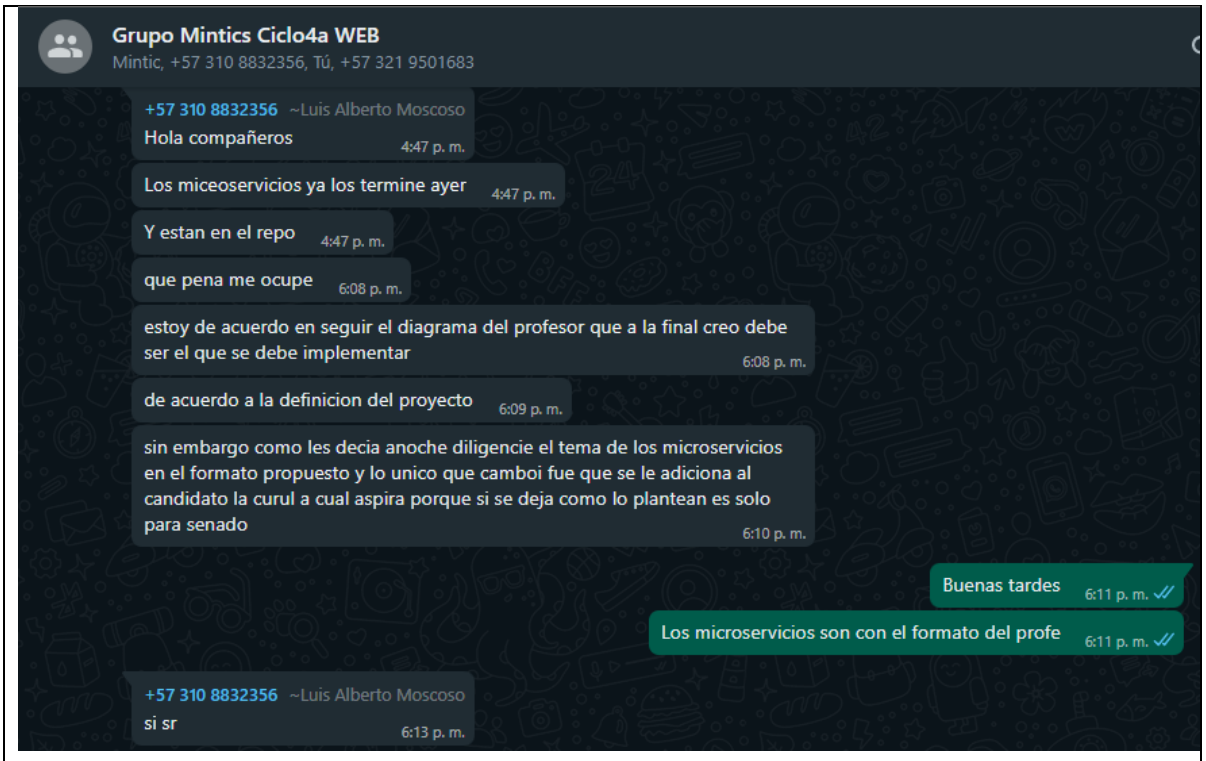
Formato de Informe de Seguimiento

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:



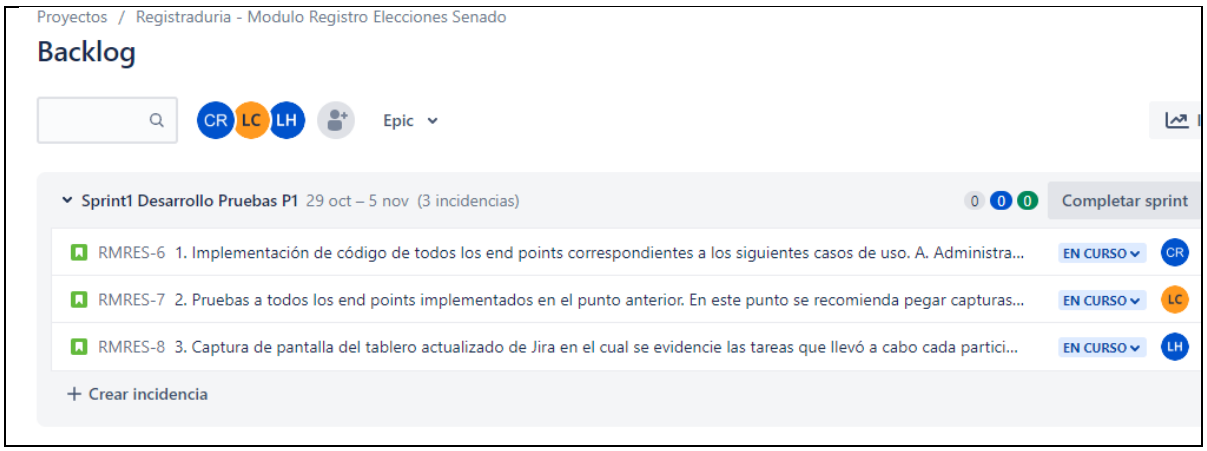
   Observaciones:

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:

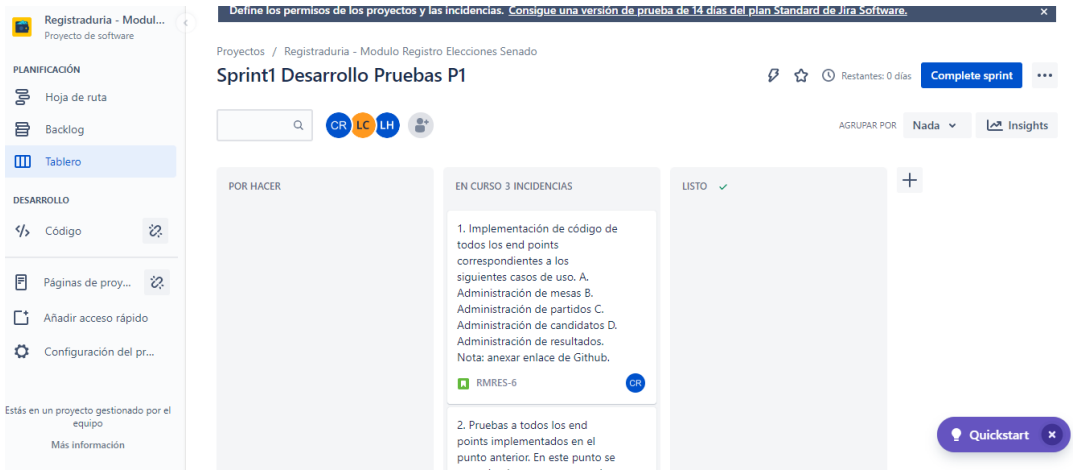
A través de Jira se organizo y distribuyo las tareas a realizar para el sprint 1

4. Reunión diaria de seguimiento - jueves.
   Pantallazo:

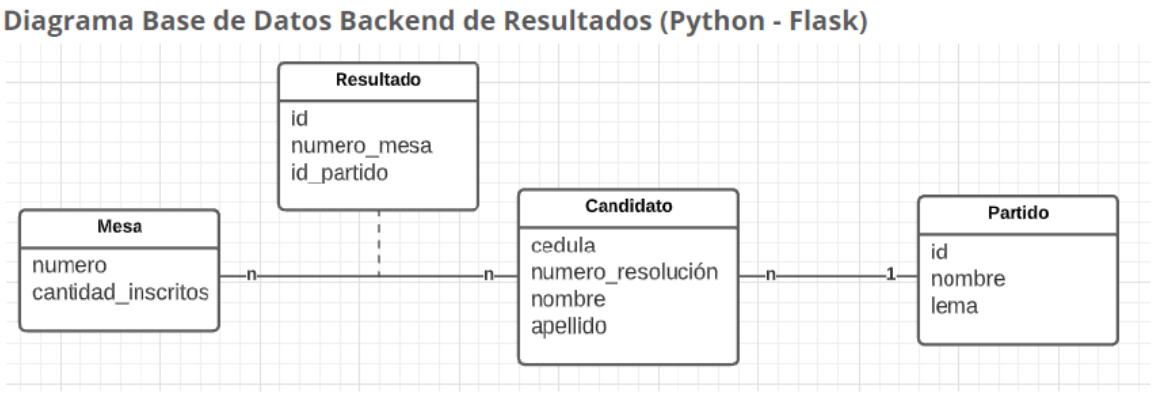
   Observaciones:

