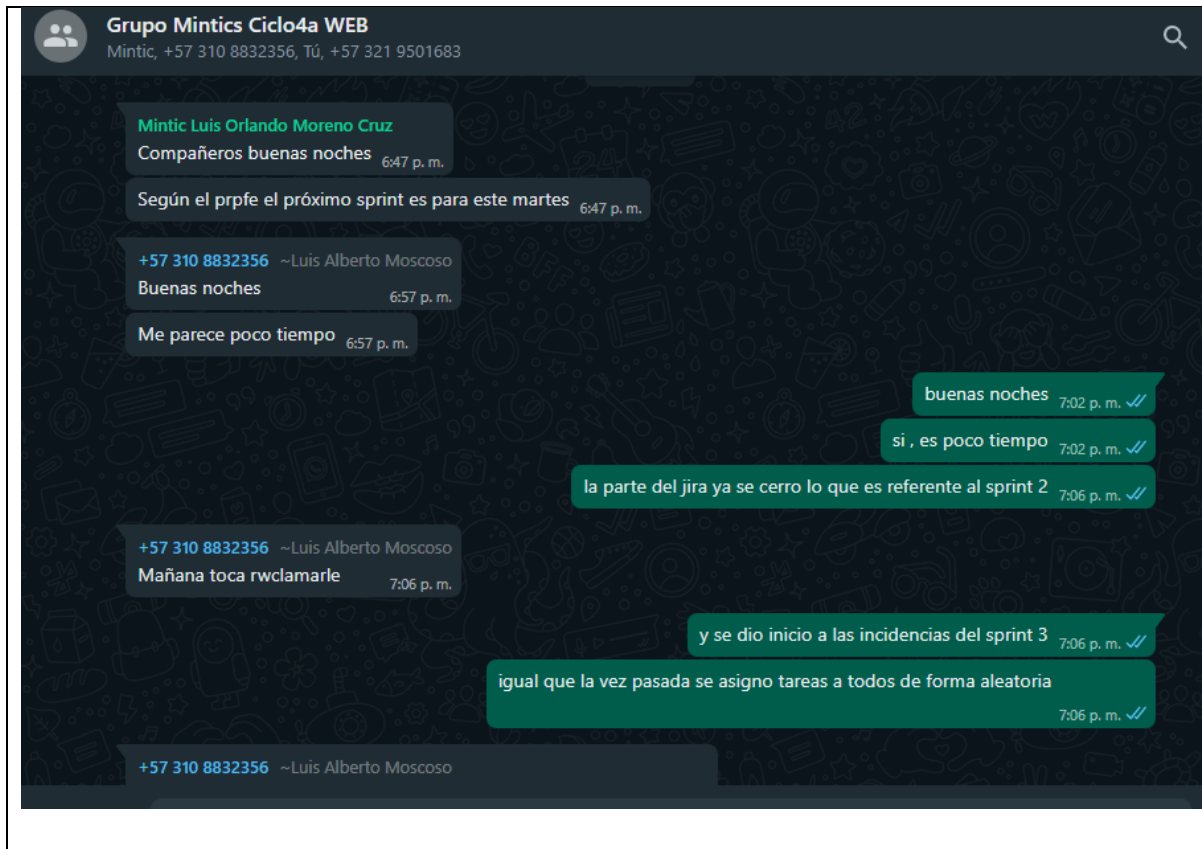


## Formato de Informe de Seguimiento

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

### 1. Primera reunión (plan inicial del sprint) - domingo.

Pantallazo:



Observaciones:

Reunión de inicio para el sprint 3 en el cual se plantearon las diferentes tareas para el desarrollo de éste

## 2. Reunión diaria de seguimiento - Lunes.

Pantallazo:

Proyectos / Registraduría - Modulo Registro Elecciones Senado

### Sprint3 Desarrollo Pruebas API

CR LH LC LT

**POR HACER 2 INCIDENCIAS**

5. Se solicita que para el punto 3 y 4 de esta entrega se anexen las capturas de imagen tanto de la petición como de la respuesta obtenida, se recomienda utilizar Postman.

