MANUAL DE USUARIO

"Sistema Básico de Expediente Electrónico - SBEE - Digitalia F2"

Documentación técnica orientada a desarrolladores

Tabla de contenido

INTRODUCCIÓN 3

DEFINICIONES, ACRÓNIMOS Y ABREVIATURAS 3

DESCRIPCIÓN DEL SISTEMA 3

PÚBLICO OBJETIVO 3

INICIO DE SESIÓN CON IDENTIDAD ELECTRÓNICA 4

RESTRICCIÓN DE DOMINIO EN LAS PETICIONES 5

Introducción

El presente documento técnico está diseñado para orientar a los desarrolladores, indicando los ajustes y consideraciones realizadas al proyecto SBEE, de manera que puedan ser replicados en otros proyectos de similares características.

Definiciones, Acrónimos y Abreviaturas

RF: Requerimiento funcional
RNF: Requerimiento no funcional
OEE: Organismos y Entidades del Estado.
UO: Unidad Organizativa.
SII: Sistema de Intercambio de Información, plataforma de interoperabilidad, que actúa como canal entre las instituciones y organizaciones del Estado para compartir información.
MITIC: Ministerio de Tecnologías de la Información y Comunicación.
Usuarios OEE: Funcionarios públicos que poseen credenciales dentro del servicio de SBEE, y según sus permisos, pueden generar y derivar expedientes y actuaciones.
GDL: Gestión de Documentos en Línea, plataforma que posibilita la gestión y obtención en línea de documentos requeridos por instituciones públicas para distintos trámites y gestiones.
SBEE: Sistema básico de expediente electrónico, permite la gestión de expediente desde su origen, derivación y cierre.
API: Interfaz de Programación de Aplicaciones, es un conjunto de funciones y procedimientos que permite integrar sistemas, permitiendo que sus funcionalidades puedan ser reutilizadas por otras aplicaciones o software.
UAPI: Usuario registrado en el sistema con el rol API

Descripción del Sistema SBEE

Capa de Servicios (Backend)

La Capa de Servicios fue construida mediante el desarrollo de servicios web del tipo REST (Representational State Transfer), en la cual los datos son intercambiados utilizando el formato de texto JSON (JavaScript Object Notation). Se encuentra sobre la tecnología Spring Boot 2.6.0 framework del lenguaje de programación java.

Abarca los servicios preparados para integrarse con la capa de aplicaciones, generación de reportes, las comunicaciones con las bases de datos involucradas, entre otros.

Capa de Aplicaciones (Frontend)

La Capa de Aplicaciones fue construida con el framework Angular 13.4.0 basado en el lenguaje JavaScript para aplicaciones web bastante potente que permite generar y reutilizar componentes web. Se utiliza el template Ultima PrimeNG from Primefaces.

Público objetivo

Los usuarios desarrolladores, integradores, u otros técnicos con conocimientos y habilidades necesarias para integrar aplicaciones o servicios con proyectos de similares características al SBEE.

Inicio de sesión con Identidad Electrónica

Nuevo y único método de inicio de sesión el login con la API de Identidad Electrónica.

Ajustes y consideraciones necesarios en el Frontend

En la pantalla de autenticación de usuarios se reemplazó la autenticación con usuario y contraseña por la opción de autenticación con Identidad Electrónica por medio del botón que se visualiza en la imagen.

Si la autenticación en identidad electrónica es exitosa se redirige al portal de SBEE con un token de autenticación el cual con una petición POST se envía al backend por medio del endpoint "/auth/mitic-login" para realizar la autenticación propia del sistema.

Ajustes y consideraciones necesarios en el Backend

Para la integración de la identidad electrónica se crearon dos endpoints:

```
@GetMapping(value = "/datos-ie", produces = MediaType.APPLICATION_JSON)
public ResponseEntity<IEDataResponse> getIEData() {
  return ResponseEntity.ok(new IEDataResponse(miticClientId, miticUrl,
tokenComponent.getStateForIe()));
}
```

