



## ESCUELA SUPERIOR DE INGENIERÍA

Programación en Internet

Grado en Ingeniería Informática

Tutorial “API RESTful”

**Autores:**

Antonio José Rivas Macías

Fanny Chunyan Vela Díaz

Puerto Real, 28 de noviembre de 2018

# Índice General

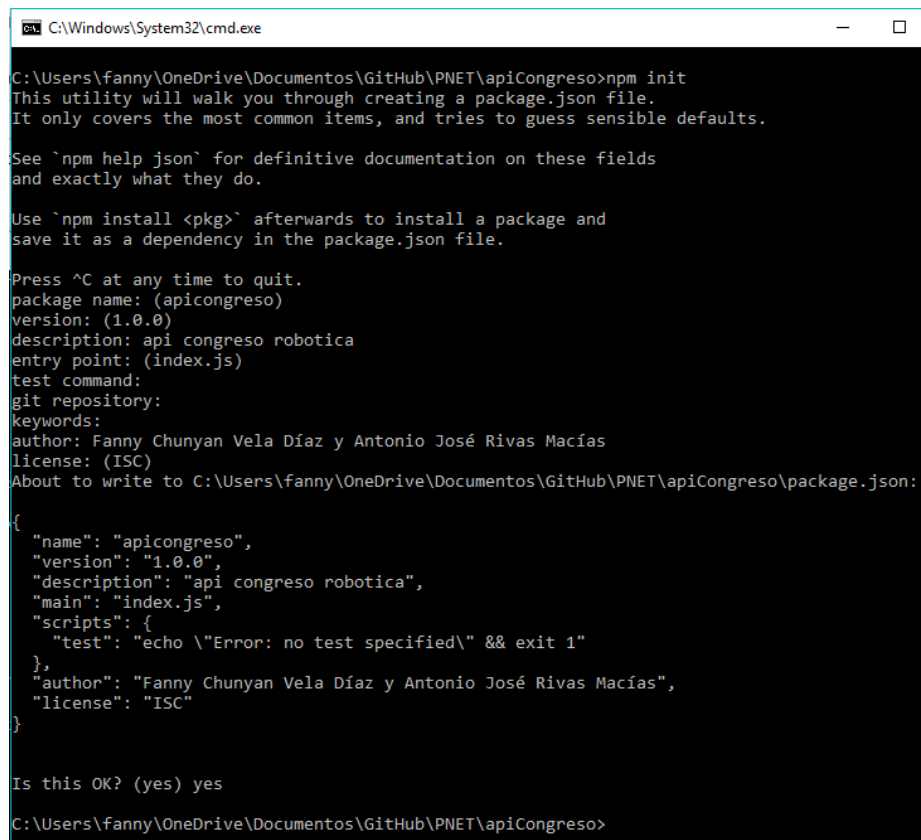
<b>1. Creación de la API RESTful.....</b>	<b>1</b>
1.1 Comprobación del funcionamiento de los métodos mediante POSTMAN.....	5
<b>2. Creación API RESTful (Cliente) .....</b>	<b>8</b>

## 1. Creación de la API RESTful

Para la creación de la API seguiremos los siguientes pasos:

1. Creamos en el directorio donde se aloja nuestra página web una carpeta. “apiCongreso” en nuestro caso.
2. Abrimos la consola en la carpeta y ejecutamos el siguiente comando:

```
npm init
```



```
C:\Windows\System32\cmd.exe
C:\Users\fanny\OneDrive\Documentos\GitHub\PNET\apiCongreso>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (apicongreso)
version: (1.0.0)
description: api congreso robotica
entry point: (index.js)
test command:
git repository:
keywords:
author: Fanny Chunyan Vela Díaz y Antonio José Rivas Macías
license: (ISC)
About to write to C:\Users\fanny\OneDrive\Documentos\GitHub\PNET\apiCongreso\package.json:
{
  "name": "apicongreso",
  "version": "1.0.0",
  "description": "api congreso robotica",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Fanny Chunyan Vela Díaz y Antonio José Rivas Macías",
  "license": "ISC"
}

Is this OK? (yes) yes
C:\Users\fanny\OneDrive\Documentos\GitHub\PNET\apiCongreso>
```

De esta forma se nos creará un archivo llamado “package.json”.

