamos. Así como en una primera cita, nuestra web busca causar una buena impresión y agradar al usuario, para lo cual utiliza HTML, CSS y JAVASCRIPT.

Instalación del ambiente

Python:

1.- Instalar el programa de "python-2.7.15.amd64"

2.- Una vez instalado nos vamos a propiedades del sistema/ configuración avanzada del sistema/ Variables de entorno y en variables del sistema/PATH agregamos una nueva que es "C:\Python27"

3.- Abrimos un cmd y nos vamos a "cd/python27/scripts"

4.- Instalamos con el comando "pip install '1,2,3,4,5,6,7,8' "

1. Flask

2. Flask-WTF

3. Flask-RESTful

4. pymongo

5. flask_cors

6. requests

7. pyjwt

8. reportlab

5.- Ya después creamos nuestros archivos y desde un cmd nuevo nada más nos dirigimos desde el cmd a la carpeta en donde se encuentre nuestro archivo y ejecutamos el comando python "nombre_del_archivo.py"

MongoDB:

1. Creamos en el disco local C una carpeta llamada “data” y adentro de esta carpeta creamos otra que se llame “db”.
2. Después iniciamos el instalador de mongo.
3. Una vez instalado nos vamos a propiedades del sistema/ configuración avanzada del sistema/ Variables de entorno y en variables del sistema/PATH agregamos una nueva que es "
"C:\Program Files\MongoDB\Server\3.6\bin".
4. Una vez agregada la variable nos pasamos a iniciar el mongo.
5. "mongod" para poder iniciar mongo
6. "mongo" para poder ver nuestras bases de datos o insertar modificar, etc...

Programación

Para poder realizar un api primero tenemos que crear nuestros documentos con extensión .py



Directions.py



Functions.py

Directions.py hace el llamado al documento de Functions.py en pocas palabras aquí es donde nosotros crearemos nuestras apis.

directions = ligamiento a las funciones de las apis.

functions = son las funciones que nuestras apis realizaran.

Directions ejemplo:

```
#####  
####  
# Api para registrar un nuevo usuario #####  
#####  
####  
@app.route('/api/registerUser', methods=['POST'])  
def postRegisterUser():  
    try:  
        strName = request.json["Name"]  
        strEmail = request.json["Email"]  
        doubWeight = request.json["Weight"]  
        doubHeight = request.json["Height"]  
        doubTotal = request.json["Total"]  
        strPassword = request.json["strPassword"]  
  
        bolResult =  
callMethod.fnRegisterUser(strName,strEmail,doubWeight,doubHeight,doubTotal,strPassword)  
        return bolResult  
    except Exception as e:  
        #print e  
        respuesta = {'intResp':'0'}  
        return jsonify(respuesta)
```