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

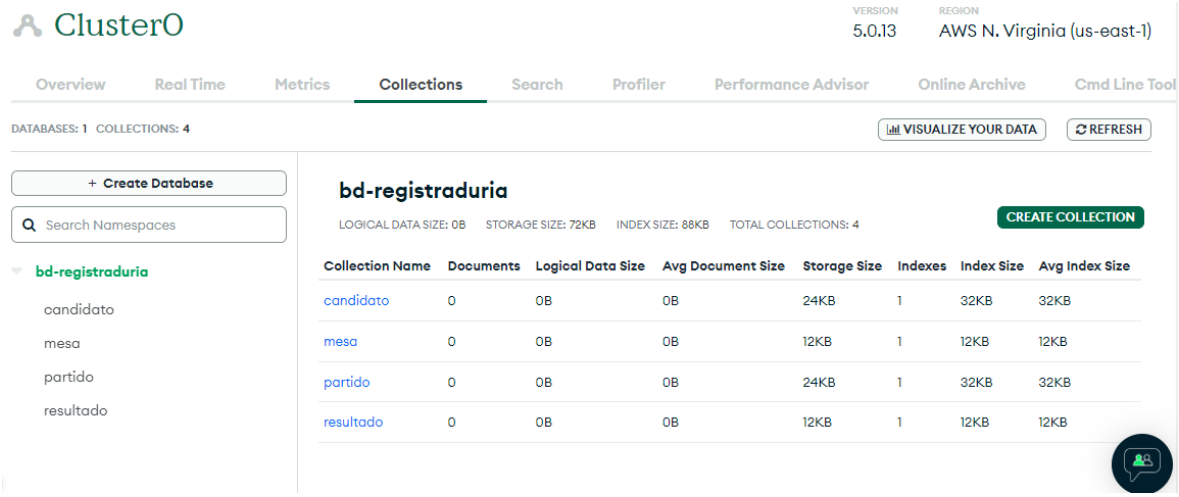


Para el desarrollo de este sprint se comenzó a elaborar el código en backend con Python en el ide PyCharm en el cual se establecieron los diferentes modelos con sus respectivos controladores, todo esto teniendo en cuenta el diagrama de resultados planteado en clases.



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.

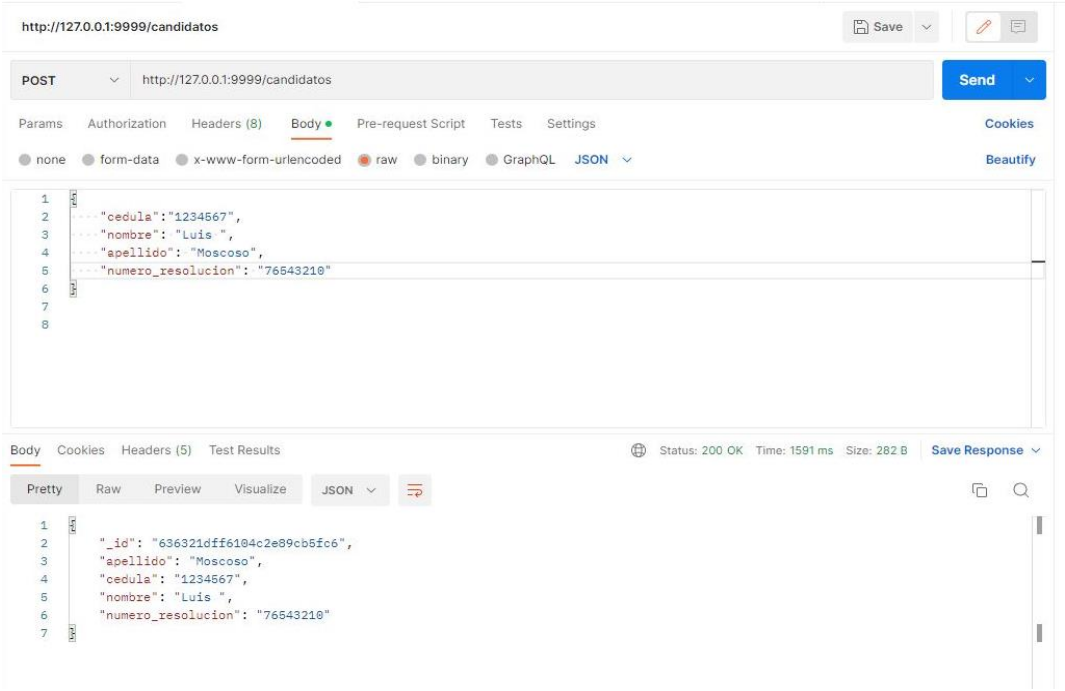


Después de vincular la base de datos en mongo con nuestro código en Python-Flask, se realizaron las pruebas de funcionamiento con los diferentes end-points propuestos en el código que se presentaba en las guías teniendo como resultado lo siguiente:

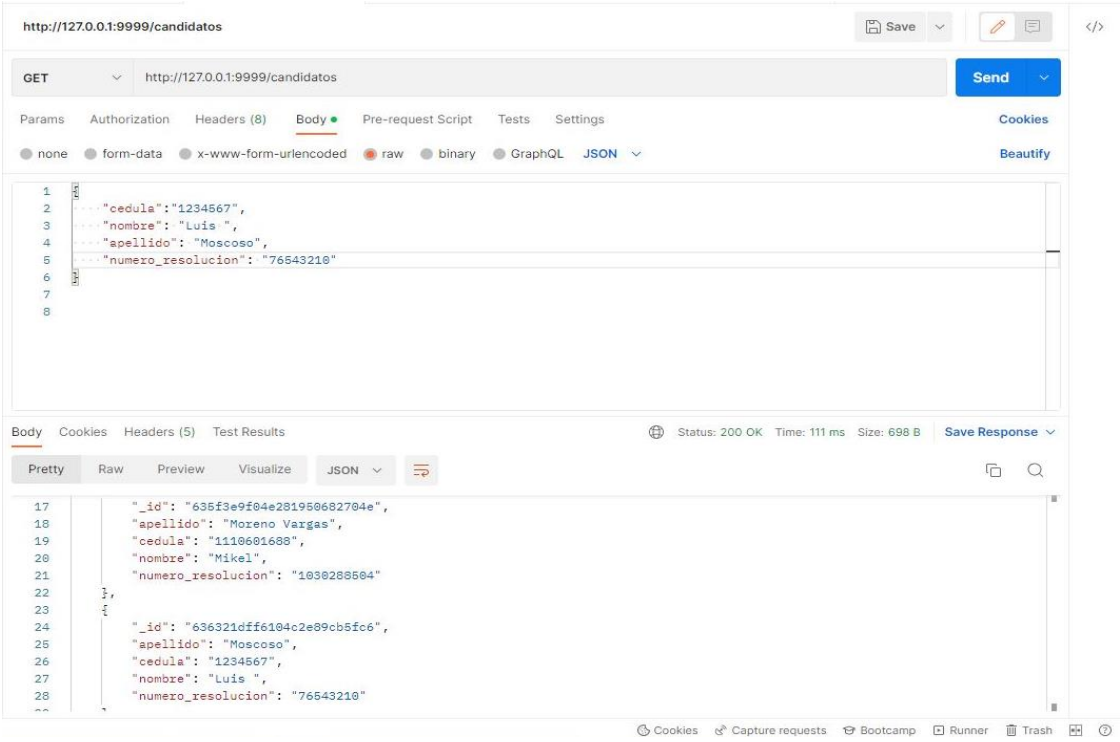
Pruebas con End-Points

Candidatos:

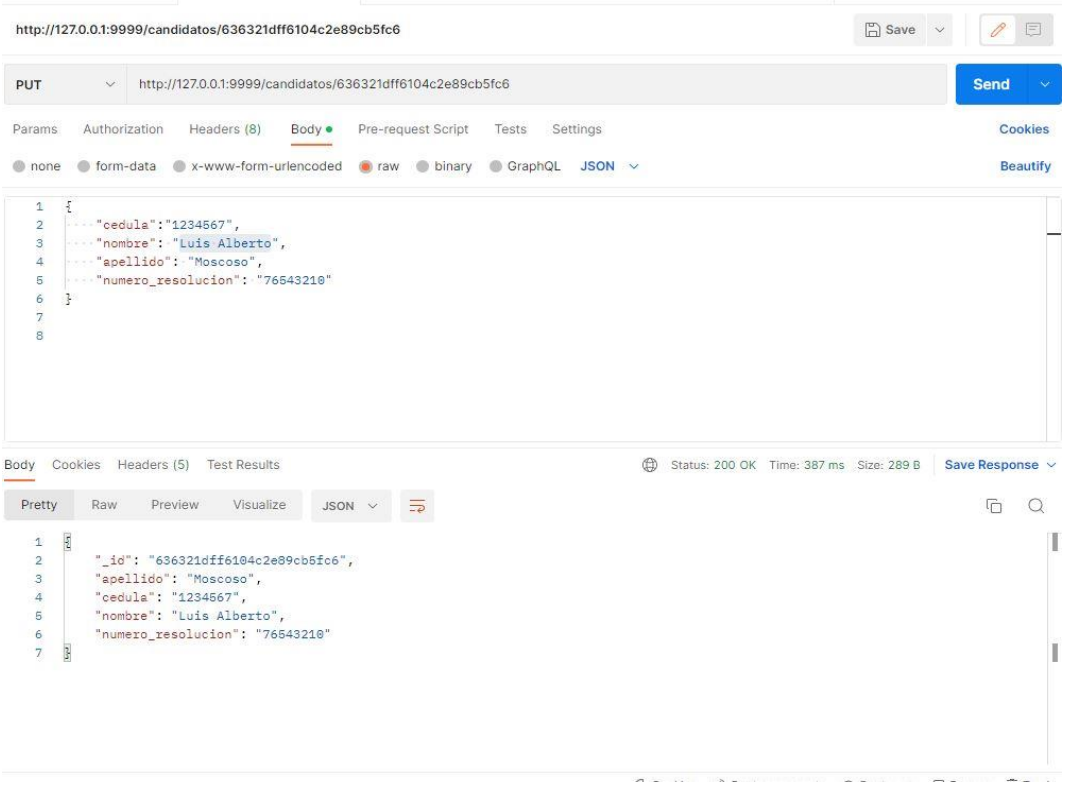
- Post



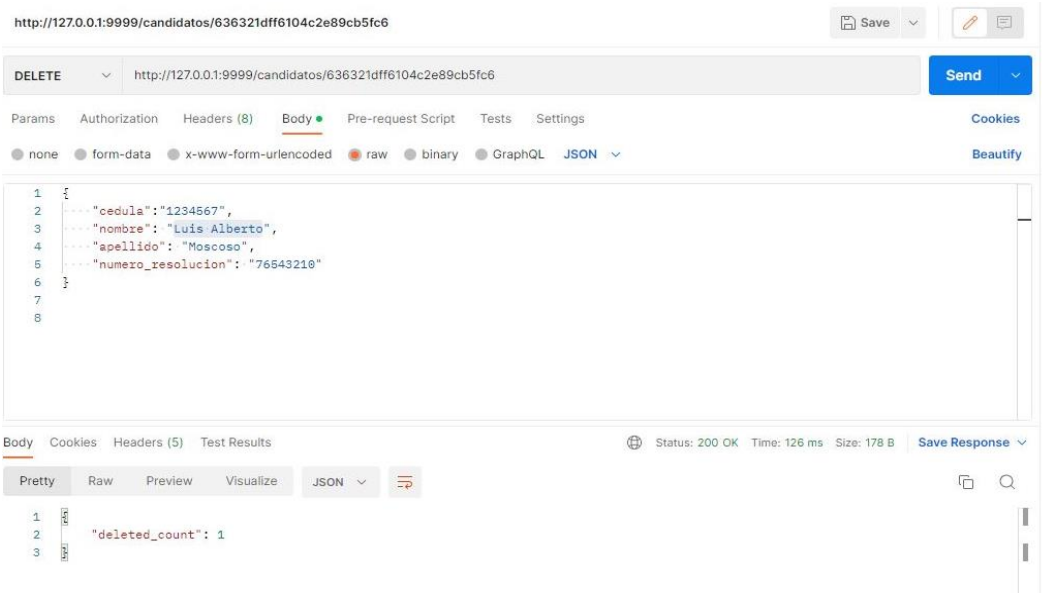
- Get



- Put

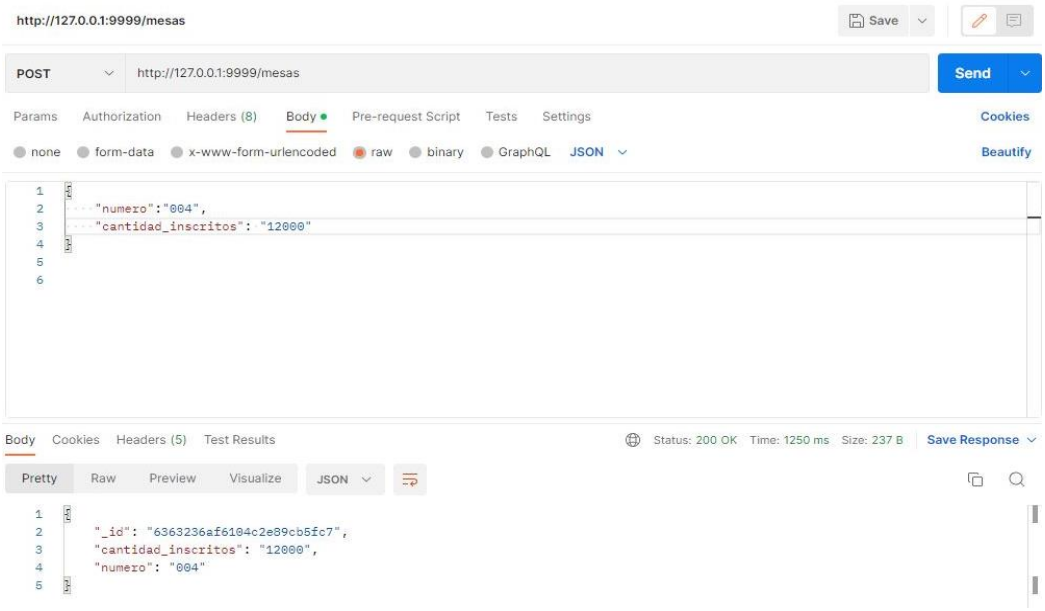

http://127.0.0.1:9999/candidatos/636321dff6104c2e89cb5fc6
Save
PUT
http://127.0.0.1:9999/candidatos/636321dff6104c2e89cb5fc6
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"cedula":"1234567",
"nombre": "Luis Alberto",
"apellido": "Moscoso",
"numero_resolucion": "76543210"
}
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 387 ms
Size: 289 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"_id": "636321dff6104c2e89cb5fc6",
"apellido": "Moscoso",
"cedula": "1234567",
"nombre": "Luis Alberto",
"numero_resolucion": "76543210"
}

- Delete

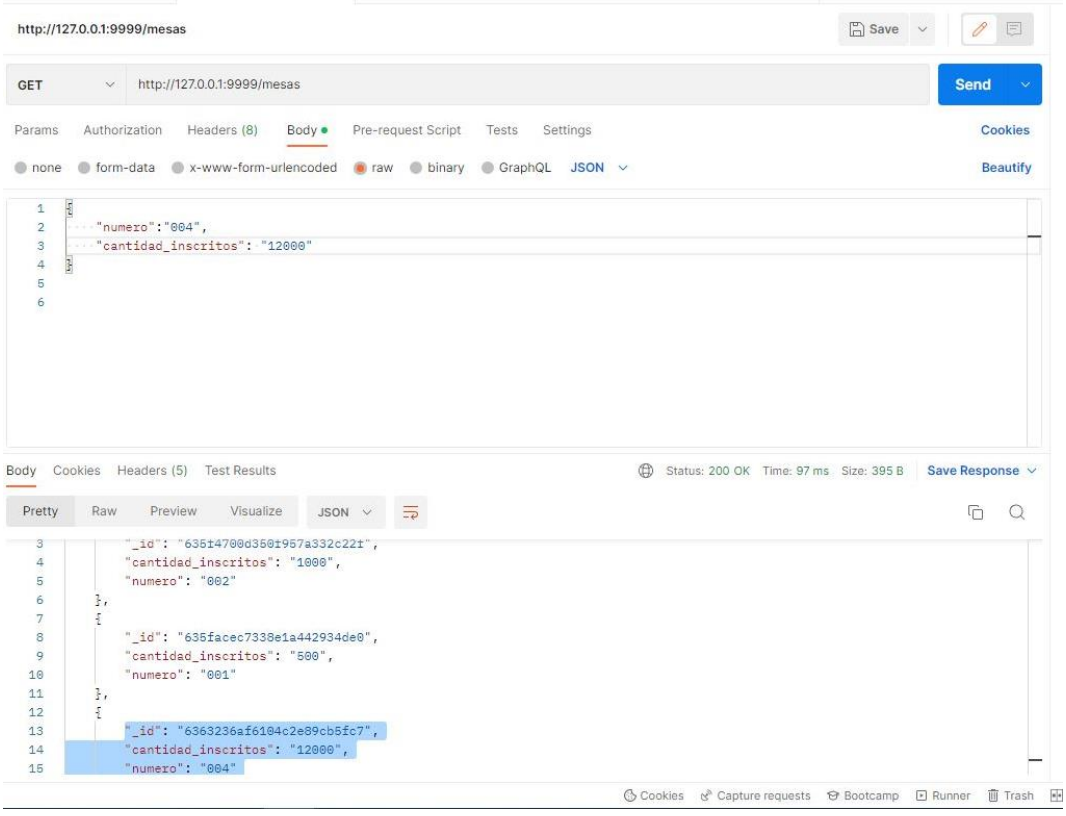

http://127.0.0.1:9999/candidatos/636321dff6104c2e89cb5fc6
Save
DELETE
http://127.0.0.1:9999/candidatos/636321dff6104c2e89cb5fc6
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"cedula":"1234567",
"nombre": "Luis Alberto",
"apellido": "Moscoso",
"numero_resolucion": "76543210"
}
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 126 ms
Size: 178 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"deleted_count": 1
}

Mesas:

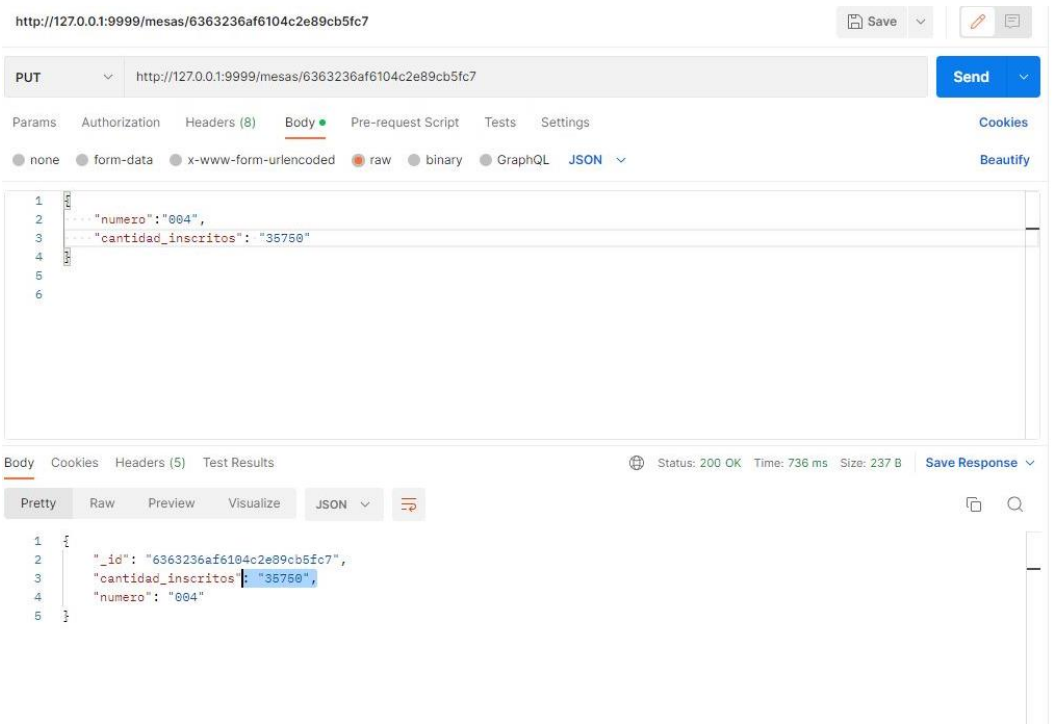
- Post



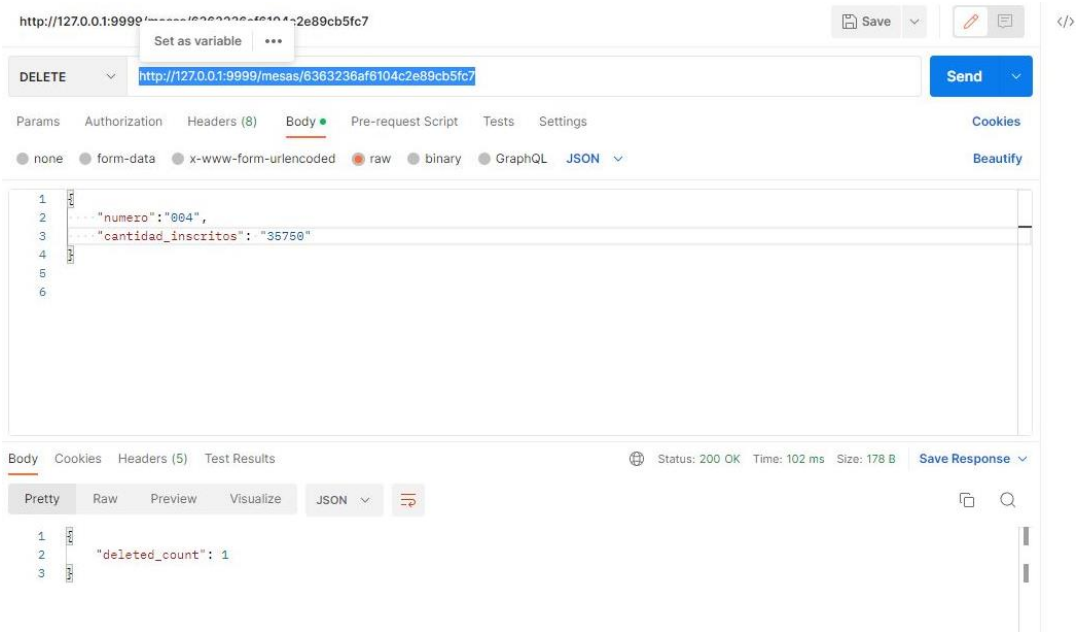
- Get



- Put

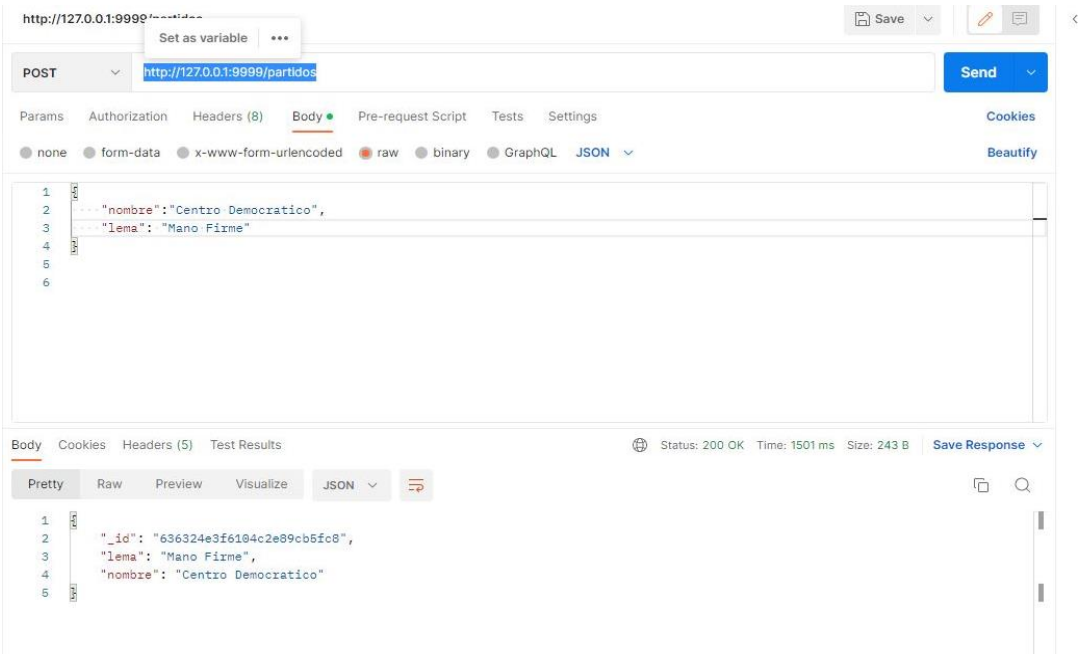

http://127.0.0.1:9999/mesas/6363236af6104c2e89cb5fc7
Save
PUT
http://127.0.0.1:9999/mesas/6363236af6104c2e89cb5fc7
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"numero":"004",
"cantidad_inscritos": "35750"
}
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 736 ms
Size: 237 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"_id": "6363236af6104c2e89cb5fc7",
"cantidad_inscritos": "35750",
"numero": "004"
}

- Delete

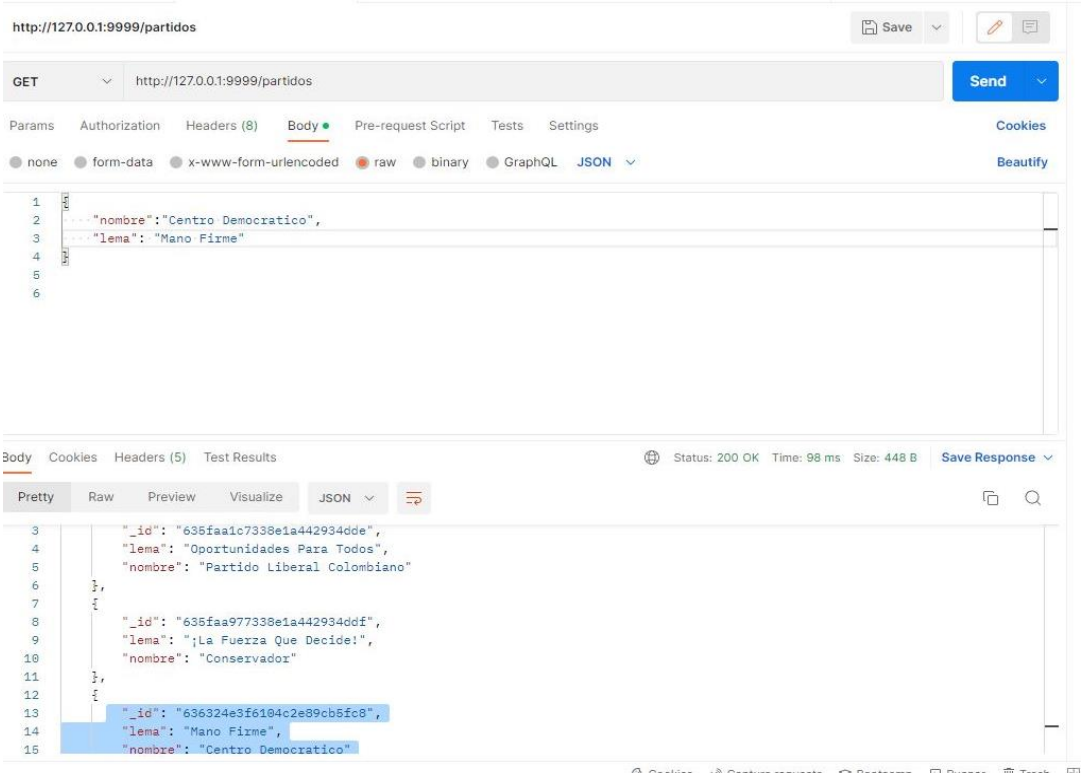

http://127.0.0.1:9999
Set as variable
Save
DELETE
http://127.0.0.1:9999/mesas/6363236af6104c2e89cb5fc7
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"numero":"004",
"cantidad_inscritos": "35750"
}
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 102 ms
Size: 178 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"deleted_count": 1
}

Partidos:

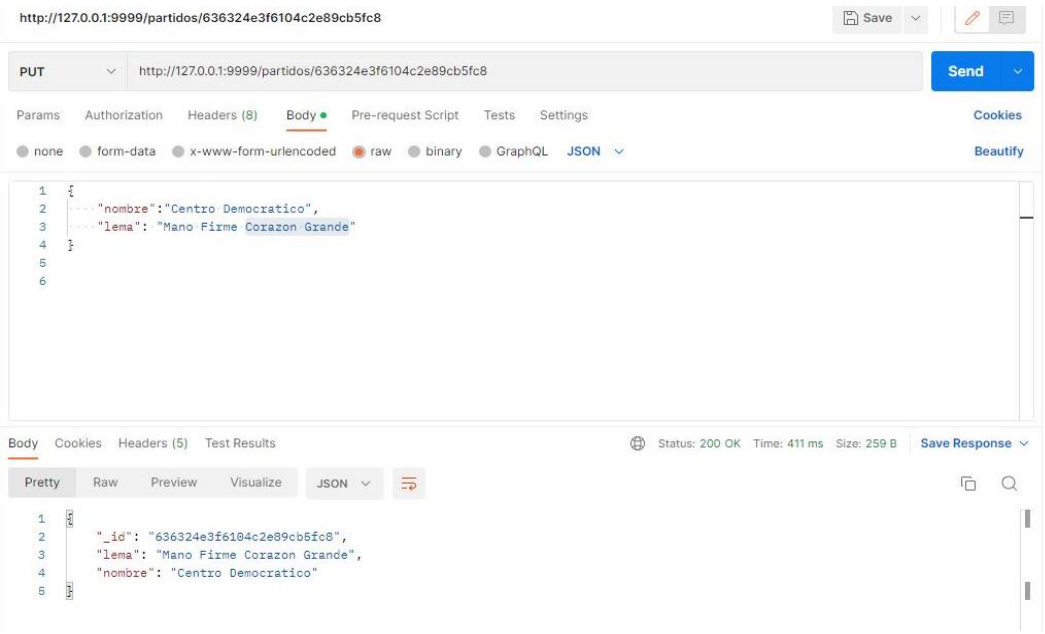
- Post



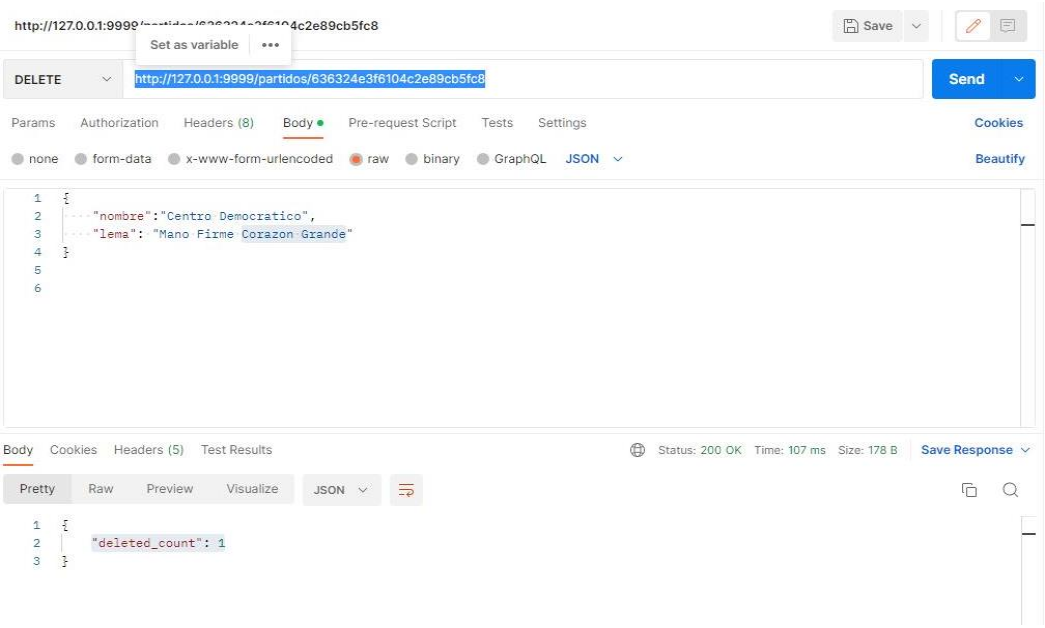
- Get



- Put

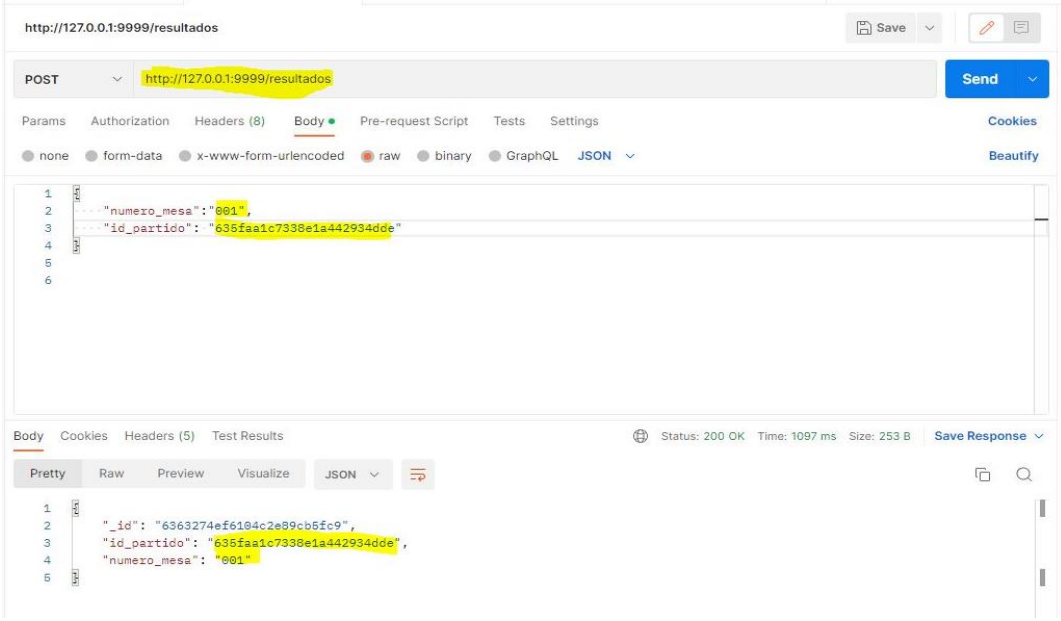

http://127.0.0.1:9999/partidos/636324e3f6104c2e89cb5fc8
Save
PUT
http://127.0.0.1:9999/partidos/636324e3f6104c2e89cb5fc8
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"nombre":"Centro Democratico",
"lema": "Mano Firme Corazon Grande"
}
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 411 ms
Size: 259 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"_id": "636324e3f6104c2e89cb5fc8",
"lema": "Mano Firme Corazon Grande",
"nombre": "Centro Democratico"
}

- Delete

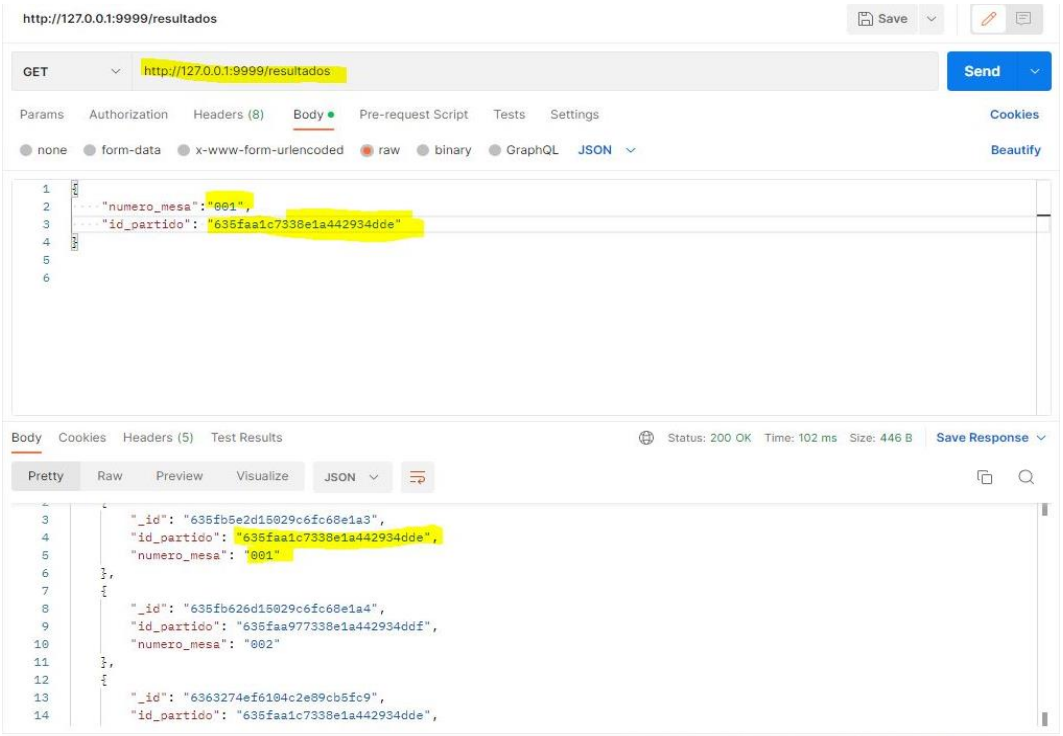

http://127.0.0.1:9999
Set as variable
Save
DELETE
http://127.0.0.1:9999/partidos/636324e3f6104c2e89cb5fc8
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"nombre":"Centro Democratico",
"lema": "Mano Firme Corazon Grande"
}
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 107 ms
Size: 178 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"deleted_count": 1
}