RMRES-16

6. Captura de pantalla del tablero actualizado de Jira en el cual se evidencie las tareas que llevó a cabo cada participante del grupo.

RMRES-17

**EN CURSO 4 INCIDENCIAS**

2. Implementación de los middlewares. Envío del enlace del Github correspondiente.

RMRES-13 LH

1. Implementación de código del api Gateway. Envío del enlace del Github correspondiente.

RMRES-12 LC

3. Pruebas al middleware en donde se evidencie como los usuarios con determinados roles asignados previamente pueden acceder a los servicios propios de su rol.

RMRES-14 CR

4. Pruebas al middleware en donde se evidencie como los usuarios que no tienen permisos se les prohíba continuar con las transacciones que no son propias de su rol.

RMRES-15 LT

**LISTO ✓**

Observaciones:

A través de Jira se organizó y distribuyó las tareas a realizar para el sprint 3

## 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>

Proyectos / Registraduria - Modulo Registro Elecciones Senado

### Backlog

Task ID	Description	Status	Assignee
RMRES-13	2. Implementación de los middlewares. Envío del enlace del Github correspondiente.	EN CURSO	LH
RMRES-12	1. Implementación de código del api Gateway. Envío del enlace del Github correspondiente.	EN CURSO	LC
RMRES-14	3. Pruebas al middleware en donde se evidencie como los usuarios con determinados roles asignados previamente...	EN CURSO	CR
RMRES-15	4. Pruebas al middleware en donde se evidencie como los usuarios que no tienen permisos se les prohíba continu...	EN CURSO	LT
RMRES-16	5. Se solicita que para el punto 3 y 4 de esta entrega se anexen las capturas de imagen tanto de la petició...	TAREAS POR HACER	
RMRES-17	6. Captura de pantalla del tablero actualizado de Jira en el cual se evidencie las tareas que llevó a cabo ca...	TAREAS POR HACER	

Implementación de código del api Gateway. Implementación de los middlewares. Envío del Pruebas al middleware en donde se evidencie como los usuarios con determinados roles asignados previamente pueden acceder a los servicios propios de su rol. Pruebas al middleware en donde se evidencie como los usuarios que no tienen permisos se les prohíba continuar con las transacciones que no son propias de su rol

La codificación implementada se encuentra en el siguiente enlace

[https://github.com/luismoscoso49/MINTIC\\_CICLO4/tree/main/BACK\\_END/Api%20Gateway](https://github.com/luismoscoso49/MINTIC_CICLO4/tree/main/BACK_END/Api%20Gateway)

## Implementación de código del api Gateway

```
1 {
2   "url-backend": "127.0.0.1",
3   "port": 7777,
4   "url-backend-security": "http://127.0.0.1:8080",
5   "url-backend-results": "http://127.0.0.1:9999"
6 }
```

## Implementación del código middleware

```
19 # -----Middleware-----#
20 @app.route("/login", methods=["POST"])
21 def create_token():
22     data = request.get_json()
23     headers = {"Content-Type": "application/json; charset=utf-8"}
24     url=dataConfig["url-backend-security"]+"/usuarios/validar"
25     response = requests.post(url, json=data, headers=headers)
26     if response.status_code == 200:
27         user = response.json()
28         expires = datetime.timedelta(seconds=60 * 60*24)
29         access_token = create_access_token(identity=user,expires_delta=expires)
30         return jsonify({"token": access_token, "user_id": user["_id"]})
31     else:
32         return jsonify({"msg": "Bad username or password"}), 401
33
34 @app.before_request
35 def before_request_callback():
36     endPoint=limpiarURL(request.path)
37     excludedRoutes=["/login"]
38     if excludedRoutes.__contains__(request.path):
39         pass
40     elif verify_jwt_in_request():
41         usuario = get_jwt_identity()
42         if usuario["rol"] is not None:
43             tienePermiso=validarPermiso(endPoint,request.method,usuario["rol"]["_id"])
44             if not tienePermiso:
```

Luego de haber realizado la respectiva estructuración del código en pycharm para el middleware, se realizaron pruebas con usuarios a los cuales previamente se les a asignado unos roles como lo son administrador, jurado y ciudadano.