Functions ejemplo:

```
#####
#####
#   Inserta un nuevo usuario Api tipo POST           #
#####
#####
def fnRegisterUser(strName,strEmail,doubWeight,doubHeight,doubTotal,strPassword):
    try:

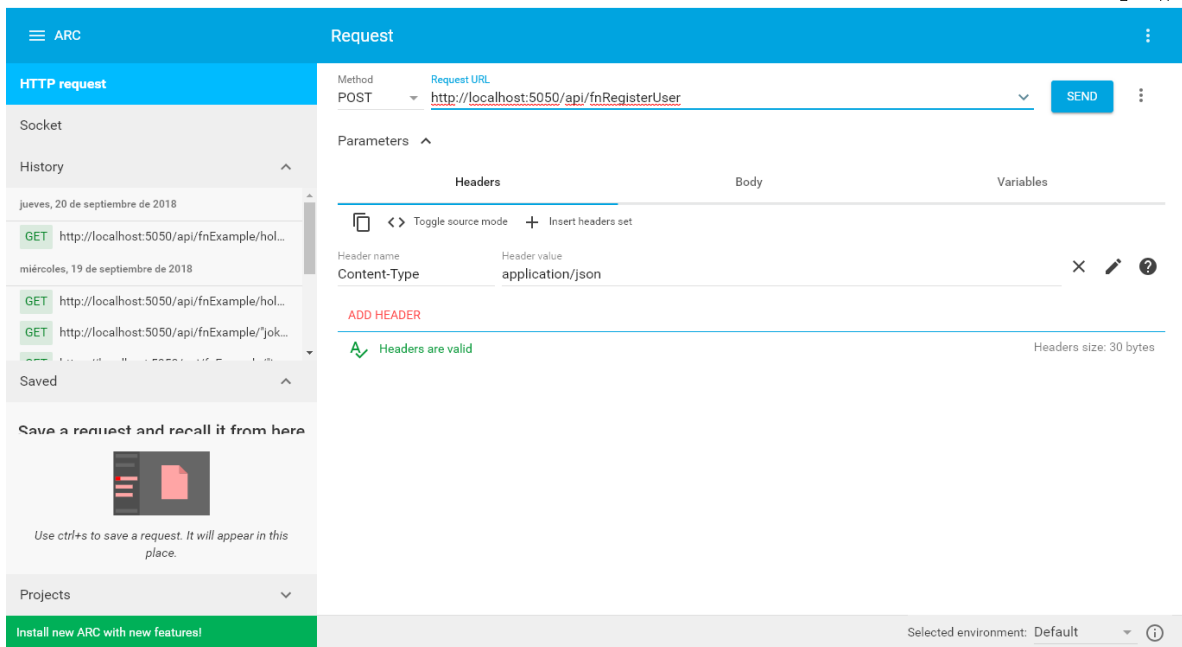
        db = connectDB()
        strEmail = strEmail.lower()
        strSearch = db.clUser.find({'strEmail':strEmail})
        if strSearch.count() != 0:
            i = {
                'intResp':'100',
                'strMessage':'This user was already registered'
            }
            return jsonify(i)

        db.clUser.insert({'strName':strName,'strEmail':strEmail,'strPassword':strPassword,
'doubHeight':float(doubHeight),
'arrIMC':[{'dteIMC':datetime.datetime.now(),'doubWeight':float(doubWeight),
'doubTotal':float(doubTotal)}}})
        i = {
            'intResp':'200',
            'strMessage':'Usuer '+strName+' registered correctly'
        }
        return jsonify(i)

    except excepcion as e:
        i = {
            'intResp':'500',
            'strMessage':'had an error'
        }

    return i
```

Aquí nosotros registraremos un nuevo usuario en la base de datos para ver si nuestra función en realidad funciona nos descargamos de google una extensión llamada “Advanced REST client”.



Si nuestra función fuera get nosotros en el method lo cambiamos a GET y los parámetros van en la de Request URL

Ejemplo get:

<http://localhost:5050/api/fnRegisterUser/parametro1/parametro2>

Antes de hacer lo de la prueba del api tenemos que habilitar nuestra base de datos y correr nuestro documento .py.