Resultados

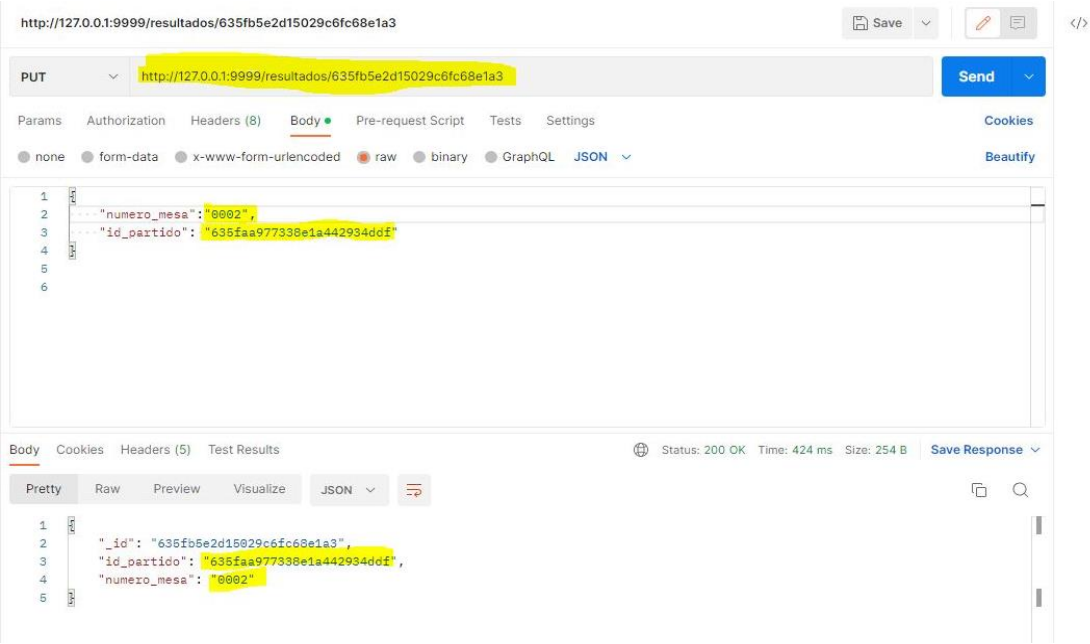
- Post



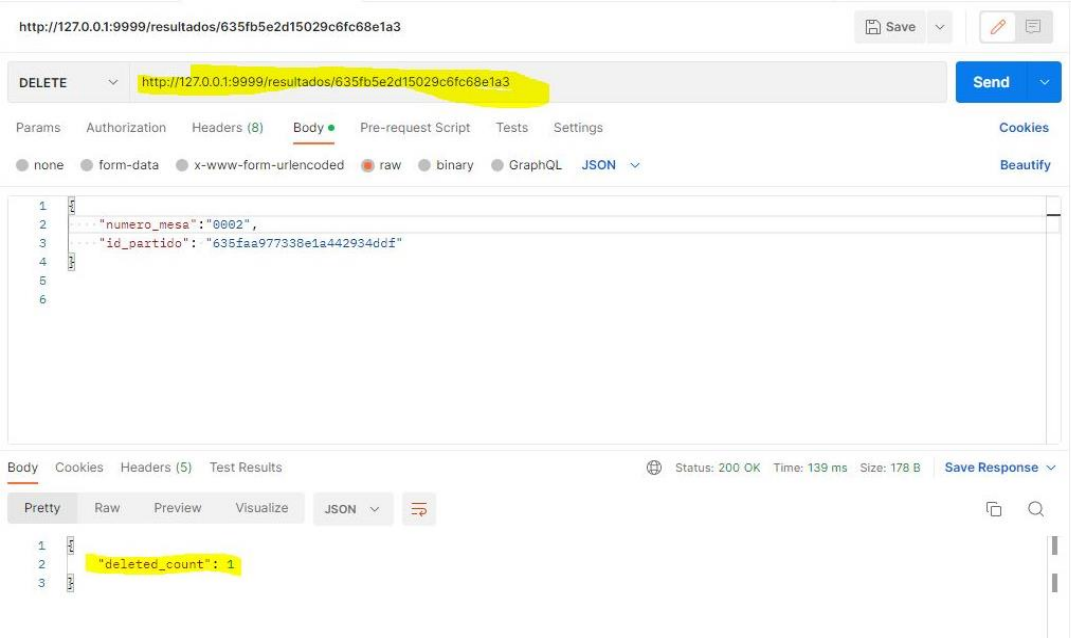
- Get



- Put



- Delete



Las pruebas realizadas fueran 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		

```
HTTP/1.1 200 OK
{
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbl90eXBlIjoiYWNjZXNzIiwiZXhwIjoxNjY2NzkzNDI5LCJqdGkiOiI5YjQ2MzAyNzY5ZTM0MjIyYmU0OTFhOGY3Mjc2OTE3ZCIsInVzZXJfaWQiOjEwMDF9.1AKm55aPMS9IvBk0BveK3vJdKLo_sGBLfcF849icnPE
}
```

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		

```
HTTP/1.1 200 OK
{
Ok
```

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		

```
HTTP/1.1 200 OK
{
  Ok
}
```

```
HTTP/1.1 400
{
   USUARIO NO PUDO SER CREADO
}
```

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		

```
HTTP/1.1 200 OK
{
  Ok
}
```

```
HTTP/1.1 400
{
   USUARIO NO PUDO SER ACTUALIZADO
}
```

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		

```
HTTP/1.1 200 OK
{
  Ok
}
```

```
HTTP/1.1 400
{
   CREDENCIALES NO VALIDAS
}
```

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		

```
HTTP/1.1 200 OK
{
  Cedula : 123
  Nombre : PEDRO PEREZ
  Resolución : 32131532321
  Curul : SENADO
}
```

```
HTTP/1.1 400
{
   CANDIDATO NO PUDO SER CREADO
}
```

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		

```
HTTP/1.1 200 OK
{
  Ok
}
```

```
HTTP/1.1 400
{
   Id CANDIDATO NO ENCONTRADO
}
```

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		

```
HTTP/1.1 200 OK
{
  Ok
}
```

```
HTTP/1.1 400
{
   Id CANDIDATO NO ENCONTRADO
}
```

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		

```
HTTP/1.1 200 OK
{
  Cedula : 123
  Nombre : PEDRO PEREZ
  Resolución : 32131532321
  Curul : SENADO
}
```

```
HTTP/1.1 400
{
   LA CONSULTA NO FUE EXITOSA, INTENTE MAS TARDE
}
```

13. ADICION DE VOTOS

Método	POST	
URL	`https://[URL_SERVER]/votaciones/addVoto`	
Parámetros		
Campo	**Tipo**	**Descripción**
		N/A

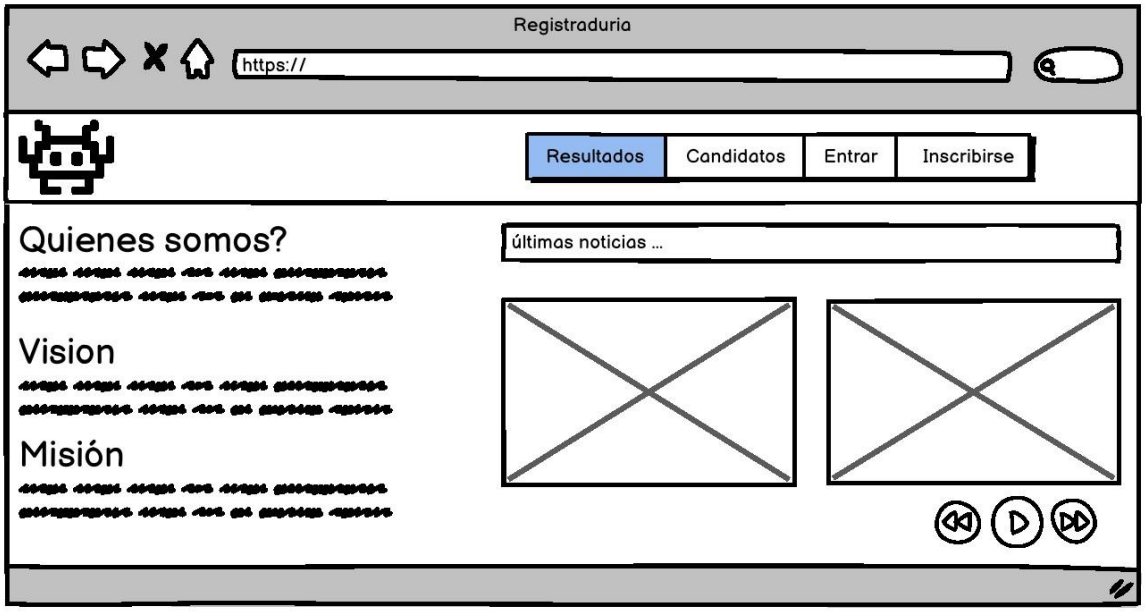
Body	JSON	{
		Id_persona : number
		Id mesa : number
		Id candidato : number
		}