3. A continuación instalamos las dependencias que se añadiran al archivo “package.json” con el siguiente comando:

```
npm install express morgan body-parser --save
```

```
C:\Users\fanny\OneDrive\Documentos\GitHub\PNET\apiCongreso>npm install express morgan body-par
ser --save
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN apicongreso@1.0.0 No repository field.

+ morgan@1.9.1
+ body-parser@1.18.3
+ express@4.16.4
added 54 packages from 36 contributors and audited 160 packages in 10.973s
found 0 vulnerabilities
```

4. Creamos un fichero “index.js” en la raíz, “apiCongreso” en nuestro ca-  
so. Debe quedar de la siguiente forma (explicación en la captura de los  
diferentes elementos):

```
apiCongreso/index.js (PNET) - Brackets
rollo Ayuda

1 //Constantes
2 const express = require('express');
3 const app = express();
4 const logger = require('morgan');
5 const http = require('http');
6 const path = require('path');
7 const PORT = process.env.PORT || 8080;
8 const bodyParser = require('body-parser');
9 const baseAPI = '/api/v1';
10
11 //Configuraciones para la app
12 app.use(bodyParser.json());
13 app.use(logger('dev'));
14 app.use(bodyParser.urlencoded({
15   extended: true
16 }));
17
18 //Iniciación del servidor
19 const server = http.createServer(app);
20
21 //Escucha del servidor por el puerto especificado (8080 por defecto)
22 server.listen(PORT, function () {
23   console.log('Server up and running on localhost:' + PORT);
24 });
25
```

5. Creamos la carpeta “routes” en la raíz (apiCongreso)

6. En su interior creamos el fichero “attendees.js” . Este debe quedar tal que así:

```

1 'use strict';
2
3 //Constantes
4 const express = require('express');
5 const router = express.Router();
6
7 let attendees = [
8   {
9     "name": "Sonia",
10    "surname": "García",
11    "dni": "11234325L",
12    "email": "sonia@gmail.com",
13    "phone": "604566593",
14    "days": 2
15  },
16  {
17    "name": "Sergio",
18    "surname": "Díaz",
19    "dni": "21339835",
20    "email": "sergio@gmail.com",
21    "phone": "784332495",
22    "days": 3
23  }
24 ];
25
26 //***** OPERACIONES *****
27
28 /** GET */
29
30 /** GET */
31 //Recupera TODOS los asistentes
32 router.get('/', function (req, res) {
33   res.send(attendees);
34 });
35
36 //Recupera UN asistente
37 router.get('/:dni', function (req, res) {
38   let dni = req.params.dni;
39   res.send(attendees.filter(m => m.dni === dni));
40 });
41
42 /** POST */
43
44 //Para crear nuevo asistente
45 router.post('/', function (req, res) {
46   attendees.push(req.body);
47   res.status(201).send("Attendee created!");
48 });
49
50 /** PUT */
51 //Actualiza la info de UN asistente
52 router.put('/:dni', function (req, res) {
53   let dni = req.params.dni;
54   attendees = attendees.filter(m => m.dni !== dni);
55   attendees.push(req.body);
56   res.status(200).send("Attendee updated!");
57 });
58
59 /** DELETE */
60 //Elimina a TODOS los asistentes
61 router.delete('/', function (req, res) {
62   attendees = [];
63   res.status(200).send("Attendees deleted!");
64 });
65
66 //Elimina UN asistente
67 router.delete('/:dni', function (req, res) {
68   let dni = req.params.dni;
69   attendees = attendees.filter(m => m.dni !== dni);
70   res.status(200).send("Attendee deleted!");
71 });
72
73 module.exports = router; //Exportamos el router creado
74

```

7. Instalamos las dependencias CORS con el siguiente comando:

```
npm install cors --save
```

```

C:\Users\fanny\OneDrive\Documentos\GitHub\PNET\apiCongreso>npm install cors --save
npm WARN apicongreso@1.0.0 No repository field.

+ cors@2.8.5
added 2 packages from 2 contributors and audited 163 packages in 13.02s
found 0 vulnerabilities

```

8. En el fichero “index.js” incluimos la siguiente línea a continuación de las demás dependencias:

```
const cors = require('cors');
```

9. Justo después añadimos la siguiente línea

```
app.use(cors());
```

10. Ahora le daremos persistencia a nuestra API. Para ello usaremos una bd. En primer lugar creamos el fichero “attendees-service.js” en el directorio “routes”. Este debe quedar de la siguiente forma:

```

'use strict';

//Constantes
const MongoClient = require('mongodb').MongoClient;
let db;
let ObjectId = require('mongodb').ObjectId;
const Attendees = function () {
  //
};

//Configuración para la bd
Attendees.prototype.connectDB = function (callback) {
  MongoClient.connect('mongodb://admin:admin1234@93.21.141.10:27021/pnet-fannyela',
    {useNewUrlParser: true},
    function (err, database) {
      if (err) {
        callback(err);
      }
      db = database.db('pnet-fannyela').collection('attendees'); //Accede a la tabla attendees
      callback(err, database);
    });
};

//***** Métodos para trabajar con la base de datos *****

// ADD
Attendees.prototype.add = function (attendee, callback) {
  return db.insert(attendee, callback);
};

// GET ONE
Attendees.prototype.get = function (id, callback) {
  return db.findOne({_id: ObjectId(id)}, callback);
};

// GET ALL
Attendees.prototype.getAll = function (callback) {
  return db.find({}).toArray(callback);
};

// UPDATE
Attendees.prototype.update = function (id, updatedAttendee, callback) {
  return db.updateOne({_id: ObjectId(id)}, {$set: updatedAttendee}, callback);
};

// REMOVE ONE
Attendees.prototype.remove = function (id, callback) {
  return db.deleteOne({_id: ObjectId(id)}, callback);
};