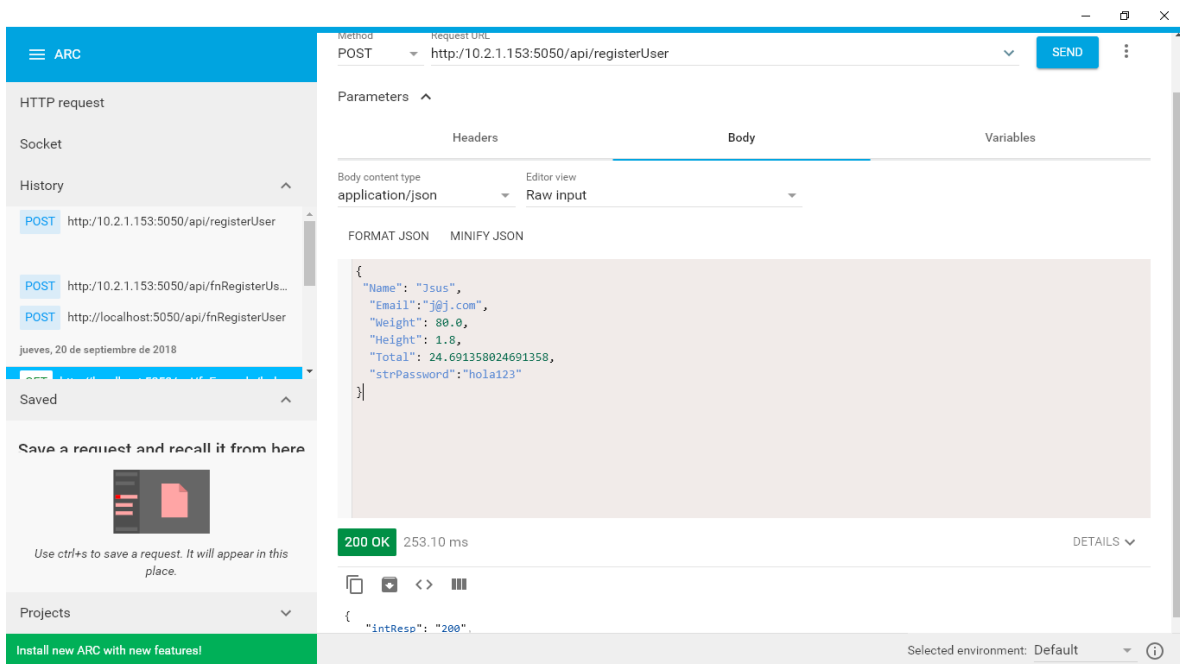
```
C:\WINDOWS\system32\cmd.exe - mongod
C:\Users\len1458>mongod
2018-10-11T17:26:46.801-0700 I CONTROL [initandlisten] MongoDB starting : pid=10124 port=27017 dbpath=C:\data\db\ 64-bit host=DESKTOP-V5F4S3B
2018-10-11T17:26:46.802-0700 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2018-10-11T17:26:46.805-0700 I CONTROL [initandlisten] db version v3.6.2
2018-10-11T17:26:46.805-0700 I CONTROL [initandlisten] git version: 489d177dbd0f0420a8ca04d39fd78d0a2c539420
2018-10-11T17:26:46.805-0700 I CONTROL [initandlisten] OpenSSL version: OpenSSL 1.0.1u-fips 22 Sep 2016
2018-10-11T17:26:46.805-0700 I CONTROL [initandlisten] allocator: tcmalloc
2018-10-11T17:26:46.806-0700 I CONTROL [initandlisten] modules: none
2018-10-11T17:26:46.806-0700 I CONTROL [initandlisten] build environment:
2018-10-11T17:26:46.806-0700 I CONTROL [initandlisten] distmod: 2008plus-ssl
2018-10-11T17:26:46.806-0700 I CONTROL [initandlisten] distarch: x86_64
2018-10-11T17:26:46.806-0700 I CONTROL [initandlisten] target_arch: x86_64
2018-10-11T17:26:46.806-0700 I CONTROL [initandlisten] options: {}
2018-10-11T17:26:46.809-0700 I - [initandlisten] Detected data files in C:\data\db\ created by the 'wiredTiger' storage engine, so setting the active storage eng
ine to 'wiredTiger'.
2018-10-11T17:26:46.810-0700 I STORAGE [initandlisten] wiredtiger_open config: create,cache_size=1385M,session_max=20000,eviction=(threads_min=4,threads_max=4),config_
base=false,statistics=(fast),log=(enabled=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),statistics_log=(wait=0),verbose=(recov
ery_progress),
2018-10-11T17:26:47.425-0700 I STORAGE [initandlisten] WiredTiger message [1539304007:424923][10124:140709919538256], txn-recover: Main recovery loop: starting at 29/1
3696
2018-10-11T17:26:47.617-0700 I STORAGE [initandlisten] WiredTiger message [1539304007:617590][10124:140709919538256], txn-recover: Recovering log 29 through 30
2018-10-11T17:26:47.761-0700 I STORAGE [initandlisten] WiredTiger message [1539304007:760990][10124:140709919538256], txn-recover: Recovering log 30 through 30
2018-10-11T17:26:49.044-0700 I CONTROL [initandlisten]
2018-10-11T17:26:49.044-0700 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-10-11T17:26:49.044-0700 I CONTROL [initandlisten] Read and write access to data and configuration is unrestricted.
2018-10-11T17:26:49.045-0700 I CONTROL [initandlisten]
2018-10-11T17:26:49.045-0700 I CONTROL [initandlisten] ** WARNING: This server is bound to localhost.
2018-10-11T17:26:49.046-0700 I CONTROL [initandlisten] Remote systems will be unable to connect to this server.
2018-10-11T17:26:49.046-0700 I CONTROL [initandlisten] Start the server with --bind_ip <address> to specify which IP
2018-10-11T17:26:49.047-0700 I CONTROL [initandlisten] addresses it should serve responses from, or with --bind_ip_all to
2018-10-11T17:26:49.047-0700 I CONTROL [initandlisten] bind to all interfaces. If this behavior is desired, start the
2018-10-11T17:26:49.047-0700 I CONTROL [initandlisten] server with --bind_ip 127.0.0.1 to disable this warning.
2018-10-11T17:26:49.048-0700 I CONTROL [initandlisten]
2018-10-11T17:26:49.048-0700 I CONTROL [initandlisten]
2018-10-11T17:26:49.049-0700 I CONTROL [initandlisten] ** WARNING: The file system cache of this machine is configured to be greater than 40% of the total memory. This
can lead to increased memory pressure and poor performance.
2018-10-11T17:26:49.049-0700 I CONTROL [initandlisten] See http://dochub.mongodb.org/core/ut-windows-system-file-cache
2018-10-11T17:26:49.050-0700 I CONTROL [initandlisten]
2018-10-11T19:26:52.936-0500 W FTDC [initandlisten] Failed to initialize Performance Counters for FTDC: WindowsPdhError: PdhExpandCounterPathW failed with 'El objet
o especificado no se encontró en el equipo.' for counter 'Memory\Available Bytes'
2018-10-11T19:26:52.936-0500 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory 'C:\data\db\diagnostic.data'
2018-10-11T19:26:52.961-0500 I NETWORK [initandlisten] waiting for connections on port 27017
```

```
C:\WINDOWS\system32\cmd.exe - python Directions.py
Microsoft Windows [Version 10.0.17134.345]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

C:\Users\lenT450>cd..
C:\Users>cd lenT450
C:\Users\lenT450>cd Desktop
C:\Users\lenT450\Desktop>cd "Proyecto apis"
C:\Users\lenT450\Desktop\Proyecto apis>cd Back
C:\Users\lenT450\Desktop\Proyecto apis\Back> python Directions.py
* Serving Flask app "Directions" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 327-057-127
* Running on http://0.0.0.0:5050/ (Press CTRL+C to quit)
```

