everis

an NTT DATA Company

DISEÑO TÉCNICO

Tótems - Porvenir

TABLA DE CONTENIDO

HISTÓRICO DEL DOCUMENTO

Versión	Fecha	Autor	Empresa
1.0	07/10/2019	Daniel Felipe Tafur Posada	everis
1.1	19/10/2019	Oscar Carantón Sánchez	everis

OBJETIVO

Este documento tiene como finalidad explicar a nivel técnico la manera en la que se desarrollaron los servicios Middleware de “Punto Digital”, la arquitectura implementada, los patrones de diseño utilizado y las tecnologías que se aplicaron.

ARQUITECTURA

1. Introducción

Los servicios “Middleware” tienen como finalidad la comunicación del aplicativo web (enfocado a la plataforma Windows) con los periféricos (Impresora, Escáner, Micrófono, Cámara, Lector de huellas), la creación de Logs (Historial de registro), y él envió de alertas a correo electrónico desde un servicio.

2. Descripción General

Se implementó la conectividad de los servicios middleware y oauth con la finalidad de generar seguridad, y así mismo se genera la conectividad con eureka para generar el registro de los puntos digitales.

Los servicios Middleware se encuentran alojados de manera local en cada uno de los puntos digitales, mientras los servicios Oauth y Eureka se encuentran en el servidor de Porvenir.

3. Patrón de desarrollo

El patrón de desarrollo utilizado fue el MVC (Modelo – Vista – Controlador). Es un diseño de arquitectura de software que se fundamente en la separación del código en tres capas diferentes, acotadas por su responsabilidad, ayudándonos así a crear desarrollos de mayor calidad.

- Modelo

  Es la capa en donde se trabaja con los datos, por lo tanto contendrá mecanismos para acceder a la información y también para actualizar su estado, habitualmente la datos se encuentran almacenados en una base de datos aunque no ocurre en este caso.

- Vista

  Es la capa que contiene el código del desarrollo que va a producir la visualización de interfaces de usuario, es decir es el código que nos permitirá renderizar al usuario final las pantallas, ventanas, páginas y formularios

- Controlador

  Es la capa que se encarga de gestionar las instrucciones que se reciben por parte del usuario, atenderlas y procesarlas. Por medio de esta capa se comunican el modelo y la vista de manera que solicita datos, manipula datos para obtener resultados y los entrega a la vista para que pueda mostrarlos

4. Métodos HTTP

Son un conjunto de métodos de petición para indicar la acción que se desea realizar para un curso determinado, cada uno de ellos implementan características diferentes, permiten comunicar al servidor lo que se quiere realizar con un recurso bajo una URL.

- GET

  Este método se utiliza cuando se necesita adquirir un archivo o recurso que se encuentra en un servidor web, este método retorna tanto las cabeceras que contienen los metadatos del recurso solicitado y el recurso en sí.

- POST

  Este método se utiliza cuando se necesita enviar información o un elemento al servidor y que lo enviado sea almacenado como un “hijo” o subelemento de un elemento o recurso ya existente en el servidor. Este método se usa para enviar información a un recurso web del servidor mas no para cargar/crear un elemento nuevo como tal.

- HEAD

  Este método realiza una acción similar al método GET, solo que a diferencia este método solo solicita los metadatos de un recurso o archivo y no todo el elemento como tal.

- PUT

  Este método crea/carga un nuevo recurso al servidor, o en caso de que el objeto ya exista en el servidor, se reemplaza el recurso existente con el recurso que se carga.

- DELETE

  Este método le solicita al servidor web que se borre un recurso en específico.

TECNOLOGIAS

- Java:

  Es un lenguaje de programación orientado a objetos con la intención de que los programadores escribieran el código solo una vez y lo ejecutaran en cualquier dispositivo, siendo esto posible gracias a que Java cuenta con JVM o Java Virtual Machine que brinda portabilidad al lenguaje, ya que hoy existen JMVs para diferentes arquitecturas para todas las plataformas.

- Spring Boot:

  Es una marco de codigo abierto basado en Java que se utiliza para el desarrollo de una aplicación junto con micro servicios.

SEGURIDAD

A continuación, se realiza la descripción de los métodos utilizados para la seguridad de los servicios Middleware.

- **Oauth**

  Es un estándar abierto que permite flujos simples de autorización para sitios web o aplicaciones informáticas, se trata de un protocolo que permite autorización segura de una API en modo estándar y simple para aplicaciones de escritorio, móviles y web. En los servicios middleware el Oauth funciona como un servicio que con unas credenciales específicas retornara un "token" (código) dinámico que se validara para aquellos servicios que dependan de aquella seguridad.

Invocación:

http://localhost:8081/auth/oauth/token

AUTORIZACIÓN	
USUARIO	totem
CONTRASEÑA	PORDIGITAL2020

BODY	
grant_type	client_credentials

Respuesta	**Parámetro**	**Código**
Correcta	status	200
	message	access_token

Respuesta	Parámetro	Código
Incorrecta	status	400
	message	Invalid_request

TRANSACCIÓN

- **Middleware**

  Está compuesto por un conjunto de servicios que asisten al desarrollo Front con la conexión a los periféricos (Impresora, Lector de huellas, Micrófono, Cámara, Escáner), la creación de Logs y él envió de alertas por correo electrónico.

  - Impresora:

    Invocación:

    https://localhost:8082/middleware/printer/printPDF

    Podrá realizar la impresión de un archivo en base 64.
    Servicio de tipo **Post**

Headers	
Authorization	Bearer (token)

Body	
file	(File in Base64)

Respuesta	Parámetro	Código
Correcta	status	200
	message	Impresión finalizada.

Respuesta	Parámetro	Código
Incorrecta	status	202
	message	Error durante la impresión.

- Escáner:

Invocación:

https://localhost:8082/middleware/DNIReader/scan

Podrá escanear documentos como cedulas, tarjetas de identidad y pasaportes.
Servicio de tipo **GET**

Headers	
Authorization	Bearer (token)

Respuesta	Parámetro	Código
Correcta	status	200
	message	Escaneo completado con éxito

Respuesta	Parámetro	Código
Incorrecta	status	201
	message	Error al completar el escaneo

Respuesta	Parámetro	Código

Incorrecta	status	202
	message	Documento invalido

- Huellero:

Invocación:

https://localhost:8082/middleware/fingerprint/active

Podrá activar el huellero para el siguiente paso.
Servicio de tipo **GET**

Headers	
Authorization	Bearer (token)

Respuesta	**Parámetro**	**Código**
Correcta	status	200
	message	Huellero activado.

Respuesta	**Parámetro**	**Código**
Incorrecta	status	201
	message	Error al activar el Huellero.

Invocación:

https://localhost:8082/middleware/fingerprint/scan

Podrá escanear una huella luego de haber activado el huellero.
Servicio de tipo **GET**

Headers	
Authorization	Bearer (token)

Respuesta	**Parámetro**	**Código**
Correcta	status	200
	message	Huellero escaneada con éxito.

Respuesta	**Parámetro**	**Código**
Incorrecta	status	201
	message	Error al escáner al huella.

Respuesta	**Parámetro**	**Código**
Incorrecta	status	202
	message	Huella invalida.

- Estado dispositivos:

Invocación:

https://localhost:8082/middleware/devices/status

Podrá conocer si se encuentran conectados los dispositivos y el estado de insumos de la impresora.
Servicio de tipo **GET**

Headers	
Authorization	Bearer (token)

Respuesta	Parámetro	Código
Correcta	status	200
	message	Éxito al verificar el estado de los dispositivos.

- Información Punto Digital:

Invocación:

https://localhost:8082/middleware/info/Totem

Podrá obtener la ip, el hostname y la mac del punto digital.
Servicio de tipo **GET**

Headers	
Authorization	Bearer (token)

Respuesta	Parámetro	Código
Correcta	status	200
	message	Servicio ejecutado con éxito.

- Actualización Middleware:

Invocación:

https://localhost:8082/middleware/update/middleware

Podrá actualizar el middleware si existe una nueva versión.
Servicio de tipo **GET**

Headers	
Authorization	Bearer (token)

Respuesta	**Parámetro**	**Código**
Correcta	status	200
	message	Actualización en proceso.

- Descarga de Logs:

Invocación:

https://localhost:8082/middleware/download/logs

Podrá descargar los logs del Middleware y del app Punto digital.
Servicio de tipo **GET**

Headers	
Authorization	Bearer (token)
Logs	“Middleware” o “Totem”
date	aaaa-mm-dd

Respuesta	**Parámetro**	**Código**
Correcta	status	200
	message	Actualización en proceso.

- Anadir Logs app Punto Digital:

Invocación:

https://localhost:8082/middleware/Logs/add

Podrá agregar los logs del app Punto digital.
Servicio de tipo **POST**

Headers	
Authorization	Bearer (token)
status	“info” o “error” o “debug”

Body	
log	(texto para el log)

Respuesta	**Parámetro**	**Código**
Correcta	status	200
	message	Agregado correctamente.

Respuesta	**Parámetro**	**Código**
Incorrecta	status	201
	message	Error al añadir el log.

REGISTRO Y DESCUBRIMIENTO.

- **Eureka**

  Es un servidor que se usa para el registro y la localización de microservicios, balanceo de carga y tolerancia a fallos. La función de Eureka es registrar las diferentes instancias de microservicios existentes, su localización, estado, metadatos. En los servicios middleware Eureka se encarga de ver cada uno de los puntos digitales paras así registrar su estado y disponibilidad.

- Estado de eureka:

Invocación:

https://localhost:8761/eureka

Podrá consultar el estado de Eureka en el servidor.
Servicio de tipo **GET**

Respuesta	**Parámetro**	**Código**
Correcta	status	200
	message	Servicio ejecutado con éxito.

Invocación:

https://localhost:8761/eureka/apps

Podrá consultar los dispositivos registrados en Eureka.
Servicio de tipo **GET**

Respuesta	**Parámetro**	**Código**
Correcta	status	200
	message	Servicio ejecutado con éxito.