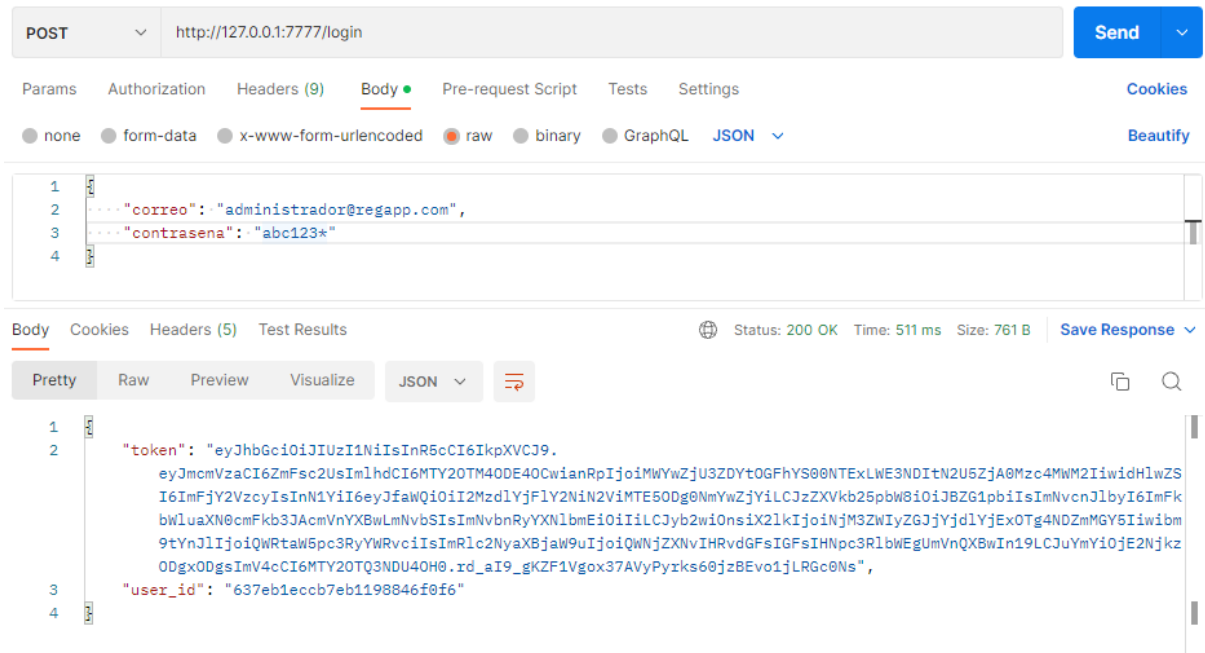
GET http://127.0.0.1:8080/usuarios

Status: 200 OK Time: 2.00 s Size: 896 B

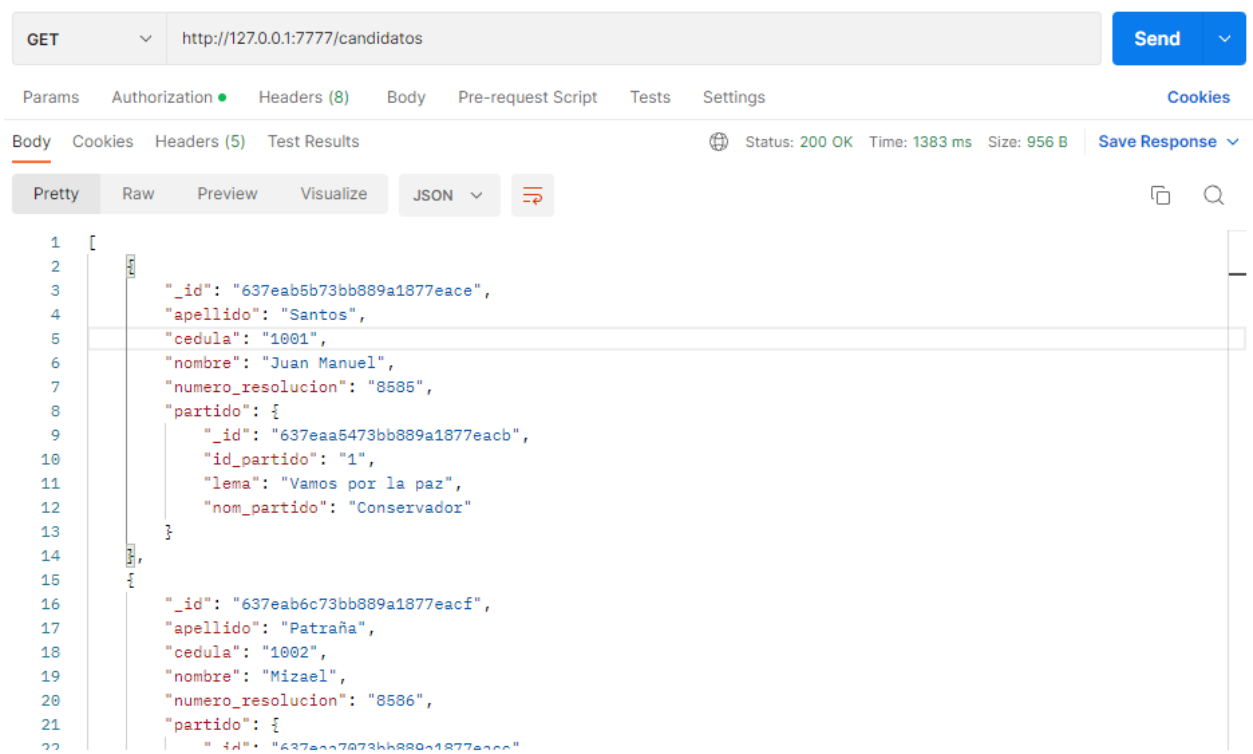
```
3  {
4    "_id": "637eb1eccb7eb1198846f0f6",
5    "seudonimo": "Admin",
6    "correo": "administrador@regapp.com",
7    "contrasena": "959e124c7187cbcdf4dac46873101167f95f6441d2539982872524f4006541b0",
8    "rol": {
9      "_id": "637eb2dbcb7eb1198846f0f9",
10     "nombre": "Administrador",
11     "descripcion": "Acceso total al sistema RegApp"
12   }
13 },
14 {
15   "_id": "637eb228cb7eb1198846f0f7",
16   "seudonimo": "Jurado",
17   "correo": "jurado@regapp.com",
18   "contrasena": "107fdb24f911e47df36c63fdabd4de0e62f186a862046ed9a0d8a248d1ac4aac",
19   "rol": null
20 },
21 {
22   "_id": "637eb243cb7eb1198846f0f8",
23   "seudonimo": "Ciudadano",
24   "correo": "ciudadano@regapp.com",
```

Como se puede observar en la anterior imagen solo el usuario Admin tiene un rol asignado que es el de Administrador, este usuario podrá acceder a unas rutas a través

de unos permisos que se le fueron otorgados previamente a ese rol. Primero accedo al login en el cual se nos asigna un token



Luego con el token obtenido me dirijo a la ruta del api Gateway que a través de la verificación que hace el middleware me dará el respectivo acceso a las rutas establecidas anteriormente.



Probamos otra ruta como con el método delete para un determinado dato con su respectivo id y se obtuvo lo siguiente

The screenshot shows a REST client interface with the following details:

- Method:** DELETE
- URL:** http://127.0.0.1:7777/candidatos/6380d814920d9e2b74be80cc
- Status:** 200 OK
- Time:** 1029 ms
- Size:** 178 B
- Body:** {"deleted\_count": 1}

The response body is displayed in a code editor with line numbers 1, 2, and 3.

Finalmente, a este usuario no se le han asignado diferentes rutas de acceso, a continuación, se puede verificar la intención de ingresar a una ruta diferente con su respuesta de acceso denegado

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** http://127.0.0.1:7777/resultados
- Status:** 401 UNAUTHORIZED
- Time:** 236 ms
- Size:** 200 B
- Body:** {"message": "Permission denied"}

The response body is displayed in a code editor with line numbers 1, 2, and 3.

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

## Pruebas de usuario sin acceso

Luego verificaremos el acceso de un usuario que aun no tiene un rol asignado

The screenshot shows a REST client interface with a POST request to `http://127.0.0.1:8080/usuarios/validar`. The request body is a JSON object with the following fields:

```
{  "correo": "ciudadano@regapp.com",  "contrasena": "abc789*"}
```

The response is a JSON object with the following fields:

```
{  "_id": "637eb243cb7eb1198846f0f8",  "seudonimo": "Ciudadano",  "correo": "ciudadano@regapp.com",  "contrasena": "",  "rol": null}
```

The status of the response is 200 OK, with a time of 118 ms and a size of 370 B.

Luego le asignamos un rol a este usuario, en este caso solo tendrá permisos de lectura como se puede observar en la descripción

The screenshot shows a REST client interface with a PUT request to `http://127.0.0.1:8080/usuarios/637eb243cb7eb1198846f0f8/rol/637eb345cb7eb1198846f0fb`. The request body is a JSON object with the following fields:

```
{  "_id": "637eb243cb7eb1198846f0f8",  "seudonimo": "Ciudadano",  "correo": "ciudadano@regapp.com",  "contrasena": "76404b43d7f817377db797a7aefc5dee2d6ff142998d55d2ceaff0606796898c",  "rol": {    "_id": "637eb345cb7eb1198846f0fb",    "nombre": "Ciudadano",    "descripcion": "Visualizacion de reportes "  }}
```

The response is a JSON object with the following fields:

```
{  "_id": "637eb243cb7eb1198846f0f8",  "seudonimo": "Ciudadano",  "correo": "ciudadano@regapp.com",  "contrasena": "76404b43d7f817377db797a7aefc5dee2d6ff142998d55d2ceaff0606796898c",  "rol": {    "_id": "637eb345cb7eb1198846f0fb",    "nombre": "Ciudadano",    "descripcion": "Visualizacion de reportes "  }}
```

The status of the response is 200 OK, with a time of 562 ms and a size of 528 B.

Después de tener su respectivo rol le asignamos algunos permisos para el acceso al backend de resultados, en este caso solo podrá ver la lista de candidatos o un candidato con su respectivo número de id:

POST ▼ http://127.0.0.1:8080/permisos-roles/rol/637eb345cb7eb1198846f0fb/permiso/637aa93a4c1a2e16f91239b6 Send ▼

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

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (8) Test Results 🌐 Status: 201 Created Time: 386 ms Size: 477 B Save Response ▼

Pretty Raw Preview Visualize JSON ▼ 🔍

```
1 {
2   "_id": "6380dd68b4aa2d3f475043c9",
3   "rol": {
4     "_id": "637eb345cb7eb1198846f0fb",
5     "nombre": "Ciudadano",
6     "descripcion": "Visualizacion de reportes "
7   },
8   "permiso": {
9     "_id": "637aa93a4c1a2e16f91239b6",
10    "url": "/candidatos",
11    "metodo": "GET"
12  }
13 }
```

POST ▼ http://127.0.0.1:8080/permisos-roles/rol/637eb345cb7eb1198846f0fb/permiso/637aa9104c1a2e16f91239b5 Send ▼

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

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (8) Test Results 🌐 Status: 201 Created Time: 376 ms Size: 479 B Save Response ▼

Pretty Raw Preview Visualize JSON ▼ 🔍

```
1 {
2   "_id": "6380ddbeb4aa2d3f475043ca",
3   "rol": {
4     "_id": "637eb345cb7eb1198846f0fb",
5     "nombre": "Ciudadano",
6     "descripcion": "Visualizacion de reportes "
7   },
8   "permiso": {
9     "_id": "637aa9104c1a2e16f91239b5",
10    "url": "/candidatos/?",
11    "metodo": "GET"
12  }
13 }
```



The screenshot shows the Postman interface for a REST client request. The URL bar at the top displays `http://127.0.0.1:7777/login`. The 'Body' tab is selected, showing a JSON request body: 

```
{  "correo": "ciudadano@regapp.com",  "contrasena": "abc789*"}
```

. The 'Headers' tab is also visible, showing a 'Content-Type' header set to 'application/json'. The 'Response' tab is selected, displaying a JSON response body: 

```
{  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJmcmVzaCI6ZmFsc2UsIm1hdCI6MTY2OTM0Tg4NCwianRpIjoia3ZjZkYzIiwiaXNjb3R1b3R5Ijoia3ZjZkYzIiwiaWF0IjoiMTY2OTM0Tg4NCw",  "user_id": "637eb243cb7eb1198846f0f8"}
```

. The status bar at the bottom indicates a 200 OK status, 597 ms time, and 750 B size.

GET

http://127.0.0.1:7777/candidatos

Send

Params

Authorization

Headers (10)

Body

Pre-request Script

Tests

Settings

Cookies

Type

Bearer T...

Token

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...

Body

Cookies

Headers (5)

Test Results

Status: 200 OK

Time: 1121 ms

Size: 862 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

```
1  [
2    {
3      "_id": "637eab5b73bb889a1877eace",
4      "apellido": "Santos",
5      "cedula": "1001",
6      "nombre": "Juan Manuel",
7      "numero_resolucion": "8585",
8      "partido": {
9        "_id": "637eaa5473bb889a1877eacb",
10       "id_partido": "1",
11       "lema": "Vamos por la paz",
12       "nom_partido": "Conservador"
13     }
14   },
15   {
16     "_id": "637eab6c73bb889a1877eacf",
17     "apellido": "Patraña",
18     "cedula": "1002",
```

Finalmente realizamos la prueba a través del método post que según lo establecido en el código me permite crear un nuevo candidato como ejemplo fijado en esta ruta, pero como este usuario no tiene permisos para acceder a esta ruta con su respectivo método, el permiso de acceso debe ser denegado

The screenshot displays a REST client interface with the following details:

- Method:** POST
- URL:** http://127.0.0.1:7777/candidatos
- Body:** A JSON object with the following fields:

```
{  "cedula": "1010",  "numero_resolucion": "168585",  "nombre": "Carlos",  "apellido": "Perez"}
```
- Status:** 401 UNAUTHORIZED
- Time:** 441 ms
- Size:** 200 B
- Response Body:** A JSON object with the following field:

```
{  "message": "Permission denied"}
```

### Formato de Informe de Retrospectiva

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

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

Positivos
<ul style="list-style-type: none"><li>• Conocer nuevas herramientas de trabajo como lo es Api Gateway que gestiona el ingreso autorizado y no autorizado a un sistema</li><li>• Investigación sobre el tema a trabajar.</li></ul>
Negativos
<ul style="list-style-type: none"><li>• Dificultad a la hora de sincronizar el grupo de trabajo</li></ul>

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

<ul style="list-style-type: none"><li>• Afianzar el trabajo en equipo.</li><li>• Profundizar en los temas con fuentes externas.</li></ul>
---