Si nuestra función está correctamente nos tendrá que devolver algo como esto.

```
{
  "intResp": "200",
  "strMessage": "User Jsus registered correctly"
}
```



Para ver la función nos vamos al archivo .py que iniciamos que en este caso es directions desde el cmd.

```
C:\WINDOWS\system32\cmd.exe - python Directions.py
* Restarting with stat
* Debugger is active!
* Debugger PIN: 327-057-127
* Running on http://0.0.0.0:5050/ (Press CTRL+C to quit)
10.2.1.153 - - [11/Oct/2018 19:36:57] "POST /api/registerUser HTTP/1.1" 200 -
10.2.1.153 - - [11/Oct/2018 19:36:59] "POST /api/registerUser HTTP/1.1" 200 -
* Detected change in 'C:\\Users\\lenT450\\Desktop\\Proyecto apis\\Back\\Directions.py', reloading
* Restarting with stat
* Debugger is active!
* Debugger PIN: 327-057-127
* Running on http://0.0.0.0:5050/ (Press CTRL+C to quit)
* Detected change in 'C:\\Users\\lenT450\\Desktop\\Proyecto apis\\Back\\BackEnd\\Functions.py', reloading
* Restarting with stat
* Debugger is active!
* Debugger PIN: 327-057-127
* Running on http://0.0.0.0:5050/ (Press CTRL+C to quit)
* Detected change in 'C:\\Users\\lenT450\\Desktop\\Proyecto apis\\Back\\Directions.py', reloading
* Restarting with stat
* Debugger is active!
* Debugger PIN: 327-057-127
* Running on http://0.0.0.0:5050/ (Press CTRL+C to quit)
pppp
10.2.1.153 - - [11/Oct/2018 19:38:07] "POST /api/registerUser HTTP/1.1" 200 -
* Detected change in 'C:\\Users\\lenT450\\Desktop\\Proyecto apis\\Back\\Directions.py', reloading
* Restarting with stat
* Debugger is active!
* Debugger PIN: 327-057-127
* Running on http://0.0.0.0:5050/ (Press CTRL+C to quit)
10.2.1.153 - - [11/Oct/2018 19:38:50] "POST /api/registerUser HTTP/1.1" 200 -
```

