FASE 1
iVoting

GRUPO 2

JOSE FERNANDO VALDEZ PEREZ-2015-03651
ANDREA MARÍA LÓPEZ FLORES-2014-04134
JUAN PABLO GARCÍA MONZÓN - 201222615
JHONATAN LEONEL LÓPEZ SANTOS-2013-25583

FIUSAC
Universidad de San Carlos de Guatemala

Contenido

Introducción

El voto electronico es un servicio que ha ido desarrollandose a lo largo de los años para darle una solución al evento del sufragio ya que cada vez hay mas personas que tienen dificultades para llegar a un centro de votación para hacer efectivo este derecho. Tropicalizando esta circunstancia a nuestro país vemos que tenemos muchos de los mismos problemas por lo que surgio este proyecto, iVoting, para dar una solución que pueda empezar a implementarse en nuestro sistema de votaciones.

El aspecto técnico para hacer realidad este proyecto es la finalidad de esta "Fase 1" ya que este trabajo desarrolla el siguiente contenido:
Diagramas de microservicios, contratos de los microservicios, descripción de los microservicios, control de versiones del proyecto, organización del equipo de trabajo, documentación del pipeline y una recomendación de que tecnologías se podrían utilizar.

Objetivos

Objetivo General

- **Explicar el aspecto tecnico para desarrollar iVoting**
- **Explicar la metodología de trabajo del Grupo 2**

Objetivo Especifico

- **Describir los microservicios de los cuales está compuesto iVoting**
- **Establecer contratos de los microservicios**
- **Recomendación de las tecnologías que se tienen como las óptimas para desarrollar iVoting**

Justificación

En nuestro país no existe todavía un sistema de votación electrónico para desarrollar un sufragio seguro y exacto. No obstante, este proyecto, iVoting, tiene el objetivo de crear una primera versión funcional de este tipo de solución ya que la situación sanitaria de estos últimos años esta forzando para que todos los procesos que se realizaban de forma presencial ahora se migren a una modalidad semi-presencial para resguardar la salud de las personas.

Metodología de trabajo

El coordinador del Grupo 2 es: Jose Fernando Valdez
El Grupo 2 utilizara la técnica ágil conocida como "Scrum". El equipo tiene asignado estos puestos:

Puesto	Responsable
Proyect Owner	Jose Fernando Valdez
Scrum Master	Juan Pablo García Monzón
Development Team	Andrea Lopez y Jhonathan Lopez

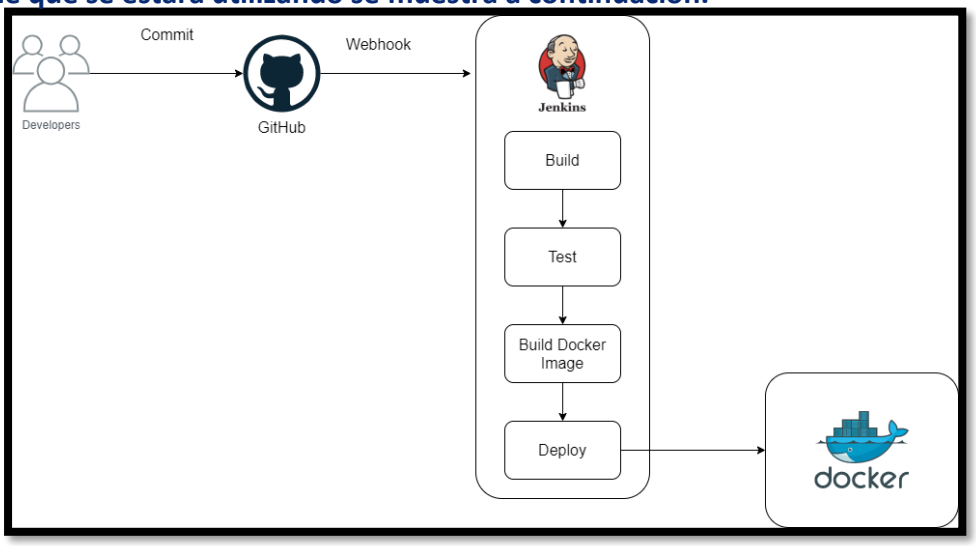
Los recursos de esta técnica que se están usando son:

- **Scrum Meetings**
- **Scrum Planning**
- **Scrum Retrospective**
- **Scrum Review**
- **Sprint**

Las Scrum Meetings se están realizando cada 2 días para gradualmente ver el progreso y los obstáculos de cada integrante.
El Scrum Planning, Retrospective y Review se realizan al terminar el Sprint.
El Sprint dura 2 semanas.

Pipeline

Un pipeline es una nueva forma de trabajar en el mundo DevOps en la integración continua.
Utilizando pipeline se puede definir el ciclo de vida completo de una aplicación.
El pipeline que se estará utilizando se muestra a continuación.



Un desarrollador genera líneas de código para una funcionalidad, al momento de finalizar esta funcionalidad se debe de comprobar si existe posibles errores en el código.
Cuando el desarrollador realiza un Commit y sube cambios a la rama principal este activara un evento para que Jenkins empiece a ejecutar cada etapa. Las etapas utilizadas son build, test, build docker y deploy, estas se describirán a continuación.