```

```

53 //REMOVE ALL
54 Attendees.prototype.removeAll = function (callback) {
55     return db.deleteMany({}, callback);
56 };
57
58 module.exports = new Attendees();

```

11. Editamos el fichero “attendees.js” eliminando el array de attendees y modificando los métodos quedando de la siguiente forma:

```

attendees.js
1  'use strict';
2
3  //Constantes
4  const express = require('express');
5  const router = express.Router();
6
7  //Importamos las conexiones con la BD
8  const attendeesService = require('./attendees-service');
9
10 //***** OPERACIONES *****/
11
12 /** GET */
13
14 //Recupera TODOS los asistentes
15 router.get('/', function (req, res) {
16     attendeesService.getAll((err, attendees) => {
17         if (err) {
18             res.status(500).send({ msg: err });
19         } else if (attendees === null) {
20             res.status(500).send({ msg: "attendees null" });
21         } else if (attendees !== null) {
22             res.status(200).send(attendees);
23         }
24     });
25 });
26
27 //Recupera UN asistente
28 router.get('/:id', function (req, res) {
29     let _id = req.params.id;
30     attendeesService.get(_id, (err, attendee) => {
31         if (err) {
32             res.status(500).send({
33                 msg: err
34             });
35         } else if (attendee === null) {
36             res.status(500).send({
37                 msg: "attendee null"
38             });
39         } else if (attendee !== null) {
40             res.status(200).send(attendee);
41         }
42     });
43 });
44
45 /** POST */
46
47 //Para crear nuevo asistente
48 router.post('/', function (req, res) {
49     let attendee = req.body;
50     attendeesService.add(attendee, (err, attendee) => {
51         if (err) {
52             res.status(500).send({
53                 msg: err
54             });
55         } else if (attendee !== null) {
56             res.send({
57                 msg: 'Attendee created!'
58             });
59         }
60     });
61 });
62
63 /** PUT */
64
65 //Actualiza la info de un asistente
66 router.put('/:id', function (req, res) {
67     const _id = req.params.id;
68     const updatedAttendee = req.body;
69     attendeesService.update(_id, updatedAttendee, (err, numUpdates) => {
70         if (err || numUpdates === 0) {
71             res.status(500).send({
72                 msg: err
73             });
74         } else {
75             res.status(200).send({
76                 msg: 'Attendee updated!'
77             });
78         }
79     });
80 });
81
82 /** DELETE */
83
84 //Elimina a todos los asistentes
85 router.delete('/', function (req, res) {
86     attendeesService.removeAll((err) => {
87         if (err) {
88             res.status(500).send({
89                 msg: err
90             });
91         } else {
92             res.status(200).send({
93                 msg: 'Attendees deleted!'
94             });
95         }
96     });
97 });
98
99 //Elimina UN asistente
100 router.delete('/:id', function (req, res) {
101     let _id = req.params.id;
102     attendeesService.remove(_id, (err) => {
103         if (err) {
104             res.status(404).send({
105                 msg: err
106             });
107         } else {
108             res.status(200).send({
109                 msg: 'Attendee deleted!'
110             });
111         }
112     });
113 });
114
115 //Exporta el router
116 module.exports = router;

```

12. Instalamos las dependencias MongoDB en la raíz (apiCongreso)

```
npm install mongodb --save
```

```
C:\Users\fanny\OneDrive\Documentos\GitHub\PNET\apiCongreso>npm install mongodb --save
npm WARN apicongreso@1.0.0 No repository field.
+ mongodb@3.1.10
added 9 packages from 6 contributors and audited 174 packages in 3.691s
found 0 vulnerabilities
```