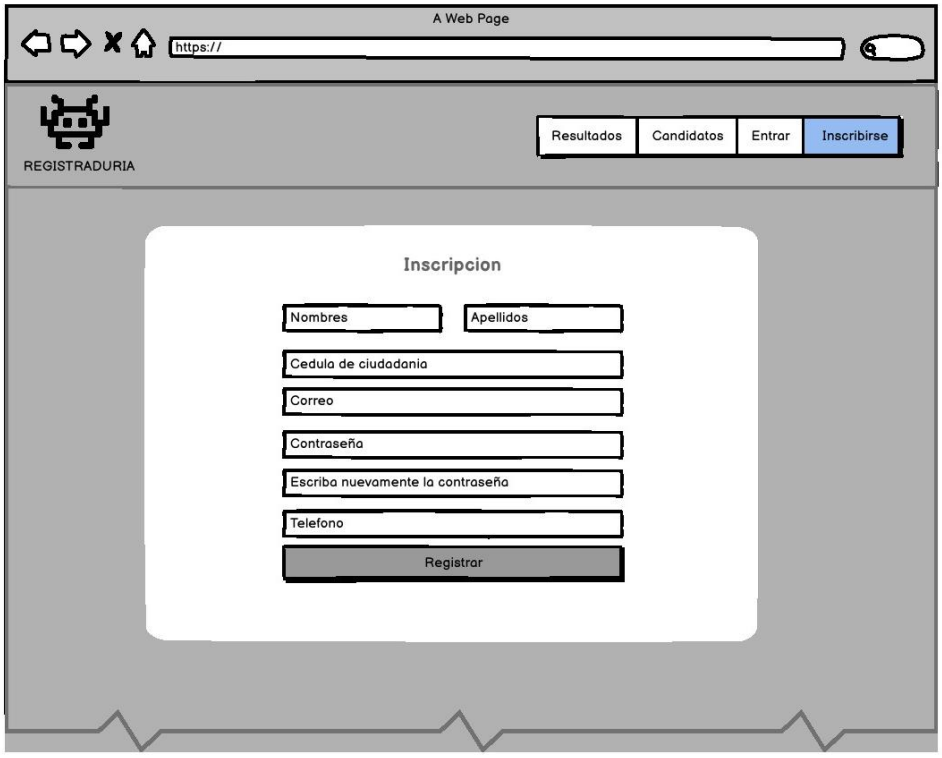
Respuesta		
Campo	**Tipo**	**Descripción**
http response	string	Ok
Ejemplo respuesta		

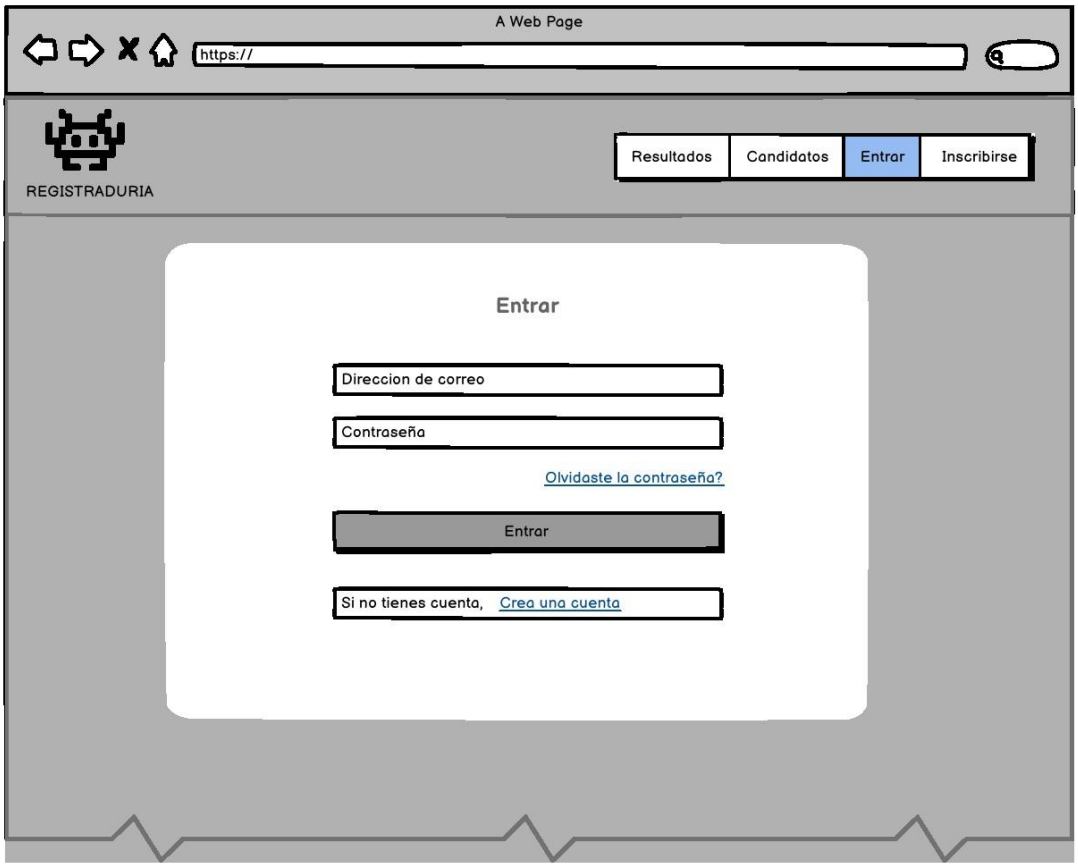
```
HTTP/1.1 200 OK
{
  Ok
}
```

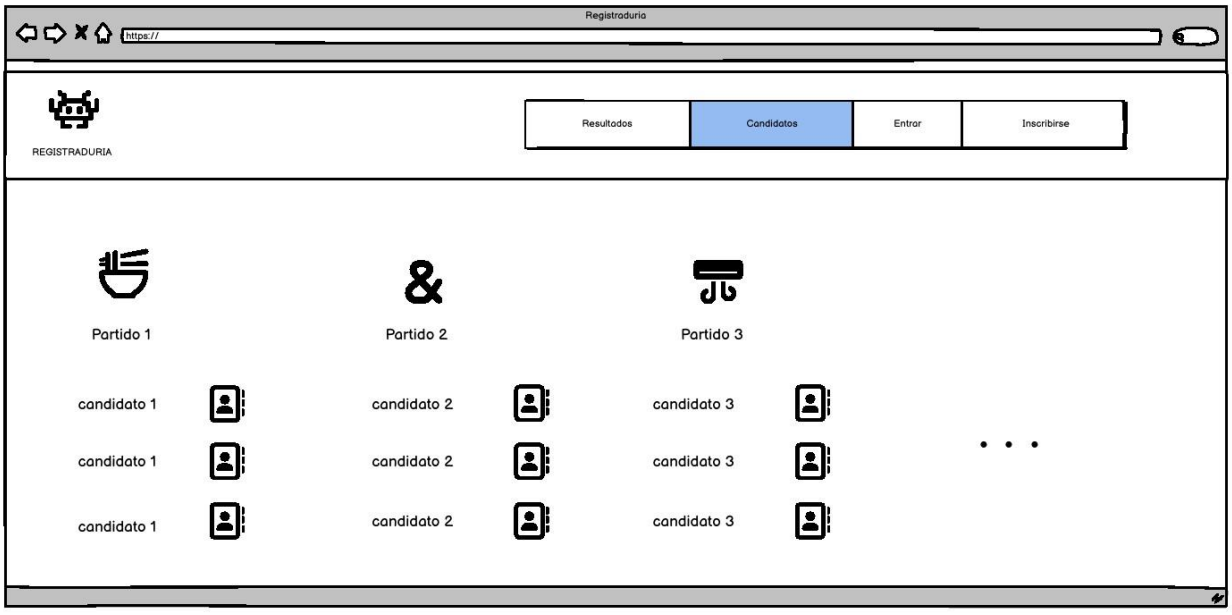
```
HTTP/1.1 400
{
   DATOS INCONSISTENTES PARA REGISTRAR EL VOTO
}
```

- Mockups


Registraduria
https://
Resultados
Candidatos
Entrar
Inscribirse
Quienes somos?
Vision
Misión
últimas noticias ...


A Web Page
https://
REGISTRADURIA
Resultados
Candidatos
Entrar
Inscribirse
Inscripcion
Nombres
Apellidos
Cedula de ciudadania
Correo
Contraseña
Escriba nuevamente la contraseña
Telefono
Registrar


A Web Page
https://
REGISTRADURIA
Resultados
Candidatos
Entrar
Inscribirse
Entrar
Direccion de correo
Contraseña
Olvidaste la contraseña?
Entrar
Si no tienes cuenta, Crea una cuenta


Registraduria
https://
REGISTRADURIA
Resultados
Candidatos
Entrar
Inscribirse
Partido 1
Partido 2
Partido 3
candidato 1
candidato 2
candidato 3
candidato 1
candidato 2
candidato 3
candidato 1
candidato 2
candidato 3


Registraduria
https://
REGISTRADURIA
Resultados
Candidatos
Entrar
Inscribirse
Partido 1
Partido 2
Partido 3
candidato 1
candidato 1
candidato 2
candidato 2
candidato 3
candidato 3