Abrimos otro cmd y ahí ponemos “mongo” una ves hecho esto usaremos la base de datos en este caso es “runners” asi que colocamos el comando “use runners”

```
C:\WINDOWS\system32\cmd.exe - mongo
C:\Users\lenT450>mongo
MongoDB shell version v3.6.2
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.6.2
Server has startup warnings:
2018-10-11T17:28:44.731-0700 I CONTROL [initandlisten]
2018-10-11T17:28:44.731-0700 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-10-11T17:28:44.731-0700 I CONTROL [initandlisten] Read and write access to data and configuration is unrestricted.
2018-10-11T17:28:44.732-0700 I CONTROL [initandlisten]
2018-10-11T17:28:44.732-0700 I CONTROL [initandlisten] ** WARNING: This server is bound to localhost.
2018-10-11T17:28:44.733-0700 I CONTROL [initandlisten] Remote systems will be unable to connect to this server.
2018-10-11T17:28:44.733-0700 I CONTROL [initandlisten] ** Start the server with --bind_ip <address> to specify
which IP
2018-10-11T17:28:44.734-0700 I CONTROL [initandlisten] ** addresses it should serve responses from, or with --
bind_ip_all to
2018-10-11T17:28:44.734-0700 I CONTROL [initandlisten] ** bind to all interfaces. If this behavior is desired,
start the
2018-10-11T17:28:44.735-0700 I CONTROL [initandlisten] ** server with --bind_ip 127.0.0.1 to disable this warni
ing.
2018-10-11T17:28:44.735-0700 I CONTROL [initandlisten]
2018-10-11T17:28:44.735-0700 I CONTROL [initandlisten]
2018-10-11T17:28:44.736-0700 I CONTROL [initandlisten] ** WARNING: The file system cache of this machine is configured
to be greater than 40% of the total memory. This can lead to increased memory pressure and poor performance.
2018-10-11T17:28:44.736-0700 I CONTROL [initandlisten] See http://dochub.mongodb.org/core/ut-windows-system-file-cache
> use runners
switched to db runners
> db.clUser.find({}).pretty()
```

Y con el comando `db.clUser.find({}).pretty()` nos mostrara el usuario guardado en la base de datos.

```
C:\WINDOWS\system32\cmd.exe - mongo
```

```
{
  "dteIMC" : ISODate("2018-10-10T12:57:33.628Z"),
  "doubWeight" : 200,
  "doubTotal" : 61.72839506172839
},
{
  "dteIMC" : ISODate("2018-10-10T17:31:11.088Z"),
  "doubWeight" : 99,
  "doubTotal" : 30.555555555555554
}
],
"strToken" : "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyZXQlOjE1MzkxOTQwMzAuMmAwMCwiOiJ1bm1kZWY1NjhlMzIzMWY2OTYSMTk1ClleHAJOjE1MzkxOTA2MzAuMmAwMCwiOiJyZW9kaWw1OjIyYW9kaG9BbWFrPbc5jb281fQ.4Kjpxa8_BrQn9RmlV7t-_8b1PdoyH037XPcmGP0Sk"}
{
  "_id" : ObjectId("5bbfed1ae64b32d108ae5c5"),
  "strPassword" : "hola123",
  "doubHeight" : 1.8,
  "strName" : "Jesus",
  "strEmail" : "j@j.com",
  "arrIMC" : [
    {
      "dteIMC" : ISODate("2018-10-11T19:38:50.552Z"),
      "doubWeight" : 80,
      "doubTotal" : 24.691358024691358
    }
  ]
}
```

Ahora para que nuestra api funcione en el html tenemos que usar un javascript.