13. Editamos el fichero “index.js” añadiendo lo siguiente:

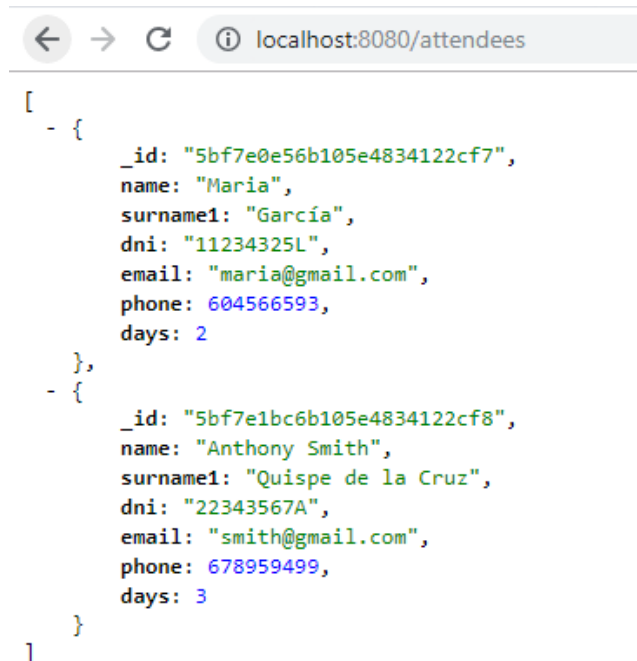
```
const attendeesService = require('./routes/
  attendees-service');
```

y actualizando el código de la inicialización del servidor por lo siguiente:

```
const attendeesService = require('./routes/
  attendees-service');
```

### 1.1. Comprobación del funcionamiento de los métodos mediante POSTMAN

#### ■ GET ALL



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/attendees'. The page content shows a JSON array of two attendee objects. The first object has an ID, name 'Maria', surname 'García', DNI '11234325L', email 'maria@gmail.com', phone '604566593', and 2 days. The second object has an ID, name 'Anthony Smith', surname 'Quispe de la Cruz', DNI '22343567A', email 'smith@gmail.com', phone '678959499', and 3 days.

```
[
  - {
    _id: "5bf7e0e56b105e4834122cf7",
    name: "Maria",
    surname1: "García",
    dni: "11234325L",
    email: "maria@gmail.com",
    phone: 604566593,
    days: 2
  },
  - {
    _id: "5bf7e1bc6b105e4834122cf8",
    name: "Anthony Smith",
    surname1: "Quispe de la Cruz",
    dni: "22343567A",
    email: "smith@gmail.com",
    phone: 678959499,
    days: 3
  }
]
```

## ■ GET ONE

localhost:8080/attendees/5bf7e1bc6b105e4834122cf8

```
[
  - {
    _id: "5bf7e1bc6b105e4834122cf8",
    name: "Anthony Smith",
    surname1: "Quispe de la Cruz",
    dni: "22343567A",
    email: "smith@gmail.com",
    phone: 678959499,
    days: 3
  }
]
```

## ■ POST

http://localhost:8080/attendees/

POST http://localhost:8080/attendees/ Send Save

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

none form-data x-www-form-urlencoded raw binary JSON (application/json) Beautify

```
1 {
2   "name": "Claudia",
3   "surname1": "Sánchez",
4   "dni": "1234567A",
5   "email": "claudia@gmail.com",
6   "phone": 657332122,
7   "days": 1
8 }
9
```

```
1 {
2   "msg": "Attendee created!"
3 }
```

localhost:8080/attendees/

```
[
  - {
    _id: "5bf7e0e56b105e4834122cf7",
    name: "Maria",
    surname1: "García",
    dni: "11234325L",
    email: "maria@gmail.com",
    phone: 604566593,
    days: 2
  },
  - {
    _id: "5bf7e1bc6b105e4834122cf8",
    name: "Anthony Smith",
    surname1: "Quispe de la Cruz",
    dni: "22343567A",
    email: "smith@gmail.com",
    phone: 678959499,
    days: 3
  },
  - {
    _id: "5bf7e2ca6b105e4834122cf9",
    name: "Claudia",
    surname1: "Sánchez",
    dni: "1234567A",
    email: "claudia@gmail.com",
    phone: 657332122,
    days: 1
  }
]
```



## ■ PUT

The screenshot shows a REST client interface with a PUT request to `http://localhost:8080/attendees/5bf7e2ca6b105e4834122cf9`. The request body is a JSON object representing an attendee's details. The response is a JSON object with a success message.

**Request:**

```
PUT http://localhost:8080/attendees/5bf7e2ca6b105e4834122cf9
```

**Body:**

```
{
  "name": "Claudia",
  "surname1": "Sánchez",
  "dni": "1234567A",
  "email": "claudia1234@gmail.com",
  "phone": 657332122,
  "days": 1
}
```

**Response:**

```
{
  "msg": "Attendee updated!"
}
```

The response body is also shown in a formatted JSON view below the response text:

```
[
  {
    _id: "5bf7e2ca6b105e4834122cf9",
    name: "Claudia",
    surname1: "Sánchez",
    dni: "1234567A",
    email: "claudia1234@gmail.com",
    phone: 657332122,
    days: 1
  }
]
```

## ■ DELETE ONE

The screenshot shows a REST client interface with a DELETE request to `http://localhost:8080/attendees/5bf7e2ca6b105e4834122cf9`. The request body is a JSON object representing an attendee's details. The response is a JSON object with a success message.

**Request:**

```
DELETE http://localhost:8080/attendees/5bf7e2ca6b105e4834122cf9
```

**Body:**

```
{
  "name": "Claudia",
  "surname1": "Sánchez",
  "dni": "1234567A",
  "email": "claudia1234@gmail.com",
  "phone": 657332122,
  "days": 1
}
```

**Response:**

```
{
  "msg": "Attendee deleted!"
}
```

The response body is also shown in a formatted JSON view below the response text:

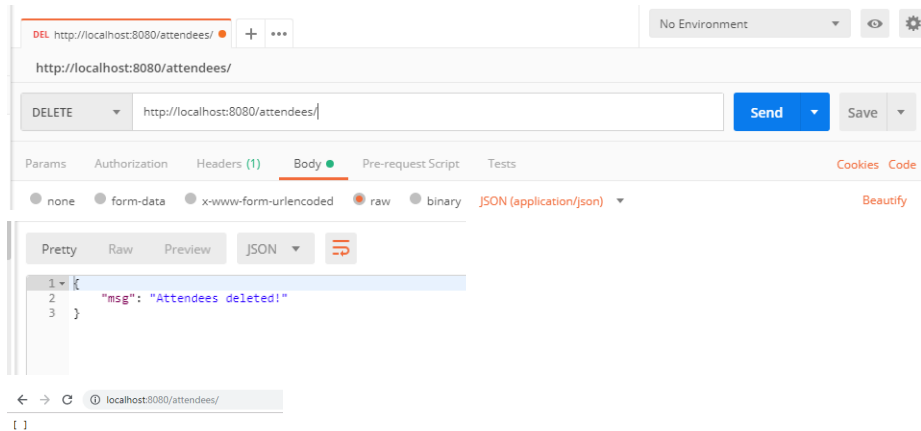
```
{
  "msg": "Attendee deleted!"
}
```

```

[
  {
    _id: "5bf7e0e56b105e4834122cf7",
    name: "Maria",
    surname1: "García",
    dni: "11234325L",
    email: "maria@gmail.com",
    phone: 604566593,
    days: 2
  },
  {
    _id: "5bf7e1bc6b105e4834122cf8",
    name: "Anthony Smith",
    surname1: "Quispe de la Cruz",
    dni: "22343567A",
    email: "smith@gmail.com",
    phone: 678959499,
    days: 3
  }
]

```

## ■ DELETE ALL



## 2. Creación API RESTful (Cliente)

Para la creación de la segunda parte del cliente de la API seguiremos los siguientes pasos utilizando AJAX y jQuery:

1. Creamos en el directorio donde se aloja nuestra página web una carpeta llamada “apiCongresoCliente” en nuestro caso.
2. Dentro de esta carpeta creamos otra llamada “js” e incluimos en el interior de esta un archivo .js llamado `wsinvocations.js`.
3. Incluimos también dentro de esta carpeta otro llamado “jquery-3.3.1.min.js” en el que añadamos el código existente en el siguiente enlace: <https://code.jquery.com/jquery-3.3.1.min.js>

4. Tras esto creamos un archivo HTML desde el que utilizamos nuestra API, normalmente se ubicara en la carpeta “apiCongresoCliente”, en nuestro caso lo llamaremos admin.HTML y lo ubicaremos a la misma altura que el resto de carpetas. (Cabe señalar que en nuestro caso utilizaremos la API tanto desde admin.HTML como desde inscripcion.HTML).
5. En este punto comenzaremos a implementar las funciones que realizan las acciones sobre los datos:

a) POST:

- 1) En la inscripcion.HTML incluimos los recuadros de texto y demas entradas de datos para introducir la inscripcion, junto con un boton que envíe el formulario (Para utilizarlo se introducen los datos y se pulsa el boton):

```
<form class="contact100-form" name = "formulario" onsubmit="return validar();">
  <div class="wrap-input100 " >
    <label for="name" class="label-input100">*Nombre:</label>
    <input class="input100" type="text" name="name" id="name" placeholder="Introduzca su nombre" required>
    <span class="focus-input100"></span>
  </div>
  <div class="wrap-input100 " >
    <label for="surname1" class="label-input100">*Apellido 1:</label>
    <input class="input100" type="text" name="surname1" id="surname1" placeholder="Introduzca su primer apellido" required>
    <span class="focus-input100"></span>
  </div>
  <div class="wrap-input100">
    <label for="surname2" class="label-input100">Apellido 2:</label>
    <input class="input100" type="text" name="surname2" id="surname2" placeholder="Introduzca su segundo apellido" >
    <span class="focus-input100"></span>
  </div>
  <div class="wrap-input100 " >
    <label for="dni" class="label-input100">*DNI/Nº Pasaporte:</label>
    <input class="input100" type="text" name="dni" id="dni" placeholder="Introduzca su dni" required>
    <span class="focus-input100"></span>
  </div>
  <div class="wrap-input100 " >
    <label for="email" class="label-input100">*Correo:</label>
    <input class="input100" type="text" name="email" id="email" placeholder="name@example.com" required>
    <span class="focus-input100"></span>
  </div>
  <div class="wrap-input100 ">
    <label for="phone" class="label-input100">Teléfono:</label>
    <input class="input100" type="text" name="phone" id="phone" placeholder="Introduzca un número de teléfono.">
    <span class="focus-input100"></span>
  </div>
  <div class="wrap-input100 " >
    <label for="dias" class="label-input100">*Nº de días:</label>
    <select name="dias" id="dias"><option value="1">1 Día</option> <option value="2">2 Días</option><option value="3">3 Días</option>
    </select>
    <span class="focus-input100"></span>
  </div>
  <div class="wrap-input100 ">
    <label for="diaentrada" class="label-input100">*Día entrada:</label>
    <input type="date" class="input100" name="diaentrada" id="diaentrada" min="2019-05-07" max="2019-05-09" required>
    <span class="focus-input100"></span>
  </div>
  <div class="wrap-input100 ">
    <label for="diasalida" class="label-input100">*Día salida:</label>
    <input type="date" class="input100" name="diasalida" id="diasalida" min="2019-05-07" max="2019-05-09" required>
    <span class="focus-input100"></span>
  </div>
  <div class="wrap-input100 ">
    <label for="presna" class="label-input100">¿Eres presna?:</label>
    <input type="checkbox" class="input100" name="presna" id="presna" >
    <span class="focus-input100"></span>
  </div>
  <div class="container-contact100-form-btn">
    <input class="contact100-form-btn" onclick="postInscripcion($('#name').val(),
    $('#surname1').val(), $('#surname2').val(), $('#dni').val(), $('#email').val(),
    $('#phone').val(), $('#dias').val(), $('#diaentrada').val(), $('#diasalida').val(), $('#presna').val())" type="button" value="ENVIAR"/>
  </div>
</form>
```

- 2) -En wsinvocations.js incluimos la funcion para introducir los datos en la BD:

```
function postInscripcion(nom, s1, s2, dni, email, tel, days, prensa) {  
  $.ajax({  
    type: "POST",  
    url: "http://localhost:8080/attendees/",  
    contentType: "application/json",  
    dataType: "text",  
    data: JSON.stringify({  
      "name": nom,  
      "surname1": s1,  
      "surname2": s2,  
      "dni": dni,  
      "email": email,  
      "phone": tel,  
      "days": days,  
      "prensa": prensa  
    }),  
    success: function(data) {  
      $("#result").html(data);  
    },  
    error: function(res) {  
      alert("ERROR " + res.statusText);  
    }  
  });  
}
```

- 3) Comprobación:

SIR 2019

Esta página dice  
Inscripción realizada con éxito

CONTACTO

Nombre: Susana

\*Apellido 1: Sánchez

Apellido 2: Fernández

\*DNI/Nº Pasaporte: 66443215

\*Correo: susana@gmail.com

Teléfono: 643322167

\*Nº de días: 1 Día

\*Dia entrada: 08/05/2019

\*Dia salida: 08/05/2019

¿Eres prensa?: ☐

ENVIAR

## Resultado

Nombre: Susana  
Apellido 1: Sánchez  
Apellido 2: Fernández  
DNI: 6644321S  
Email: susana@gmail.com  
Teléfono: 643322167  
Días de asistencia: 1  
Prensa: on

### b) GET (ALL):

- 1) En la admin.HTML incluimos un boton que devuelva todas las inscripciones almacenadas (Para utilizarlo basta con pulsa el boton):

```
<input class="boton" style="margin-left:100px" onclick="getInscripciones()" type="button" value="Mostrar asistentes" />
```

- 2) En wsinvocations.js incluimos la funcion para acceder a los datos en la BD:

```
function getInscripciones() {  
$.ajax({  
  type: "GET",  
  url: "http://localhost:8080/attendees/",  
  success: function(data) {  
    var attendees = data;  
    var HTMLText = "<ul>";  
  
    for( var i = 0 ; i < attendees.length ; i ++){  
      var attendeeAux = attendees[i];  
      HTMLText += "<li> Nombre: " + attendeeAux.name +  
        "<br>Apellido 1: " + attendeeAux.surname1 +  
        "<br>Apellido 2: " + attendeeAux.surname2 +  
        "<br>DNI: " + attendeeAux.dni +  
        "<br>Email: " + attendeeAux.email +  
        "<br>Teléfono: " + attendeeAux.phone +  
        "<br>Días de asistencia: " + attendeeAux.days +  
        "<br>Prensa: " + attendeeAux.prensa +  
        "</li><br><br>";  
    }  
    HTMLText += "</ul>";  
    $("#result").html(HTMLText);  
    /* $("#result").html(JSON.stringify(data)); */  
  },  
  error: function(res) {  
    alert("ERROR: "+ res.statusText);  
  }  
});  
}
```