1. **Stage Build. En esta etapa se encarga de compilar el código y de instalar dependencias para generar una instancia de la aplicación.**
2. **Stage Test. En esta etapa se ejecutan pruebas para validar el comportamiento de nuestro código, en esta etapa evita errores lleguen a los usuarios finales.**
3. **Stage Build Docker. En esta etapa se encarga de construir las imágenes Docker, etiquetando la imagen y posteriormente agregando a un repositorio de Docker para después utilizarlo al momento de realizar el Deploy.**
4. **Stage Deploy. Una vez construido la imagen se inicia la aplicación a partir de la imagen creada.**

Microservicios

Seguridad

Para iVoting lo más importante es garantizar a los votantes que su voto es secreto y seguro, para que puedan tener confianza en la aplicación y se redusca la cantidad de personas que votan de forma presencial. Para esto existirá un micoservicio encargado de encriptar la información al momento de realizar un voto, es decir la información de la persona que realizó el voto. En el momento de mostrar los resultados, solamente se mostrará el todal de personas que votaron por las diferentes opciones.

De igual forma, la información de las personas registradas será encriptada para evitar el mal uso de ésta.

Para garantizar seguridad en la autenticación de los usuarios, al momento que un usuario ingrese a la plataforma se le enviará un tocken para confirmar la autenticidad del ingreso. Esto se realizará utilizando JWT (Json Web Token).

Con estas medidas se espera una aplicación segura y confiable para realizar distintas elecciones.

Comunicación entre los servicios

Diseño del bus de mensajería (Broker)
En esta arquitectura, toda la comunicación se enruta a través de un grupo de brokers. Los brokers son programas de servidor que ejecutan algunos algoritmos de enrutamiento avanzados.

Cada microservicio se conecta a un broker. El microservicio puede enviar y recibir mensajes a través de la misma conexión. El servicio que envía mensajes se llama editor y el receptor se llama suscriptor. Los mensajes se publican sobre un "tema" en particular. Un suscriptor recibe esos mensajes para los temas a los que se ha suscrito.

El broker utilizado para esta arquitectura es: RabbitMQ

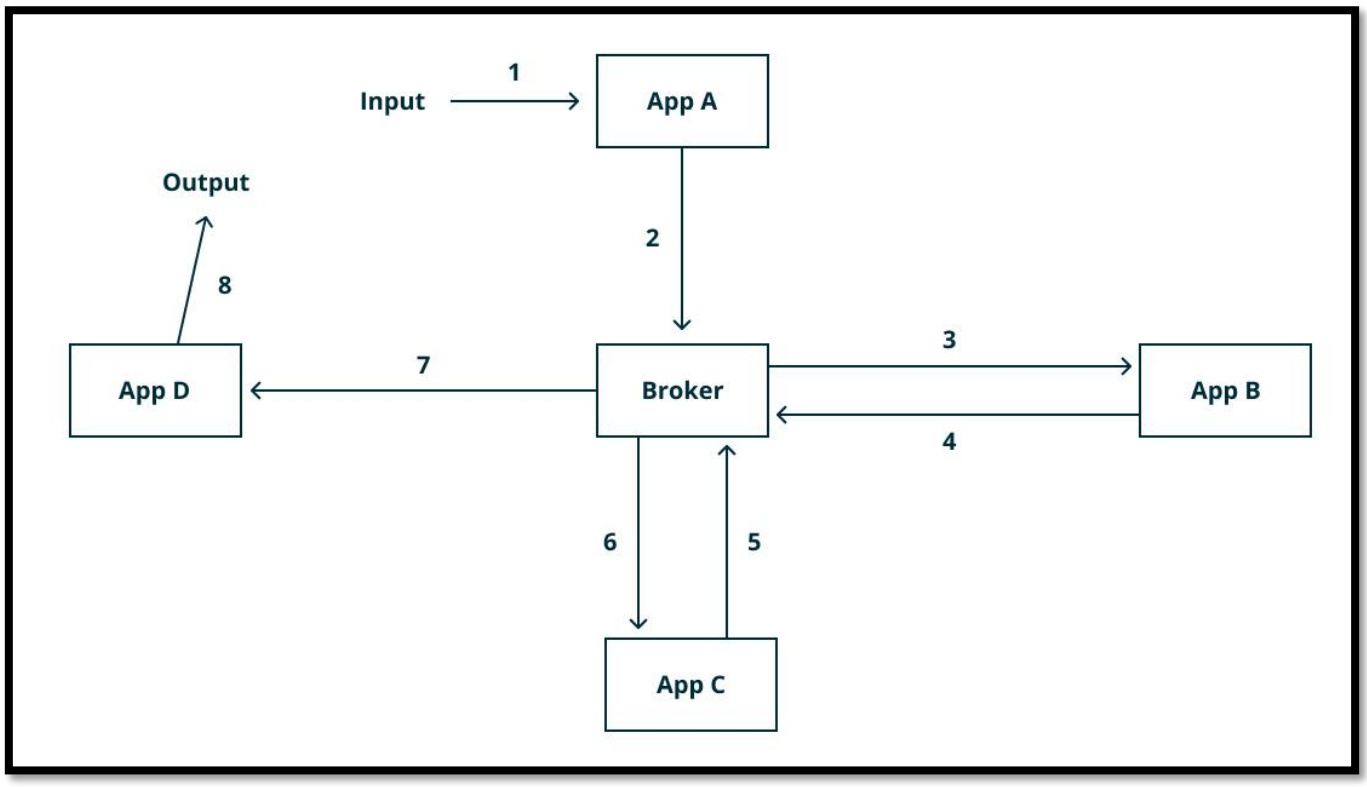


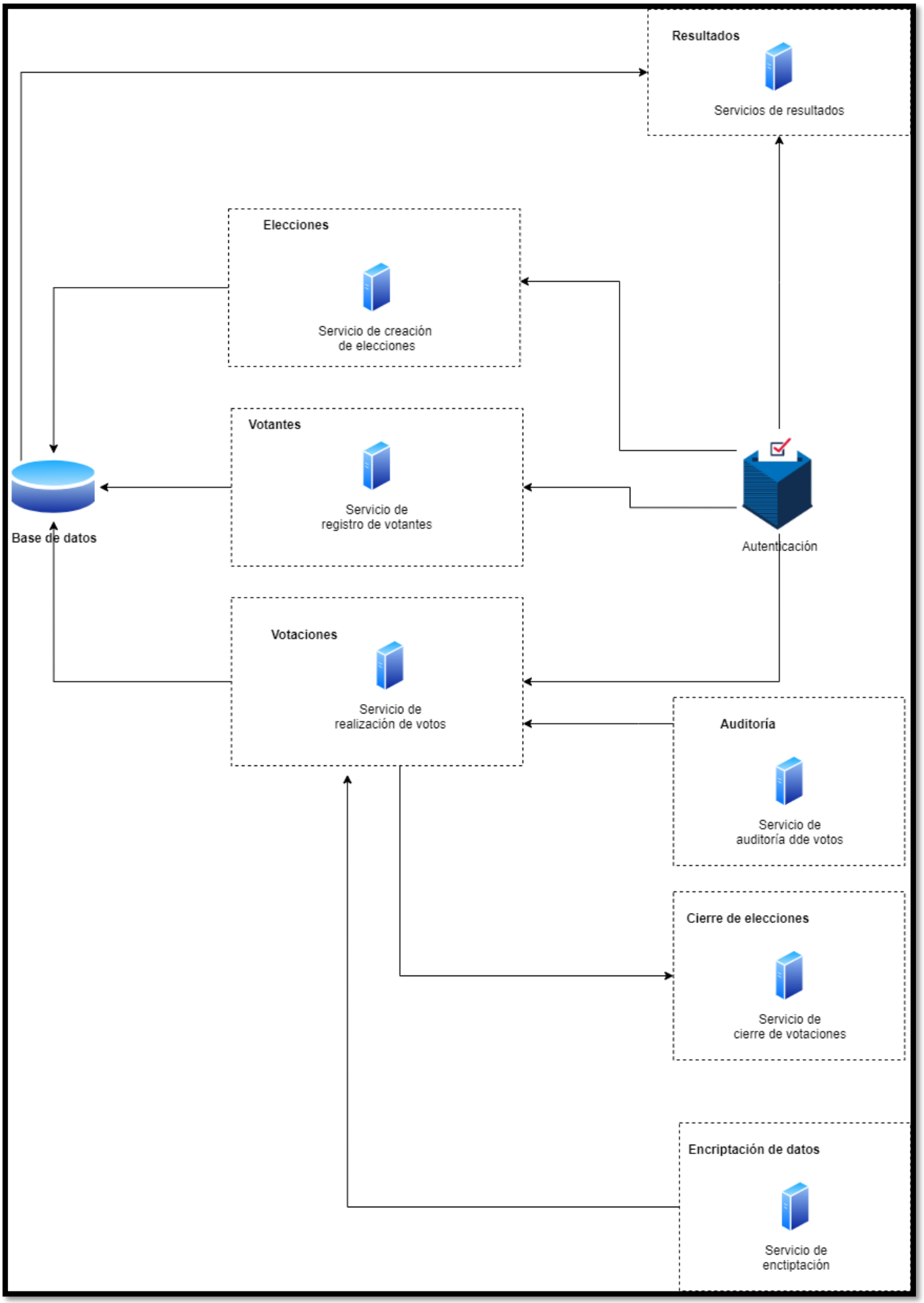
Diagrama de interacción de los microservicios

El sistema iVoting consta de nueve microservicios para realizar las funcionalidades solicitadas. A continuación, se detalla cada uno de ellos

- **Creación de elecciones: Interfaz gráfica que permite la crear y administrar elecciones, recibe toda la información para que la elección pueda ser generada. Las elecciones se almacenan en la base de datos.**
- **Registro de ciudadanos para votar: Permite el ingreso de votantes, solicitando la información completa de la persona. Cada registro es almacenado en la base de datos.**
- **Base de datos: Donde se guardará toda la información utilizada en la aplicación.**
- **Votaciones: Permite la realización de votaciones en las elecciones creadas, solamente pueden acceder las personas que previamente fueron registradas. La información se almacenará en la base de daos.**
- **Resultados: Consulta los resultados almacenados en la base de datos sobre una de las elecciones creadas.**
- **Cierre elecciones: Se comunica con las votaciones ya que le envía al microservicio de Votaciones en qué momento deben cerrar las votaciones para poder realizar el conteo de votos.**
- **Auditoria: Microservicio comunicado con el microservicio de votaciones, es el encargado de verificar cantidad de votos y autenticidad de los mismos.**

- **Encriptación: Se encarga de encriptar los datos de las votaciones, para garantizar que el voto es secreto y nadie puede acceder a la información.**

- **Autenticación: Toda persona que desea participar en una votación, debe ingresar y validar su usuario por medio de un token.**


Resultados
Servicios de resultados
Elecciones
Servicio de creación
de elecciones
Votantes
Servicio de
registro de votantes
Base de datos
Autenticación
Votaciones
Servicio de
realización de votos
Auditoría
Servicio de
auditoría dde votos
Cierre de elecciones
Servicio de
cierre de votaciones
Encriptación de datos
Servicio de
enctiptación

Autenticación

Este microservicio utilizará un modelo Cliente-Token en el cual el token es utilizado para indicar la identidad del usuario, el contenido del token se encontrará encriptado por cuestiones de seguridad.

Se utilizará [JWT (Json Web Token)][JWT] para definir el formato del token, asi mismo como su contenido y estructura de encriptación.
La estructura consiste unicamente de 3 partes: Header, Payload y Signature.

Header

Contiene el tipo, valor fijado como JWT y el algoritmo hash utilizado en JWT.

```
{
   "typ": "JWT",
   "alg": "HS256"
}`
```

Payload

Incluye informacion estandar la cual consiste en: user id, expiration date, y user name

```
{
   "id": 123,
   "name": "Mena Meseha",
   "is_admin": true,
   "expire": 1558213420
}
```

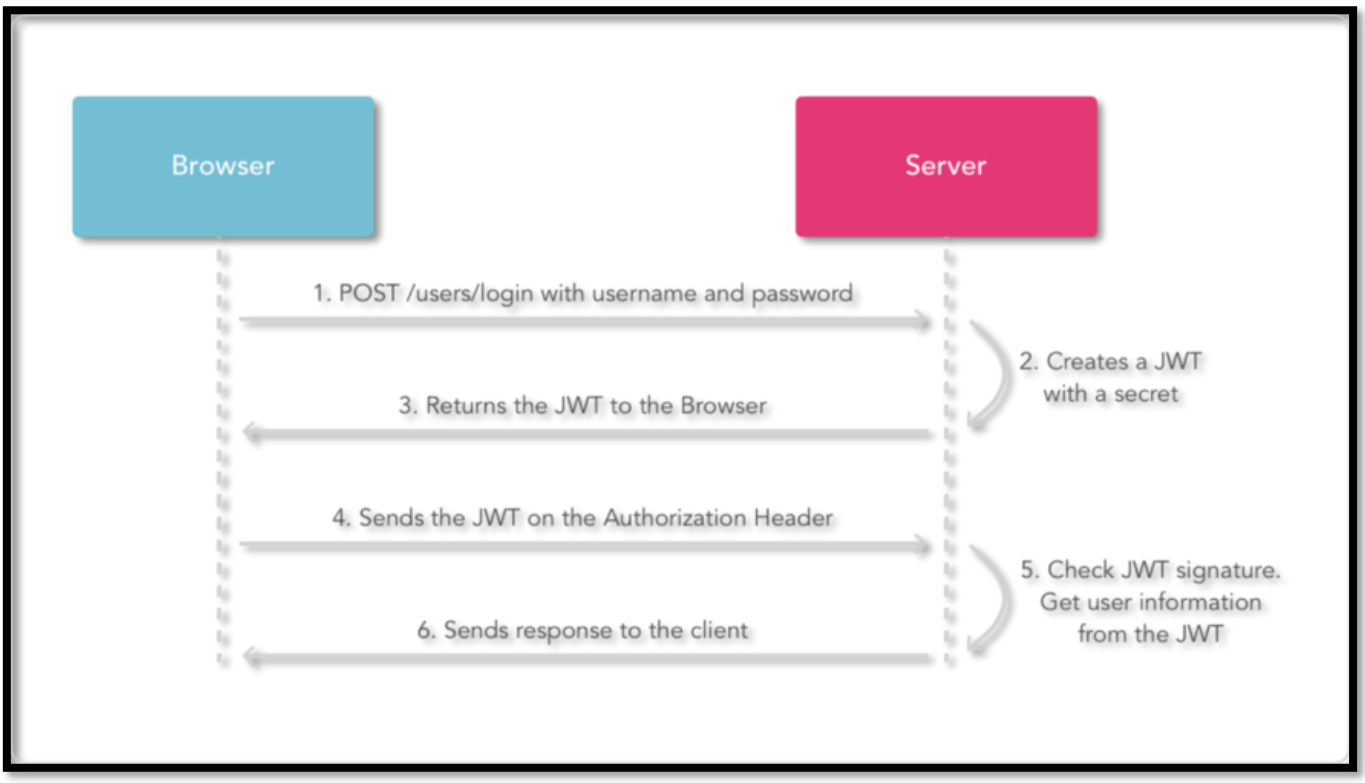
Signature

Este campo es utilizado para el cliente para verificar la identidad del token.

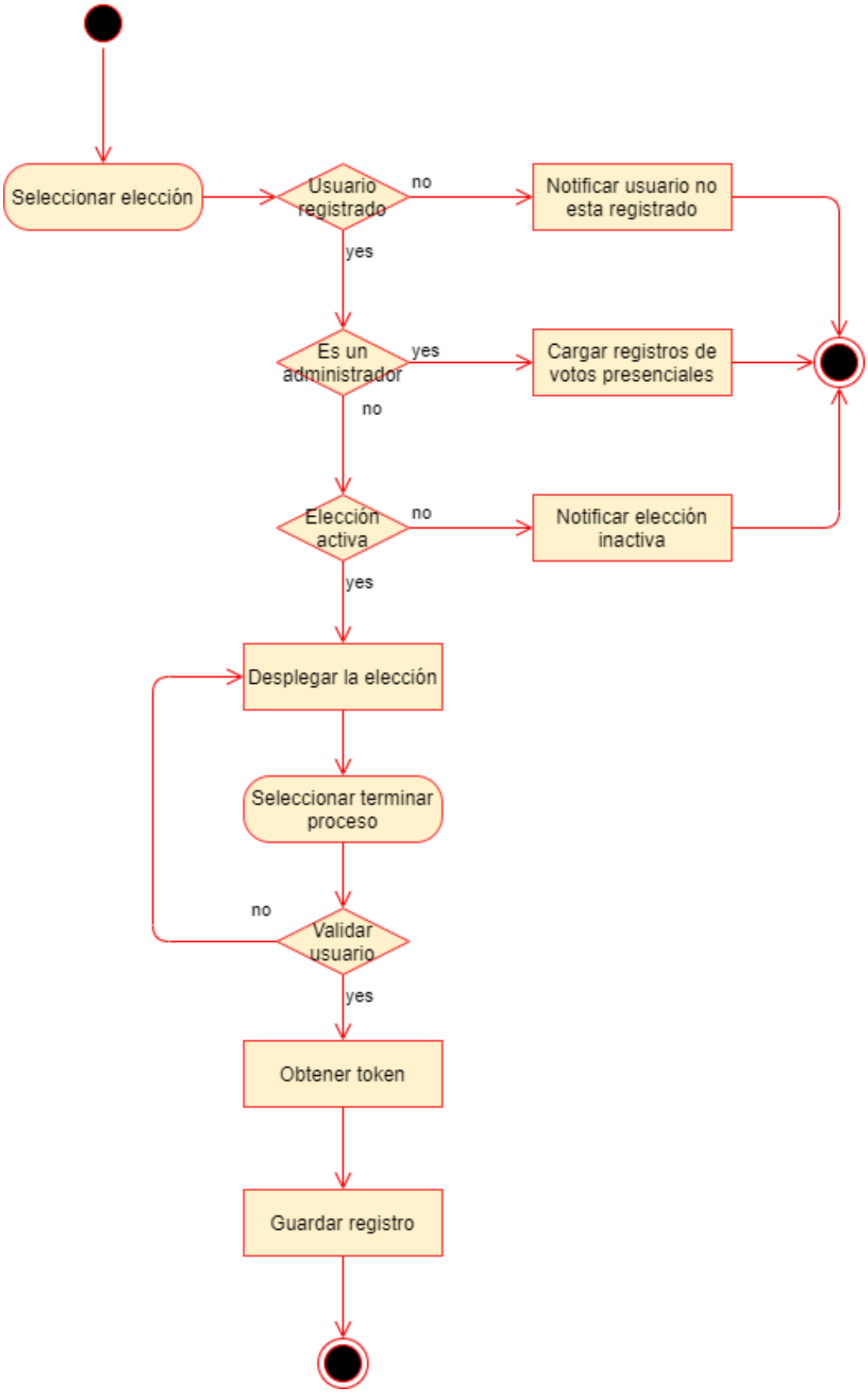
```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret
)
```

Al utilizar esta estructura el resultado final obtenido será parecido a esto:
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpZCI6MTIzLCJuYW1lIjoiTWVuYSBNZXNlaGEiLCJpc19hZG1pbiI6dHJ1ZSwiZXhwaXJlIjoxNTU4MjEzNDIwfQ.Kmy_2WCPbpg-aKQmiLaKFLxb5d3rOC71DHexncH_AcQ

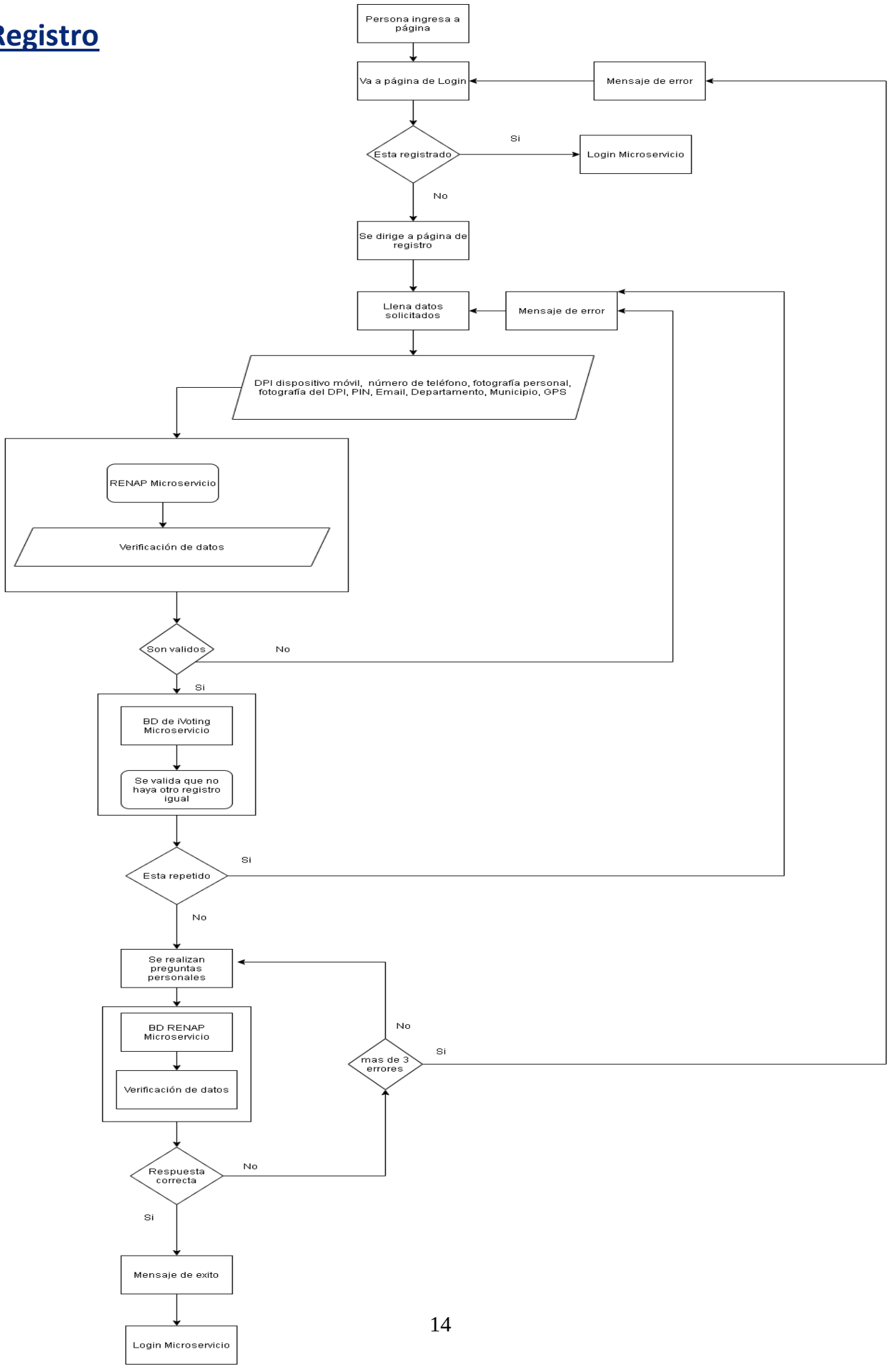
Flujo básico que el microservicio utiliza


Browser
Server
1. POST /users/login with username and password
2. Creates a JWT with a secret
3. Returns the JWT to the Browser
4. Sends the JWT on the Authorization Header
5. Check JWT signature. Get user information from the JWT
6. Sends response to the client

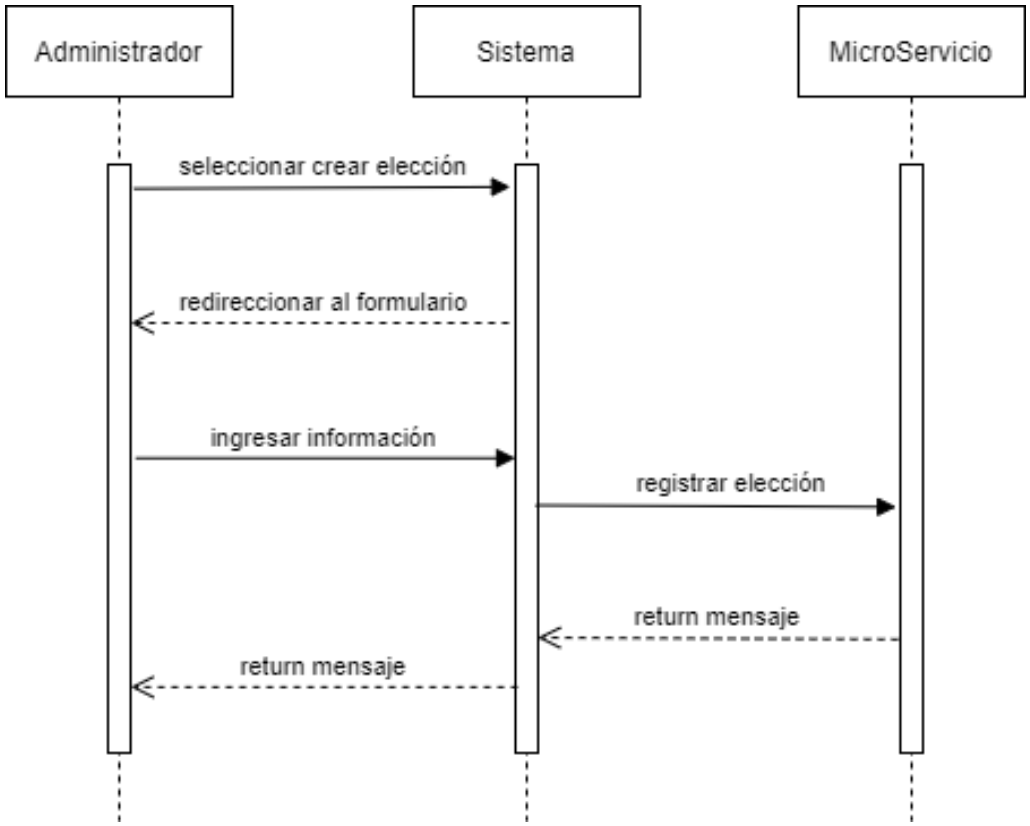
Emisión de voto



Registro


Persona ingresa a página
Va a página de Login
Mensaje de error
Esta registrado
Si
Login Microservicio
No
Se dirige a página de registro
Llena datos solicitados
Mensaje de error
DPI dispositivo móvil, número de teléfono, fotografía personal, fotografía del DPI, PIN, Email, Departamento, Municipio, GPS
RENAP Microservicio
Verificación de datos
Son validos
No
Si
BD de iVoting Microservicio
Se valida que no haya otro registro igual
Esta repetido
Si
No
Se realizan preguntas personales
BD RENAP Microservicio
Verificación de datos
mas de 3 errores
No
Si
Respuesta correcta
No
Si
Mensaje de exito
Login Microservicio

Creación de elección



Contratos

Carga masiva RENAP

Archivo csv

Encabezado	Tipo	Descripción
cui	Cadena	Código único de identificación
nombres	Cadena	Nombre completo del usuario
apellidos	Cadena	Apellidos del usuario
fecha_nacimiento	Cadena	Formato según ISO 8601 YYYY-MM-DD
lugar_municipio	Cadena	Nombre del municipio: ‘Mixco’, ‘Chiantla’, ‘Villa Nueva’
lugar_departamento	Cadena	Nombre de departamento: ‘Petén’, ‘Guatemala’
lugar_pais	Cadena	Nombre de país: ‘Guatemala’, ‘El Salvador’
nacionalidad	Cadena	Nacionalidad del ciudadano: ‘Guatemala’, ‘El Salvador’
sexo	Cadena	Dos posibles valores válidos: • masculino -> ‘M‘ • femenino -> ‘F’
estado_civil	Entero	Valores: • 0 soltero • 1 casado
servicio_militar	Entero	Basados en el art. 15 de LEPP los valores son: • 0 No presta servicio • 1 Presta servicio
privado_libertad	Entero	• 0 libre • 1 privado
foto	Cadena	En base64
padron	Entero	Numero de empadoramiento

1. Base de datos de ciudadanos

ID: SA-000

Historia de usuario:

Como administrador del sistema quiero un servicio REST para poder consultar los datos de los ciudadanos en el RENAP	**Prioridad**	Alto
	Estimado	10
	Módulo	RENAP

Criterio de aceptación:

El servicio debe tener la siguiente configuración:

Ruta:/api/renap/cui
Método: GET
Formato de entrada: JSON

Header:

Atributo	**Tipo**	**Descripción**
Content-type	header	application/json
Authentication	header	token <TOKEN>

Parámetros de entrada:

Atributo	**Tipo**	**Descripción**
ciu	Entero	Código único de identificación

Formato de salida: JSON
Código respuesta exitosa: HTTP 200
Parámetros de salida exitosa:

Atributo	**Tipo**	**Descripción**
cui	Cadena	Código único de identificación
nombres	Cadena	Nombre completo del usuario
apellidos	Cadena	Apellidos del usuario
fecha_nacimiento	Cadena	Formato según ISO 8601 YYYY-MM-DD
lugar_municipio	Cadena	Nombre del municipio: ‘Mixco’, ‘Chiantla’, ‘Villa Nueva’

lugar_departamento	Cadena	Nombre de departamento: ‘Petén’, ‘Guatemala’
lugar_pais	Cadena	Nombre de país: ‘Guatemala’, ‘El Salvador’
nacionalidad	Cadena	Nacionalidad del ciudadano: ‘Guatemala’, ‘El Salvador’
sexo	Cadena	Dos posibles valores válidos: • masculino -> ‘M‘ • femenino -> ‘F’
estado_civil	Entero	Valores: • 0 soltero • 1 casado
servicio_militar	Entero	Basados en el art. 15 de LEPP los valores son: • 0 No presta servicio • 1 Presta servicio
privado_libertad	Entero	• 0 libre • 1 privado
foto	Cadena	En base64

Código de respuesta fallida:

Se utilizara la siguiente lista de errrores como referencia: https://docs.microsoft.com/es-es/partner/develop/error-codes

Código	**Descripción**
Número de error	Descripción del error

Parámetros de salida fallida:

Atributo	**Tipo**	**Descripción**
status	Número	Indica el tipo de error ocurrido.
mensaje	Cadena	Muestra un detalle del error ocurrido.

Ejemplo de parámetros de entrada:

```
{
        "cui":
}
```

Ejemplo de parámetros de salida exitosa:

Ej 1:

```
{
        "status": 200,
        "cui":"",
        "nombres":"",
```

```
            "apellidos":"",
            "fecha_nacimiento":"",
            "lugar_municipio": "",
            "lugar_departamento":"",
            "lugar_pais":"",
             "nacionalidad":"",
             "sexo":"",
             "estado_civil":"",
             "servicio_militar":"",
             "privado_libertad":"",
              "foto":"",
               “padron”:
}

Ej 2:

{
            "status": 200,
            "cui":"1111222223333",
            "nombres":"Sandra",
            "apellidos":"Solis",
            "fecha_nacimiento":"1994-11-01",
            "lugar_municipio": "Mixco",
            "lugar_departamento":"Guatemala",
            "lugar_pais":"Guatemala",
             "nacionalidad":"Guatemala",
             "sexo": "F",
             "estado_civil": 0,
             "servicio_militar": 0,
             "privado_libertad": 0,
              "foto":"aG9sYS5qcGc="
}
```


Ejemplo de parámetros de salida fallida:


```
{
            "status": 400,
            "mensaje" :
}
```


2. Autenticación

ID: SA-001		
Historia de usuario:		
Como administrador del sistema Quiero un servicio REST Para poder acceder a los usuarios en el sistema	**Prioridad**	Alta
	Estimado	10

	Módulo	Acceso

Criterio de aceptación:

El servicio se debe conectar a un servidor JWT y retornar un token válido para autenticar al usuario y retornar los datos del usuario.

El servicio debe tener la siguiente configuración:

Ruta:/api/autenticacion
Método: POST
Formato de entrada: JSON
Header:

Atributo	**Tipo**	**Descripción**
Content-type	header	application/json
Authentication	header	token <TOKEN>

Parámetros de entrada:

Atributo	**Tipo**	**Descripción**
email	Cadena	Correo electrónico del usuario.
pass	Entero	Código de 6 dígitos del usuario.*
rol	Entero	El rol puede ser: 1 - administrador 2 - ciudadano

Formato de salida: JSON
Código respuesta exitosa: HTTP 200
Parámetros de salida exitosa:

Atributo	**Tipo**	**Descripción**
token	Cadena	Token de autenticación.

Código de respuesta fallida:
Se utilizara la siguiente lista de errrores como referencia: https://docs.microsoft.com/es-es/partner/develop/error-codes

Código	**Descripción**
Número de error	Descripción del error

Parámetros de salida fallida:

Atributo	Tipo	Descripción
status	Número	Indica el tipo de error ocurrido.
mensaje	Cadena	Muestra un detalle del error ocurrido.

Ejemplo de parámetros de entrada:

```
{
        "email":,
        "pass":
}
```

Ejemplo de parámetros de salida exitosa:

```
{
        "status": 200,
        "token" :
}
```

Ejemplo de parámetros de salida fallida:

```
{
        "status": 400,
        "mensaje" :
}
```

*El pass es el pin que menciona el enunciado.

3. Resultados

ID: SA-003		
Historia de usuario:		
Como usuario del sistema Quiero un servicio REST Para poder visualizar el resultado de las votaciones de una elección enviando un identificador de la elección.	**Prioridad**	Alta
	Estimado	10
	Módulo	Resultados
Criterio de aceptación:		
El sistema debe validar que la elección esté activa y garantizar el voto secreto para los usuarios. El servicio debe tener la siguiente configuración: **Ruta**:/api/resultado/:ideleccion **Método**: GET		

Formato de entrada: explícito en la ruta.

Formato de salida: JSON
Código respuesta exitosa: HTTP 200

Header:

Atributo	**Tipo**	**Descripción**
Content-type	header	application/json
Authentication	header	token <TOKEN>

Parámetros de entrada:

Atributo	**Tipo**	**Descripción**
id_eleccion	Entero	Número de identificador de la elección

Formato de salida: JSON
Código respuesta exitosa: HTTP 200
Parámetros de salida exitosa:

Atributo	**Tipo**	**Descripción**
token	Cadena	Token de autenticación.
titulo	Cadena	Nombre de la elección
fecha_inicio	Cadena	Fecha de inicio de votación
fecha_fin	Cadena	Fecha de finalización de votación
votantes_registrados	Número	Cantidad de votantes que se registraron y se espera que voten.
votos_totales	Número	Cantidad de votos de personas que votaron.
datos_votacion	Arreglo	Arreglo de votaciones por partido y ubicación. Ver ejemplo adjunto.

Código de respuesta fallida:

Se utilizara la siguiente lista de errrores como referencia: https://docs.microsoft.com/es-es/partner/develop/error-codes

Código	**Descripción**
Número de error	Descripción del error

Parámetros de salida fallida:

Atributo	Tipo	Descripción
status	Número	Indica el tipo de error ocurrido.
mensaje	Cadena	Muestra un detalle del error ocurrido.

Ejemplo de parámetros de entrada:

```
{
        "id_eleccion":
}
```

Ejemplo de parámetros de salida exitosa:

```
{

        “status”:200,
       "titulo":”TituloEleccion”,
        "fecha_inicio":”DD/MM/YYYY”,
        "fecha_fin":”DD/MM/YYYY”,
        "votantes_registrados":127,
        "votos_totales":#####,
        "cargo":#,
        "datos_votacion":[
                "nulos":{
                        "postulante": "nulo",
                        "votos_recibidos":42,
                        "ubicacion":[
                        {
                                "pais": "",
                                "municipio": "Mixco",
                                "departamento": "Guatemala",
                                "cantidad_votos": 40
                        },
                        {
                                "pais": "",
                                "municipio": "Antigua Guatemala",
                                "departamento": "Sacatepequez",
                                "cantidad_votos":2
                        }
                        ]
                },
                "PAN":{
                        "postulante": { “presidente”:“juan paco”,”vicepresidente”:”pedro delamar” },
                        "votos_recibidos":42,
                        "ubicacion":[
                        {
                                "pais": "",
                                "municipio": "Mixco",
                                "departamento": "Guatemala",
                                "cantidad_votos": 40
                        },
                        {
                                "pais": "",
                                "municipio": "Antigua Guatemala",
                                "departamento": "Sacatepequez",
```

```
                                "cantidad_votos":2
                        }
                        ]
                },
                "UNE":{
                        "postulante": { “presidente”: "Pedro", vicepresidente: “Manuel Carias”},
                        "votos_recibidos": 45,
                        "ubicacion":[
                        {
                                "pais": "",
                                "municipio": "Mixco",
                                "departamento": "Guatemala",
                                "cantidad_votos": 40
                        },
                        {
                                "pais": "",
                                "municipio": "Antigua Guatemala",
                                "departamento": "Sacatepequez",
                                "cantidad_votos":2
                        },
                        {
                                "pais": "USA",
                                "municipio": "",
                                "departamento": "",
                                "cantidad_votos":3
                        }
                        ]
                }
        ]

}
```


Ejemplo de parámetros de salida fallida:


```
{
        "status": 400,
        "mensaje"  “Bad request"":
}
```

Conclusiones

- **Los microservicios mas importantes ya que estos deben de tener la posibilidad de ser usados en otro sistema son: base de datos, resultados y autenticación.**
- **La tecnología para hacer contenedores de los microservicios, excepto de la base de datos es Docker.**
- **El formato que se usara en el microservicio de Encriptación es JWT.**
- **El bróker para enviar y recibir mensajes en la comunicación de los microservicios es RabbitMQ.**