Endpoint GET el cual nos permite obtener datos de configuración necesarias para que el frontend pueda redireccionar al portal de Identidad Electrónica.

```
@PostMapping(value = "/mitic-login", consumes = MediaType.APPLICATION_JSON, produces
= MediaType.APPLICATION_JSON)
 public ResponseEntity<?> miticLogin(@RequestBody AuthMiticRequest payload) throws
Exception, IOException {
  try{
    if(!tokenComponent.validateMiticSignature(payload.getToken()))
      throw new Exception("No se pudo verificar firma - Código Autorización. JWT: " +
payload.getToken());
    if(!tokenComponent.validateMiticToken(payload.getToken(), payload.getState()))
      throw new Exception("Error al validar Código Autorización, token expirado o state
invalido. JWT: " + payload.getToken() + ", State: " + payload.getState());
    String accessToken = tokenComponent.obtenerTokenMitic(payload.getToken());
    if(accessToken == null)
      throw new Exception("No se pudo obtener el Token de acceso");
    if(!tokenComponent.validateMiticSignature(accessToken))
      throw new Exception("No se pudo verificar firma - Token de Acceso. Token: " +
accessToken);
    JsonNode accessNode = tokenComponent.validateMiticAccessToken(accessToken);
    if(accessNode == null)
      throw new Exception("Error al validar Token de Acceso, token expirado o state invalido.
JWT: " + payload.getToken());
    Usuario usuario = usuarioService.getUserByCiActivo(accessNode.get("sub").textValue());
    if(usuario == null)
      return new ResponseDTO("El usuario no cuenta con permiso suficiente para acceder al
sistema.", Statics.ALERTA_ERROR, HttpStatus.OK).construirLogin();
    if (usuario.getEstadoUsuario().equals(Statics.TIPO_ESTADO_INACTIVO))
      return new ResponseDTO("El usuario se encuentra inactivo", Statics.ALERTA_ERROR,
HttpStatus.OK).construirLogin();
    if(usuario.getInstitucion().getEstadoInstitucion().equals(Statics.TIPO_ESTADO_INACTIVO))
      return new ResponseDTO("La institución del usuario se encuentra inactiva",
Statics.ALERTA_ERROR, HttpStatus.OK).construirLogin();
    if(usuario.getRoles().size() == 0)
```

```
      return new ResponseDTO("El usuario no cuenta con permiso suficiente para acceder al sistema", Statics.ALERTA_ERROR, HttpStatus.OK).construirLogin();
    return ResponseEntity.ok(doOwnLogin(usuario));
  }catch (IncorrectResultSizeDataAccessException ex){
    logger.info(ex.getMessage());
    ex.printStackTrace();
    return new ResponseDTO("Se encontró más de un usuario activo, favor contacté con el mitic.", Statics.ALERTA_ERROR, HttpStatus.INTERNAL_SERVER_ERROR).construirLogin();
  }catch (Exception ex){
    logger.info(ex.getMessage());
    ex.printStackTrace();
    return new ResponseDTO("Error al Intentar loguear", Statics.ALERTA_ERROR, HttpStatus.INTERNAL_SERVER_ERROR).construirLogin();
  }
 }
```

Endpoint POST el cual recibe como datos el token de autorización y "state" el cual representa un código generado en el frontend asociado a la transacción actual.

El endpoint realiza una serie de validaciones en el siguiente orden:

- *Verificación de la firma del token con el "client secret" que está configurado en el backend.*
- *Verificación de la expiración del token (5 minutos desde su creación) y del claim "state".*

Si pasa las validaciones anteriores se procede a obtener el token de acceso por medio de una petición POST.

- *El token de acceso pasa por las mismas validaciones que el token de autorización citadas anteriormente.*
- *Al pasar todas las validaciones citadas se obtiene el token decodificado el cual contiene información del usuario asociado al token y se procede con las validaciones propias del sistema.*
- *Validación de existencia del usuario por medio del número cédula de identidad.*
- *Validación de usuario activo.*
- *Validación de institución del usuario activo.*
- *Validación de roles.*

Si pasa todas las validaciones se retorna el código 200 con el token de acceso e información del usuario.

En caso de que se encuentre más de un usuario activo para la cédula de identidad se retorna el mensaje de error " Se encontró más de un usuario activo, favor contacté con el mitic."
Para cualquier otro error se registra en los logs el motivo y se retorna el mensaje "Error al Intentar loguear"

Archivo de configuración

*El archivo **application.properties** debe contar con lo siguiente configuracion:*

```
## IE
mitic.auth.secret=1234
mitic.auth.url=devidentidad.mitic.gov.py
mitic.auth.client.id=43fbd71a-92bb-4e39-9104-abcdefghi123
mitic.auth.rest=/rest/authentication
```

Donde:

- *mitic.auth.secret: parámetro secret indicado por el MITIC utilizado para firmar el token.*
- *mitic.auth.url: servidor de autenticación indicado por el MITIC*
- *mitic.auth.client.id: id de cliente indicado por el MITIC el cual utiliza para identificar a la aplicación.*
- *mitic.auth.rest: endpoint utilizado para obtener el token de acceso.*

Casos de prueba

Observación: las url base y callback deben ser configuradas por el MITIC para poder llevar a cabo el flujo completo el ambiente correspondiente.

Pasos:

i. *El usuario ingresa a la opción "Ingresá con tu Identidad Electrónica".*
ii. *El sistema redirige al usuario al portal de Identidad Electrónica donde le solicita sus credenciales.*

iii. *Una vez que el portal de Identidad Electrónica haya verificado las credenciales, redirige al usuario al portal de SBEE con los datos del usuario.*
iv. *El sistema SBEE verifica a través de la cédula del usuario si el mismo posee rol, si está activo y vigente.*

v. *Si el usuario está verificado, el sistema SBEE muestra el Dashboard principal, caso contrario, emitirá un mensaje de error en el cual indique que el mismo no cuenta con permiso de acceso y redirige al inicio de sesión.*

En caso de error se emitirá uno de los siguientes mensajes:

- *" Se encontró más de un usuario activo, favor contacté con el mitic."*
- *"Error al Intentar loguear"*

Restricción de dominio en las peticiones

El sistema sólo deberá responder a dominios establecidos en las configuraciones.