### 3) Comprobación:

#### Resultado

Nombre: Carlota  
Apellido 1: Rodríguez  
Apellido 2: Benítez  
DNI: 77238893D  
Email: carlota@hotmail.com  
Teléfono: 678876599  
Días de asistencia: 2  
Prensa: on

Nombre: Maria  
Apellido 1: Rodríguez  
Apellido 2:  
DNI: 77238893A  
Email: maria@hotmail.com  
Teléfono: 678876591  
Días de asistencia: 1  
Prensa: on

Nombre: Mario  
Apellido 1: Vázquez  
Apellido 2:

---

#### c) GET (ONE):

- 1) En la admin.HTML incluimos el recuadro de texto para introducir el ID y un boton que devuelva la inscripcion de la persona con ese ID (Para utilizarlo se introduce el ID y se pulsa el boton):

```
<h6 style="margin-left:90px;">Identificador </h6>  
<input style="margin-left:90px;" class="boxtext" type="text" id="id"><br>  
<input class="boton" style="margin-left:100px;" onclick="getInscripcion($('#id').val())" type="button" value="Obtener Asistencia"/>
```

- 2) En wsinvocations.js incluimos la funcion para acceder a los datos en la BD:

```
function getInscripcion(id) {
$.ajax({
  type: "GET",
  url: "http://localhost:8080/attendees/" + id,
  success: function(data) {
    var attendee = data;
    var HTMLText = "<p>Nombre: " + attendee[0].name +
    "<br>Apellido 1: " + attendee[0].surname1 +
    "<br>Apellido 2: " + attendee[0].surname2 +
    "<br>DNI: " + attendee[0].dni +
    "<br>Email: " + attendee[0].email +
    "<br>Teléfono: " + attendee[0].phone +
    "<br>Días de asistencia: " + attendee[0].days +
    "<br>Prensa: " + attendee[0].prensa; + "</p>"
    $("#resul").html(HTMLText);
  },
  error: function(res) {
    alert("ERROR: "+ res.statusText); }
});
}
```

- 3) Comprobación

Escoja una opción

Mostrar asistentes

Eliminar asistentes

Identificador

Obtener asistente

Eliminar asistente

Resultado

Nombre: María  
Apellido 1: Rodríguez  
Apellido 2:  
DNI: 77238893A  
Email: maria@hotmail.com  
Teléfono: 678876591  
Días de asistencia: 1  
Prensa: on

- d) PUT:

- 1) En la admin.HTML incluimos los recuadros de texto y demas entradas de datos para introducir la inscripcion actualizada, junto con un boton que envíe el formulario (Para utilizarlo se introducen los datos actualizados y se pulsa el boton)

```

<h6 style="margin-left:90px;">Identificador </h6>
<input style="margin-left:90px;" class="boxtext" type="text" id="id"><br>
<h5 style="margin-left:90px;">Actualizar datos de un asistente</h5><br>
<h6 style="margin-left:90px;">Nota: Recuerde rellenar todos los campos</h6>
<div id="form">
  <form class="contact100-form" name = "formulario" onsubmit="return validar();" >
    <div class="wrap-input100 " >
      <label for="name" class="label-input100">* ID:</label>
      <input class="input100" type="text" name="name" id="ident" placeholder="Introduzca el identificador" required>
      <span class="focus-input100"></span>
    </div>
    <div class="wrap-input100 " >
      <label for="name" class="label-input100">*Nombre:</label>
      <input class="input100" type="text" name="name" id="name" placeholder="Modificar nombre" required>
      <span class="focus-input100"></span>
    </div>
    <div class="wrap-input100 " >
      <label for="surname1" class="label-input100">*Apellido 1:</label>
      <input class="input100" type="text" name="surname1" id="surname1" placeholder="Modificar primer apellido" required>
      <span class="focus-input100"></span>
    </div>
    <div class="wrap-input100">
      <label for="surname2" class="label-input100">Apellido 2:</label>
      <input class="input100" type="text" name="surname2" id="surname2" placeholder="Modificar segundo apellido" >
      <span class="focus-input100"></span>
    </div>
    <div class="wrap-input100 " >
      <label for="dni" class="label-input100">*DNI/N. Pasaporte:</label>
      <input class="input100" type="text" name="dni" id="dni" placeholder="Modificar dni" required>
      <span class="focus-input100"></span>
    </div>
    <div class="wrap-input100" >
      <label for="email" class="label-input100">*Correo:</label>
      <input class="input100" type="text" name="email" id="email" placeholder="Modificar correo: name@example.com" required>
      <span class="focus-input100"></span>
    </div>
    <div class="wrap-input100 ">
      <label for="phone" class="label-input100">*Teléfono:</label>
      <input class="input100" type="text" name="phone" id="phone" placeholder="Modificar número de teléfono" required>
      <span class="focus-input100"></span>
    </div>
    <div class="wrap-input100 " >
      <label for="dias" class="label-input100">*N. de días:</label>
      <select name="dias" id="dias">
        <option value="1">1 Día</option>
        <option value="2">2 Días</option>
        <option value="3">3 Días</option>
      </select>
      <span class="focus-input100"></span>
    </div>
    <div class="wrap-input100 ">
      <label for="prensa" class="label-input100">*Prensa:</label>
      <input type="checkbox" class="input100" name="prensa" id="prensa" >
      <span class="focus-input100"></span>
    </div>
    <input style="margin-left:50px;" class="button" onclick="putInscripcion($('#ident').val(), $('#name').val(), $('#surname1').val(), $('#surname2').val(), $('#dni').val(), $('#email').val(), $('#phone').val(), $('#dias').val(), $('#prensa').val())" type="button" value="Actualizar asistente"/>
  </form>
</div>

```



- 2) En wsinvocations.js incluimos la funcion para actualizar los datos en la BD

```
function putInscripcion(id, nom, surnamel, surname2, dni, email, phone, dias, prensa){
$.ajax({
  type: "PUT",
  url: "http://localhost:8080/attendees/" + id,
  data:{
    "name": nom,
    "surnamel": surnamel,
    "surname2": surname2,
    "dni": dni,
    "email": email,
    "phone": phone,
    "days": dias,
    "prensa": prensa
  },
  success: function(data){
    var HTMLText = "Asistente actualizado con éxito.";
    $("#result").html(HTMLText);
  },
  error: function(res){
    alert("ERROR: " + res.statusText);
  }
});
}
```

- 3) Comprobación:

* ID:	5bfe9059bdd9a91f782869ee
*Nombre:	María
*Apellido 1:	Rodríguez
Apellido 2:	Modificar segundo apellido
*DNI/Nº Pasaporte:	77238893A
*Correo:	mariarodriguez@gmail.com
*Teléfono:	678876591
*Nº de días:	1 Día ▾
*Prensa	<input checked="" type="checkbox"/>

Actualizar asistente

Resultado

Asistente actualizado con éxito.

Escoja una opción

Mostrar asistentes

Eliminar asistentes

Identificador

5bfe9059bdd9a91f782869ee

Obtener asistente

Eliminar asistente

Resultado

Nombre: María  
Apellido 1: Rodríguez  
Apellido 2:  
DNI: 77238893A  
Email: **maria.rodriguez@gmail.com**  
Teléfono: 678878591  
Días de asistencia: 1  
Prensa: on

e) DELETE (ALL):

- 1) En la admin.HTML incluimos un boton que elimine todas las inscripciones almacenadas (Para utilizarlo basta con pulsa el boton):

```
<input class="boton" style="margin-left:40px" onclick="deleteInscripciones()" type="button" value="Eliminar asistentes" />
```

- 2) -En wsinvocations.js incluimos la funcion para eliminar los datos de la BD

```
function deleteInscripcion(id) {
$.ajax({
  type: "DELETE",
  url: "http://localhost:8080/attendees/" + id,
  success: function(data) {
    var HTMLText = "Asistente eliminado con éxito";
    $("#resul").html(HTMLText);
  },
  error: function(res) {
    alert("ERROR: " + res.statusText);
  }
});
}
```

- 3) Comprobación:  
Al eliminar a todos, nos aparecerá un mensaje de éxito.

f) DELETE (ONE):

- 1) En la admin.HTML incluimos el recuadro de texto para introducir el ID y un boton que elimine la inscripcion de la persona con ese ID (Para utilizarlo se introduce el ID y se pulsa el boton)

```
<h6 style="margin-left:90px;">Identificador </h6>
<input style="margin-left:90px;" class="boxText" type="text" id="id"><br>
<input class="boton" style="margin-left:40px" onclick="deleteInscripcion($('#id').val())" type="button" value="Eliminar asistente"/><br>
```

- 2) -En wsinvocations.js incluimos la funcion para eliminar los datos de la BD

```
function deleteInscripciones(){
$.ajax({
  type: "DELETE",
  url: "http://localhost:8080/attendees/",
  success: function(data){
    var HTMLText = "Los asistentes fueron eliminados con éxito.";
    $("#result").html(HTMLText);
  },
  error: function(res){
    alert("ERROR: "+ res.statusText);
  }
});
}
```

- 3) Comprobación:

Escoja una opción	Resultado
<div>Mostrar asistentes</div> <div>Eliminar asistentes</div>	Asistente eliminado con éxito
Identificador	
<div>5bfe9059bdd9a91f782869ee</div>	
<div>Obtener asistente</div> <div>Eliminar asistente</div>	