Javascript ejemplo:

```
var strIpAddress = "10.2.1.153" //Variable para hacer las conexiones a las
apis
                                //Aqui se coloca la direccion ipv4 para
poder usar nuestro sitio en otro dispositivo.
```

En la variable strIpAddress nosotros vamos a cambiarla por nuestra ip.

```

/*#####
# Inserta un usuario a la base de datos                                #
# Date:20/09/2018                                                    #
#####*/
function fnPutInsertNewUser(){
    var strName = $('#strName').val();
    var strEmail = $('#strEmail').val();
    var doubWeight = $('#doubWeight').val();
    var doubHeight = $('#doubHeight').val();
    var doubTotal = $('#doubTotal').val();
    var strPassword = $('#strPassword').val();
    var strConfirmPassword = $('#strConfirmPassword').val();
    if(strName != '' && strEmail != '' && doubWeight != '' && doubHeight
!= '' && doubTotal != ''
    && strPassword != '' && strConfirmPassword != '' && strPassword ==
strConfirmPassword){
        $.ajax({
            type: 'POST',

```

```

        url: 'http://' + strIpAddress + ':5050/api/registerUser',
        contentType: 'application/json; charset=utf-8',
        dataType: 'json',
        data: JSON.stringify({
            'Name': strName,
            'Email':strEmail,
            'Weight':doubWeight,
            'Height':doubHeight,
            'Total':doubTotal,
            'strPassword':strPassword
        }),
        success: function (result) {
            if(result.intResp == '200'){
                $('#mdlMessages').modal('show');
                $('#mdlMessagesTitle').text('Successfull');
                $('#mdlMessagesBody').text(result.strMessage);
                $('#mdlMessagesBtnAction').html('<button
type="button" onclick="window.location.replace(\'index.html\')" class="btn
btn-default" data-dismiss="modal">Close</button>');
            }else{
                $('#mdlMessages').modal('show');
                $('#mdlMessagesTitle').text('Error');
                $('#mdlMessagesBody').text(result.strMessage);
            }
        },
        error: function (result) {
            alert('had a error');
        }
    });
}
}

```

Html5 ejemplo:

```
<!--Inicio Form----->
<div class="container">
  <h2>Basic Form</h2>
  <form method="POST"> <!--action="insertar.php" method="POST"-->

    <div class="form-group">
      <label for="text">Name:</label>
      <input type="text" class="form-control" id="strName"
placeholder="Enter Name" name="Name" required>
    </div>

    <div class="form-group">
      <label for="email">Email:</label>
      <input type="email" class="form-control" id="strEmail"
placeholder="Enter email" name="Email" required>
    </div>

    <div class="form-group">
      <label for="password">Password:</label>
      <input type="password" class="form-control" id="strPassword"
placeholder="Password" name="pwd">
    </div>

    <div class="form-group">
      <label for="confirmPassword">Confirm Password:</label>
      <input type="password" class="form-control"
id="strConfirmPassword" placeholder="Confirm Password" min="0"
name="confirmPwd">
    </div>

    <div class="form-group">
      <label for="weight">Weight:</label>
      <input type="number" class="form-control" step="any"
id="doubWeight" onclick="fnImc()" onkeyup="fnImc()" placeholder="Weight"
min="0" name="Sal">
    </div>

    <div class="form-group">
      <label for="height">Height:</label>
      <input type="number" class="form-control" step="any"
id="doubHeight" onclick="fnImc()" onkeyup="fnImc()" placeholder="Height"
min="0" name="Incomes">
    </div>
```

```

        <div class="form-group">
            <label for="total">Total:</label>
            <input type="number" class="form-control" id="doubTotal"
placeholder="Total" name="Total" disabled>
        </div>

        <!-- <button type="submit" onclick="fnGetMessage()" class="btn btn-
success">Accept</button>-->
        <input type="button" onclick="fnPutInsertNewUser()" class="btn btn-
success" value="Accept">
        <button type="reset" class="btn btn-danger">Delete</button>

    </form>
</div>
<!--Fin Form----->

```

Mini Project JACR

file:///C:/Users/lenT450/Desktop/Projecto%20apis/Front/form.html

Basic Form

Name:

Email:

Password:

Confirm Password:

Weight:

Height:

Total: