

Formato de Informe de Seguimiento

Nombre del equipo:	Grupo 3 - Luis Alberto Moscoso Herrera - Luis Fernando Cortes - Luis Orlando Moreno Cruz - Luis Camilo Rosero Grijalba
Sprint No.:	1 (Uno)

1. Primera reunión (plan inicial del sprint) - lunes.

Pantallazo:



Observaciones:

Reunión de inicio para el sprint 1 y de retrospectiva en el cual se tomaron en cuenta las correcciones por parte del formador.

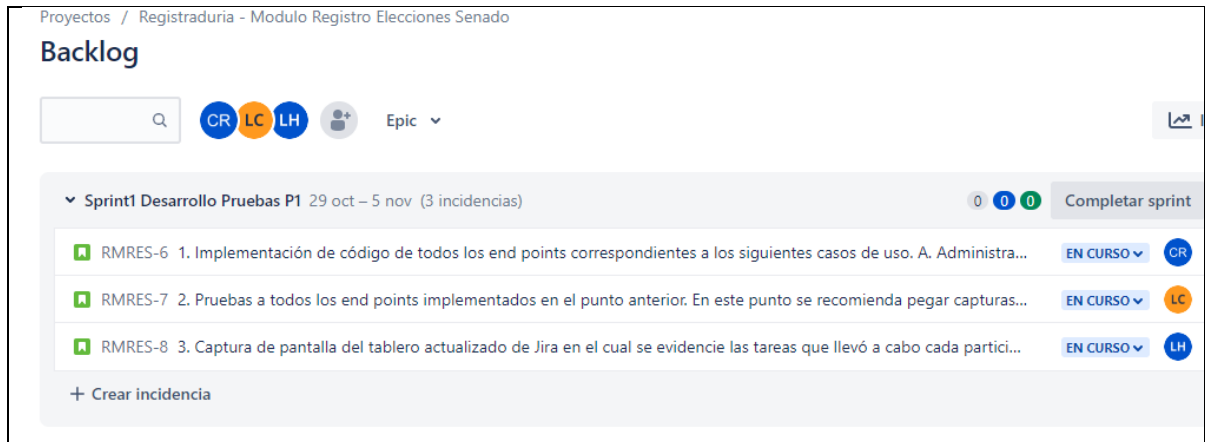
2. Reunión diaria de seguimiento - martes.

Pantallazo:

Observaciones:

3. Reunión diaria de seguimiento - miércoles.

Pantallazo:



Observaciones:

A través de Jira se organizo y distribuyo las tareas a realizar para el sprint 1

4. Reunión diaria de seguimiento - jueves.

Pantallazo:



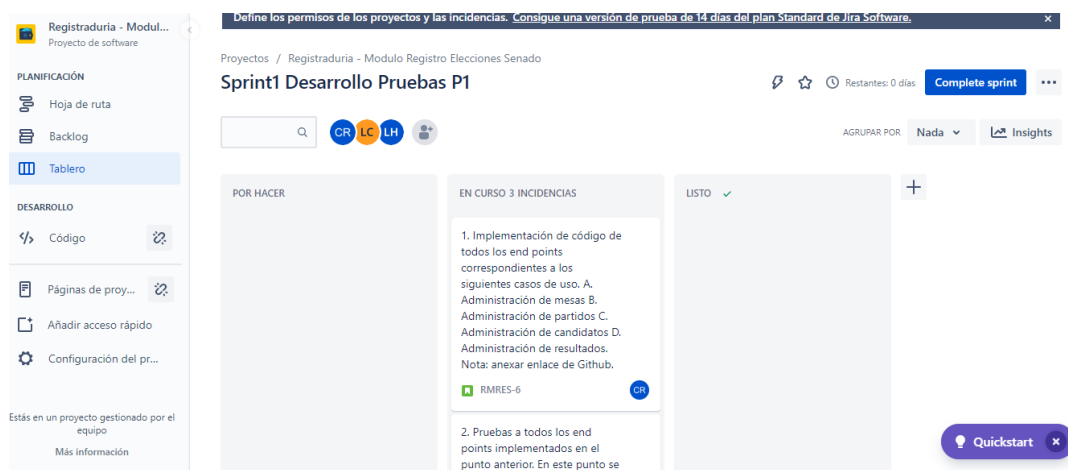
Observaciones:



Resumen de actividades

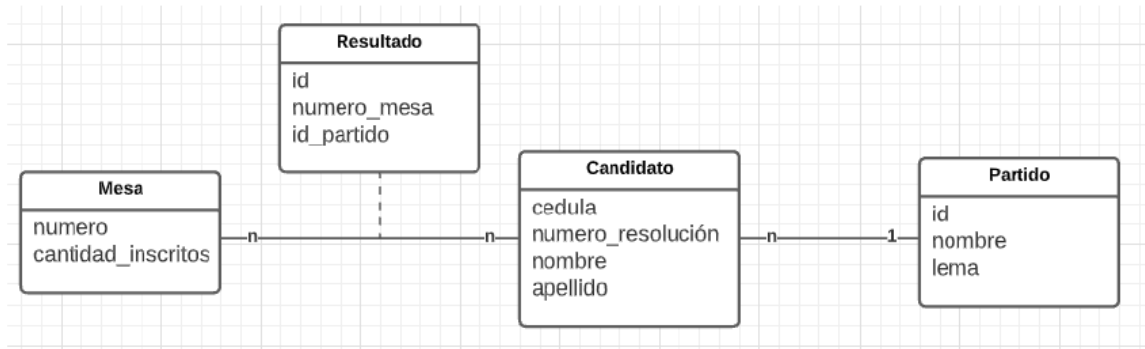
Como parte inicial de este sprint se organizó y distribuyo las diferentes actividades en la plataforma jira, la cual puede ser verificada en el siguiente enlace:

<https://kamirocode.atlassian.net/jira/software/projects/RMRES/boards/4>



Para el desarrollo de este sprint se comenzó a elaborar el código en backend con Python en el ide PyCharm en el cual se establecieron los diferentes modelos con sus respectivos controladores, todo esto teniendo en cuenta el diagrama de resultados planteado en clases.

Diagrama Base de Datos Backend de Resultados (Python - Flask)



También se crearon los diferentes repositorios mediante los cuales se puede realizar el CRUD, respectivamente se fue adicionando estas funciones a cada una de las entidades creadas en los modelos. Toda la codificación mencionada se la puede verificar en el siguiente enlace:

https://github.com/luismoscoso49/MINTIC_CICLO4

Además de la elaboración del código en Python se realizó la apertura de una cuenta en la página web de MongoDB, esto con el fin de acceder a una base de datos en forma remota que nos permita realizar las pruebas y posterior despliegue del software que está en desarrollo.

La imagen muestra la interfaz de MongoDB ClusterO. En la parte superior, se indica la versión 5.0.13 y la región AWS N. Virginia (us-east-1). El menú de navegación incluye Overview, Real Time, Metrics, Collections (seleccionado), Search, Profiler, Performance Advisor, Online Archive y Cmd Line Tool.

En la sección 'Collections', se muestra la base de datos 'bd-registraduria' con 4 colecciones. La tabla de colecciones es la siguiente:

Collection Name	Documents	Logical Data Size	Avg Document Size	Storage Size	Indexes	Index Size	Avg Index Size
candidato	0	0B	0B	24KB	1	32KB	32KB
mesa	0	0B	0B	12KB	1	12KB	12KB
partido	0	0B	0B	24KB	1	32KB	32KB
resultado	0	0B	0B	12KB	1	12KB	12KB

Después de vincular la base de datos en mongo con nuestro código en Python-Flask, se realizaron las pruebas de funcionamiento con los diferentes end-points propuestos en el código que se presentaba en las guías teniendo como resultado lo siguiente:

Pruebas con End-Points

Candidatos:

- Post

http://127.0.0.1:9999/candidatos

POST http://127.0.0.1:9999/candidatos

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "cedula": "1234567",
3   "nombre": "Luis ",
4   "apellido": "Moscoso",
5   "numero_resolucion": "76543210"
6 }
7
8
```

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 1591 ms Size: 282 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "_id": "636321dfff6104c2e89cb5fc6",
3   "apellido": "Moscoso",
4   "cedula": "1234567",
5   "nombre": "Luis ",
6   "numero_resolucion": "76543210"
7 }
```

- Get

http://127.0.0.1:9999/candidatos

GET http://127.0.0.1:9999/candidatos

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "cedula": "1234567",
3   "nombre": "Luis ",
4   "apellido": "Moscoso",
5   "numero_resolucion": "76543210"
6 }
7
8
```

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 111 ms Size: 698 B Save Response

Pretty Raw Preview Visualize JSON

```
17 {
18   "_id": "635f3e9f04e281950682704e",
19   "apellido": "Moreno Vargas",
20   "cedula": "1110601688",
21   "nombre": "Mikel",
22   "numero_resolucion": "1030288564"
23 },
24 {
25   "_id": "636321dfff6104c2e89cb5fc6",
26   "apellido": "Moscoso",
27   "cedula": "1234567",
28   "nombre": "Luis ",
29   "numero_resolucion": "76543210"
30 }
31 }
```

⌘ Cookies ⌘ Capture requests ⌘ Bootcamp ⌘ Runner ⌘ Trash ⌘ ?

- Put

http://127.0.0.1:9999/candidatos/636321dff6104c2e89cb5fc6 Save 🔗

PUT ⌵ http://127.0.0.1:9999/candidatos/636321dff6104c2e89cb5fc6 Send ⌵

Params Authorization Headers (8) **Body** ● Pre-request Script Tests Settings Cookies

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL **JSON** ⌵ Beautify

```
1 {
2   "cedula": "1234567",
3   "nombre": "Luis Alberto",
4   "apellido": "Moscoso",
5   "numero_resolucion": "76543210"
6 }
7
8
```

Body Cookies Headers (5) Test Results 🌐 Status: 200 OK Time: 387 ms Size: 289 B Save Response ⌵

Pretty Raw Preview Visualize **JSON** ⌵ 🔍

```
1 {
2   "_id": "636321dff6104c2e89cb5fc6",
3   "apellido": "Moscoso",
4   "cedula": "1234567",
5   "nombre": "Luis Alberto",
6   "numero_resolucion": "76543210"
7 }
```

- Delete

http://127.0.0.1:9999/candidatos/636321dff6104c2e89cb5fc6 Save 🔗

DELETE ⌵ http://127.0.0.1:9999/candidatos/636321dff6104c2e89cb5fc6 Send ⌵

Params Authorization Headers (8) **Body** ● Pre-request Script Tests Settings Cookies

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL **JSON** ⌵ Beautify

```
1 {
2   "cedula": "1234567",
3   "nombre": "Luis Alberto",
4   "apellido": "Moscoso",
5   "numero_resolucion": "76543210"
6 }
7
8
```

Body Cookies Headers (5) Test Results 🌐 Status: 200 OK Time: 126 ms Size: 178 B Save Response ⌵

Pretty Raw Preview Visualize **JSON** ⌵ 🔍

```
1 {
2   "deleted_count": 1
3 }
```

Mesas:

- Post

http://127.0.0.1:9999/mesas

POST http://127.0.0.1:9999/mesas

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 1
2 2
3 3
4 4
5 5
6 6
```

Status: 200 OK Time: 1250 ms Size: 237 B Save Response

Pretty Raw Preview Visualize JSON

```
1 1
2 2
3 3
4 4
5 5
```

- Get

http://127.0.0.1:9999/mesas

GET http://127.0.0.1:9999/mesas

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 1
2 2
3 3
4 4
5 5
6 6
```

Status: 200 OK Time: 97 ms Size: 395 B Save Response

Pretty Raw Preview Visualize JSON

```
3 1
4 2
5 3
6 4
7 5
8 6
9 7
10 8
11 9
12 10
13 11
14 12
15 13
```

- Put

http://127.0.0.1:9999/mesas/6363236af6104c2e89cb5fc7

PUT http://127.0.0.1:9999/mesas/6363236af6104c2e89cb5fc7 Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   "numero": "004",
3   "cantidad_inscritos": "35750"
4 }
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 736 ms Size: 237 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "_id": "6363236af6104c2e89cb5fc7",
3   "cantidad_inscritos": "35750",
4   "numero": "004"
5 }
```

- Delete

http://127.0.0.1:9999/mesas/6363236af6104c2e89cb5fc7

DELETE http://127.0.0.1:9999/mesas/6363236af6104c2e89cb5fc7 Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   "numero": "004",
3   "cantidad_inscritos": "35750"
4 }
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 102 ms Size: 178 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "deleted_count": 1
3 }
```

Partidos:

- Post

http://127.0.0.1:9999/partidos

POST

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Beautify

```
1 {
2   "nombre": "Centro Democrático",
3   "lema": "Mano Firme"
4 }
5
6
```

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 1501 ms Size: 243 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "_id": "636324e3f6104c2e89cb5fc8",
3   "lema": "Mano Firme",
4   "nombre": "Centro Democrático"
5 }
```

- Get

http://127.0.0.1:9999/partidos

GET

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Beautify

```
1 {
2   "nombre": "Centro Democrático",
3   "lema": "Mano Firme"
4 }
5
6
```

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 98 ms Size: 448 B Save Response

Pretty Raw Preview Visualize JSON

```
3   "_id": "635faa1c7338e1a442934dde",
4   "lema": "Oportunidades Para Todos",
5   "nombre": "Partido Liberal Colombiano"
6 },
7 {
8   "_id": "635faa977338e1a442934ddf",
9   "lema": "¡La Fuerza Que Decide!",
10  "nombre": "Conservador"
11 },
12 {
13   "_id": "636324e3f6104c2e89cb5fc8",
14   "lema": "Mano Firme",
15   "nombre": "Centro Democrático"
16 }
```


- Put

http://127.0.0.1:9999/partidos/636324e3f6104c2e89cb5fc8

PUT http://127.0.0.1:9999/partidos/636324e3f6104c2e89cb5fc8 Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   ---"nombre":"Centro Democratico",
3   ---"lema": "Mano Firme Corazon Grande"
4 }
5
6
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 411 ms Size: 259 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "_id": "636324e3f6104c2e89cb5fc8",
3   "lema": "Mano Firme Corazon Grande",
4   "nombre": "Centro Democratico"
5 }
```

- Delete

http://127.0.0.1:9999/partidos/636324e3f6104c2e89cb5fc8

DELETE http://127.0.0.1:9999/partidos/636324e3f6104c2e89cb5fc8 Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   ---"nombre":"Centro Democratico",
3   ---"lema": "Mano Firme Corazon Grande"
4 }
5
6
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 107 ms Size: 178 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "deleted_count": 1
3 }
```

Resultados

- Post

http://127.0.0.1:9999/resultados Save Send

POST http://127.0.0.1:9999/resultados Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies Beautify

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL ☐ JSON

```
1 [JSON]
2 {
3   "numero_mesa": "001",
4   "id_partido": "635faa1c7338e1a442934dde"
5 }
6
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 1097 ms Size: 253 B Save Response

Pretty Raw Preview Visualize **JSON** Copy Search

```
1 {
2   "_id": "6363274ef6104c2e89cb5fc9",
3   "id_partido": "635faa1c7338e1a442934dde",
4   "numero_mesa": "001"
5 }
```

- Get

http://127.0.0.1:9999/resultados Save Send

GET http://127.0.0.1:9999/resultados Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies Beautify

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL ☐ JSON

```
1 [JSON]
2 {
3   "numero_mesa": "001",
4   "id_partido": "635faa1c7338e1a442934dde"
5 }
6
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 102 ms Size: 446 B Save Response

Pretty Raw Preview Visualize **JSON** Copy Search

```
1 {
2   "_id": "635fb5e2d15029c6fc68e1a3",
3   "id_partido": "635faa1c7338e1a442934dde",
4   "numero_mesa": "001"
5 },
6 {
7   "_id": "635fb626d15029c6fc68e1a4",
8   "id_partido": "635faa977338e1a442934ddf",
9   "numero_mesa": "002"
10 },
11 {
12   "_id": "6363274ef6104c2e89cb5fc9",
13   "id_partido": "635faa1c7338e1a442934dde",
14   "numero_mesa": "001"
15 }
```

- Put

The screenshot shows a REST client interface with a PUT request to `http://127.0.0.1:9999/resultados/635fb5e2d15029c6fc68e1a3`. The request body is a JSON object: `{ "numero_mesa": "0002", "id_partido": "635faa977338e1a442934dd1" }`. The response status is 200 OK, and the response body is a JSON object: `{ "_id": "635fb5e2d15029c6fc68e1a3", "id_partido": "635faa977338e1a442934dd1", "numero_mesa": "0002" }`.

```
PUT http://127.0.0.1:9999/resultados/635fb5e2d15029c6fc68e1a3

{
  "numero_mesa": "0002",
  "id_partido": "635faa977338e1a442934dd1"
}
```

Status: 200 OK Time: 424 ms Size: 254 B

```
{
  "_id": "635fb5e2d15029c6fc68e1a3",
  "id_partido": "635faa977338e1a442934dd1",
  "numero_mesa": "0002"
}
```

- Delete

The screenshot shows a REST client interface with a DELETE request to `http://127.0.0.1:9999/resultados/635fb5e2d15029c6fc68e1a3`. The response status is 200 OK, and the response body is a JSON object: `{ "deleted_count": 1 }`.

```
DELETE http://127.0.0.1:9999/resultados/635fb5e2d15029c6fc68e1a3
```

Status: 200 OK Time: 139 ms Size: 178 B

```
{
  "deleted_count": 1
}
```

Las pruebas realizadas fueron exitosas, se verificó también en la base de datos a través de la pagina principal de mongo los datos ingresados, mostrados, actualizados y finalmente borrados, como se desarrollo con los diferentes métodos implementados en el código backend.

Informe de Retrospectiva

Nombre del equipo:	Grupo 3
Sprint No.:	1(Uno)

1. Resumen de los aspectos positivos y negativos del sprint:

- Como aspecto positivo: Se conoció un nuevo sistema con el cual se puede crear una base de datos como lo es MongoDB, además de la forma de conexión entre el servidor, el código implementado y la base de datos alojada en MongoDB
- Como aspecto negativo: El tiempo no fue el suficiente para poder abordar la temática vista y profundizarla más a fondo, de igual manera no se pudo contar con el tiempo suficiente para cumplir con el cronograma propuesto.

2. Propuesta (o propuestas) de cambio seleccionada(s) para el siguiente sprint:

Como tal sobre los Sprints no hay ninguna propuesta sin embargo teniendo en cuenta el tiempo disponible para el desarrollo de la temática propuesta, pensamos que es importante hacer una explicación un poco más detallada del código usado en los ejemplos de programación, así como también la temática dada

Teniendo en cuenta los aportes del formador en la entrega del sprint 0, se realizaron las correcciones sobre los diagramas para los microservicios y la estructuración de los mockups:

SEGURIDAD

1. CONSULTA DE INFORMACION DEL USUARIO DEL SISTEMA

Método	GET	
URL	https://[URL_SERVER]/votaciones/getUsuario/id=?	
Parámetros		
Campo	Tipo	Descripción
email	string	Cuenta de correo del usuario
Password	String	clave

Respuesta		
Campo	Tipo	Descripción
Token	string	De Seguridad + id del usuario
Ejemplo respuesta		
<pre>HTTP/1.1 200 OK { eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2t1b190eXB1Ijo1YW NjZXNzIiwiaXhwIjoxNjY2NzgzNDI5LCJqdGkiOiI5YjQ2MzAyNzY5ZTM0M jIyYmU0OTFhOGY3Mjc2OTE3ZCIsInVzZXJfaWQiOiJEWmDF9.1AKm55aPMS9 IvBk0BveK3vJdKLo_sGBLfcF849icnPE }</pre>		

2. PERMISO DE INGRESO A VISTA

Método	GET	
URL	https://[URL_SERVER]/votaciones/Consultar/id=?/obj=?	
Parámetros		
Campo	Tipo	Descripción
Id_usuario	Number	Identificador del usuario
Respuesta		
Campo	Tipo	Descripción
http response	string	Ok
Ejemplo respuesta		
<pre>HTTP/1.1 200 OK { Ok</pre>		

3. CREACION DE USUARIO

Método	POST	
URL	https://[URL_SERVER]/votaciones/newUsuario	
Parámetros		
Campo	Tipo	Descripción
Header		N/A
Body	JSON	{ cedula : number Nombre: string Edad: number Correo: string Contraseña: string Rol: number }
Respuesta		
Campo	Tipo	Descripción
http response	string	Ok
Ejemplo respuesta		
<pre>HTTP/1.1 200 OK { Ok }</pre>		
<pre>HTTP/1.1 400 { USUARIO NO PUDO SER CREADO }</pre>		

4. MODIFICACION USUARIO

Método	POST	
URL	https://[URL_SERVER]/votaciones/updUsuario/id=?	
Parámetros		
Campo	Tipo	Descripción
Id_usuario	Number	Identificador usuario
Body	JSON	{ cedula: number Nombre: string Edad: number Rol: number }
Respuesta		
Campo	Tipo	Descripción
http response	string	Ok
Ejemplo respuesta		
<pre>HTTP/1.1 200 OK { Ok }</pre>		
<pre>HTTP/1.1 400 { USUARIO NO PUDO SER ACTUALIZADO }</pre>		

5. ELIMINACION USUARIO

Método	DELETE	
URL	https://[URL_SERVER]/votaciones/delUsuario/id=?	
Parámetros		
Campo	Tipo	Descripción
Id_usuario	Number	Identificador usuario
Body		
Respuesta		
Campo	Tipo	Descripción
http response	string	Ok
Ejemplo respuesta		
<pre>HTTP/1.1 200 OK { Ok }</pre>		
<pre>HTTP/1.1 400 { CREDENCIALES NO VALIDAS }</pre>		

6. LISTA DE USUARIO DEL SISTEMA

Método	GET	
URL	https://[URL_SERVER]/votaciones/getUsuarios	
Parámetros		
Campo	Tipo	Descripción
Body		
Respuesta		
Campo	Tipo	Descripción
http	JSON	{ cedula : number Nombre: string Edad: number Correo: string Rol: number }
Ejemplo respuesta		
HTTP/1.1 200 OK		
{ cedula : 123 Nombre: PEDRO PEREZ Edad: 18 Correo: pedro@perez.com Rol: CIUDADANO }		
HTTP/1.1 500		
{ Solicitud no pudo ser atendida. Intente mas tarde }		

7. DATOS DEL PARTIDO

Método	GET	
URL	https://[URL_SERVER]/votaciones/getPartido/id=?	
Parámetros		
Campo	Tipo	Descripción
Idpartido	Number	Identificador de Partido
Body		
Respuesta		
Campo	Tipo	Descripción
http	JSON	{ IdPartido : number Nombre: string Lema: string }
Ejemplo respuesta		
HTTP/1.1 200 OK		
{ idpartido : 1 nombre: "CUALQUIERA" lema : "LOS MISMOS CON LAS MISMAS" }		
HTTP/1.1 500		
{ DATOS NO VALIDOS }		

8. DATOS DE LA MESA

Método	GET	
URL	https://[URL_SERVER]/votaciones/getMesa/id=?	
Parámetros		
Campo	Tipo	Descripción
Id_mesa	Number	No. De Mesa de Votación
Body		
Respuesta		
Campo	Tipo	Descripción
http	JSON	{ Id mesa : number inscritos: number }
Ejemplo respuesta		
HTTP/1.1 200 OK		
{ Id_mesa : 12 inscritos: 100 }		
HTTP/1.1 500		
{ DATOS NO VALIDOS }		

9. ADICION DE CANDIDATOS

Método	POST	
URL	https://[URL_SERVER]/votaciones/newCandidato	
Parámetros		
Campo	Tipo	Descripción
Header		N/A
Body	JSON	{ cedula : number Nombre: string resolucion: string curul : string }
Respuesta		
Campo	Tipo	Descripción
http response	string	Ok
Ejemplo respuesta		
<pre>HTTP/1.1 200 OK { Cedula : 123 Nombre : PEDRO PEREZ Resolución : 32131532321 Curul : SENADO }</pre>		
<pre>HTTP/1.1 400 { CANDIDATO NO PUDO SER CREADO }</pre>		

10. MODIFICACION CANDIDATOS

Método	PUT	
URL	https://[URL_SERVER]/votaciones/updCandidato	
Parámetros		
Campo	Tipo	Descripción
Id	Number	ID candidato
Body	JSON	{ Nombre: string resolucion: string curul : string }
Respuesta		
Campo	Tipo	Descripción
http response	string	Ok
Ejemplo respuesta		
<pre>HTTP/1.1 200 OK { Ok }</pre>		
<pre>HTTP/1.1 400 { Id CANDIDATO NO ENCONTRADO }</pre>		

11. ELIMINACION CANDIDATO

Método	DELETE	
URL	https://[URL_SERVER]/votaciones/delCandidato	
Parámetros		
Campo	Tipo	Descripción
Id	Number	Id candidato
Body		
Respuesta		
Campo	Tipo	Descripción
http response	string	Ok
Ejemplo respuesta		
<pre>HTTP/1.1 200 OK { Ok }</pre>		
<pre>HTTP/1.1 400 { Id CANDIDATO NO ENCONTRADO }</pre>		

12. LISTAR CANDIDATOS

Método	GET	
URL	https://[URL_SERVER]/votaciones/getCandidatos	
Parámetros		
Campo	Tipo	Descripción
Header		N/A
Body		
Respuesta		
Campo	Tipo	Descripción
http response	string	Ok
Ejemplo respuesta		
<pre>HTTP/1.1 200 OK { Cedula : 123 Nombre : PEDRO PEREZ Resolución : 32131532321 Curul : SENADO }</pre>		
<pre>HTTP/1.1 400 { LA CONSULTA NO FUE EXITOSA, INTENTE MAS TARDE }</pre>		

13. ADICION DE VOTOS

Método	POST	
URL	https://[URL_SERVER]/votaciones/addVoto	
Parámetros		
Campo	Tipo	Descripción
		N/A
Body	JSON	{ Id_persona : number Id mesa : number Id candidato : number }
Respuesta		
Campo	Tipo	Descripción
http response	string	Ok
Ejemplo respuesta		
<pre>HTTP/1.1 200 OK { Ok }</pre>		
<pre>HTTP/1.1 400 { DATOS INCONSISTENTES PARA REGISTRAR EL VOTO }</pre>		

- Mockups