Ajustes y consideraciones necesarios en el Frontend

No es necesario ningún ajuste ni consideración en el FrontEnd

Ajustes y consideraciones necesarios en el Backend

Agregar el método corsFilter() en el archivo SecurityConfig.java, o en su defecto el archivo con el annotation @Configuration.

El archivo quedaría de la siguiente manera:

```
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class SecurityConfig extends WebSecurityConfigurerAdapter {

…

        @Value("${app.dominios.habilitados}")
        private String dominiosHabilitados;

…

        @Bean
        public CorsFilter corsFilter() {
                List<String> dominios = new ArrayList<String>();
                if (dominiosHabilitados != null && dominiosHabilitados.contains(",")) {
                        dominios = Arrays.asList(dominiosHabilitados.trim().split("\\s*,\\s*"));
                } else {
                        dominios.add(dominiosHabilitados.trim());
                }

                final UrlBasedCorsConfigurationSource source = new
UrlBasedCorsConfigurationSource();
```

```
                final CorsConfiguration config = new CorsConfiguration();
                config.setAllowCredentials(true);
                config.setAllowedOrigins(dominios);
                config.setAllowedHeaders(Arrays.asList("Origin", "Content-Type", "Accept"));
                config.setAllowedMethods(Arrays.asList("GET", "POST", "PUT", "OPTIONS",
"DELETE", "PATCH"));
                source.registerCorsConfiguration("/**", config);
                return new CorsFilter(source);
        }

…

}
```

Donde:

- *La variable **dominiosHabilitados** obtiene la lista de dominios habilitados a partir del archivo de configuración: **application.properties***
- *Se instancia la configuración de **CORS** en la variable **config**.*
- *Se crea una lista de dominios a partir de la variable **dominiosHabilitados**, y se asigna a la variable **config** como **allowed Origins** (orígenes permitidos).*
- *Se asignan las cabeceras y métodos que serán permitidos por la configuración.*
- *Se registra la nueva configuración de **CORS**.*

Archivo de configuración

*El archivo **application.properties** debe contar con lo siguiente configuracion:*

```
## DOMINIOS HABILITADOS
app.dominios.habilitados=http://localhost:4201,http://localhost:4202
```

*Donde el parámetro **app.dominios.habilitados** contendrá la lista de dominios, separados por coma, a los que el backed permitirá realizar solicitudes.*

Casos de prueba

*El FrontEnd, por cada petición envía en su cabecera **Origin** el dominio de origen. Con esa información, es posible preparar una petición con cURL, agregando la cabecera **Origin**, para simular que proviene de un dominio particular.*

```
curl --location '{{dominio_backend}}/api/auth/login' \
--header 'Content-Type: application/json' \
--data-raw '{
  "username": "usuario",
  "password": "pass"
}' -H "Origin: {{dominio_frontend}}" -H "Access-Control-Request-Method: GET"
```

Caso de dominio no permitido:

```
curl --location 'http://localhost:8081/api/auth/login' --header 'Content-Type:
application/json' --data-raw '{
  "username": "usuario",
  "password": "pass"
}' -H "Origin: http://localhost:3000" -H "Access-Control-Request-Method: GET"
```

Invalid CORS request

Caso de dominio permitido:

```
 curl --location 'http://localhost:8081/api/auth/login' --header 'Content-Type:
application/json' --data-raw '{
  "username": "usuario",
  "password": "pass"
}' -H "Origin: http://localhost:4201" -H "Access-Control-Request-Method: GET"
```

***{"accessToken":"eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJmbWFuY2lhQG1pdGljLmdvdi5weSIsIm
V4cCI6MTcxODE1MTYxMCwiaWF0IjoxNzE4MTM3MjEwfQ.0HZwcE-zvSNGTfaDJ_wB6il6Rnf
eI7Us8N4XN6pyBEVNzZB1RMEkdFNVGALA0ufDykOYnWRC0zKvw5iv47qKzg"}***

JULIO MANUEL PACIELLO CORONEL

Digitally signed by JULIO MANUEL PACIELLO CORONEL
DN: CN=JULIO MANUEL PACIELLO CORONEL, SERIALNUMBER=CI1490175, G=JULIO MANUEL, SN=PACIELLO CORONEL, OU=F2, O=CERTIFICADO CUALIFICADO DE FIRMA ELECTRONICA, C=PY
Reason: I am the author of this document
Location:
Date: 2024.06.17 17:32:50-04'00'
Foxit PDF Editor Version: 13.0.1

Ing. Julio Paciello Coronel
Coordinador Especialista TIC
Centro de Desarrollo Sostenible SA