Documento Arquitectura de Biometria

Requerimiento:	**R-XYZ Invictus - Biometria**
Autor:	Andres David Rodriguez Ospina
Revisado por:	Oscar Mauricio Paredes Ortega
Fecha diligenciamiento:	12/07/2024

CONTROL DE CAMBIOS - Arquitectura Control de cambios	
Fecha	
Autor	
Descripción	
Versión	

1. Arquitectura de infraestructura

Cliente	
Sistema operativo	Ubuntu 20.0.4
Browser	Navegadores basados en Chromium y Renderizados por Blink, Firefox
Tecnología utilizada	APK Nativas, Arquitectura orientada a eventos
Lenguaje utiliado	Javascript (Framework Vue3)
Servidor	
Sistema operativo	Linux/X86_64
Servidos de aplicaciones	N/A
Lenguaje utilizado	Javascript (Interprete Node.js v18)
Frameworks utilizados	Restify
Base de datos	
Sistema operativo	Serverless
Motor	Aurora 8.0.mysql_aurora.3.03.1
Integraciones	
Proveedor	
Tecnología	

Ubicación documento ténico	
Dirección pruebas	
Dirección producción	

2. Contexto

Introducción

Este documento describe la arquitectura para implementar un sistema de biometría y enrolamiento en la arquitectura de Invictus. El objetivo es garantizar que todas las operaciones como pagos y otras funcionalidades configuradas para biometría, requieran autenticación biométrica antes de ser completadas.

Descripción

Se requiere agregar el nuevo dominio de biometría al nuevo modelo de arquitectura en Invictus.

Restricciones técnicas

Restricción	Justificación
Implementación en NodeJS 18, Vue3	El lenguaje en el que se construye la aplicación debe se acorde a donde va a ejecutar la aplicación, el backend de la aplicación debe ser construido en NodeJs 18, el frontend de la aplicación debe ser construida en Vue 3.
Cloud Native	El domino de biometria debe ser construido acorde al patrón de diseño definido para Invictus, en donde la solución como microservicio debe ser escalable, elástica y stateless
Implementación como dominio	El diseño de la solución debe ser contemplada como dominio de negocio independiente, garantizando la modularidad e independencia de otros módulos
Independencia del S.O	La solución debe poder ejecutarse en cualquier sistema operativo que permite la ejecución de un navegador moderno
Operación en móviles y equipos de escritorio	La aplicación debe operar en dispositivos móviles con versión de Android 10 o superior, asi como en equipos de escritorio con version Windows 10+ o Linux Ubuntu 18+

Desarrollo

Se requiere implementar el modulo de Biometria en Invictus.

Para desarrollar el modulo de Biometria en Invictus, se deben tener en cuenta los pasos desde la concepción del producto, hasta la conciliación del producto con el aliado. Teniendo esto en cuenta, contamos con varias fases a desarrollar, las cuales detallamos a continuación:

- Creación y definición del microfront end de biometría.
- Creación y definición del microbackend de biometría.
- Modelo de Datos
- Servicios de Biometría con el socket
- Módulo de Enrolamiento biométrico
- Módulo de Autenticación dactilar

Una vez segmentado todo el requerimiento en funcionalidades específicas, podemos trazar objetivos por cada funcionalidad y detallar a profundidad la misma de modo que se oriente correctamente la ejecución y el desarrollo de cada una. A continuación profundizaremos en las necesidades de cada funcionalidad detallada anteriormente:

Creación y definición del microfront end de Biometria

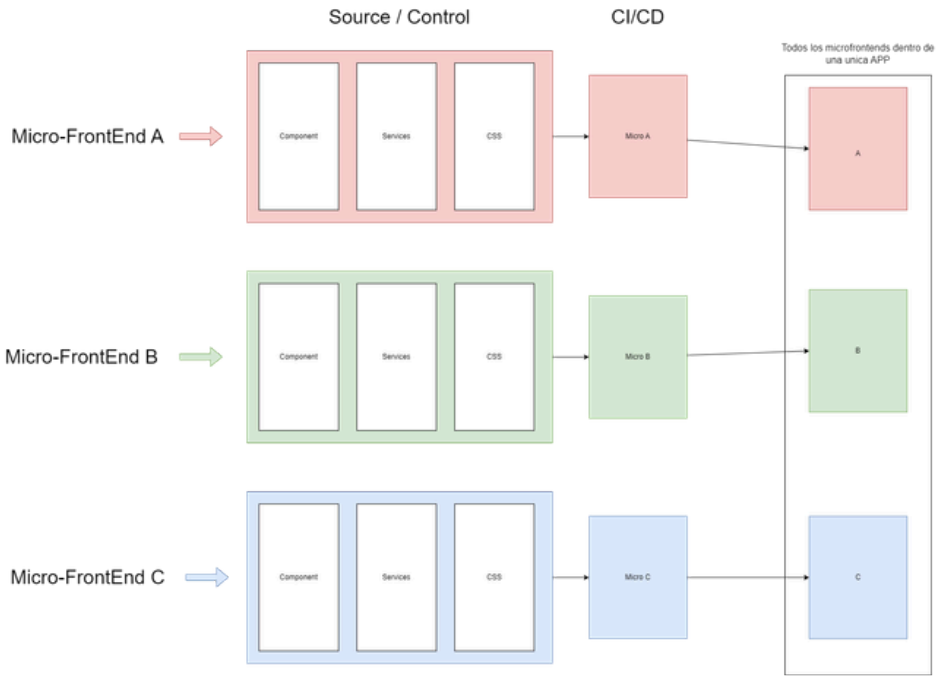
Consiste en crear un micro frontend para un nuevo dominio: biometrics. Este micro frontend tendrá integrada ambas aplicaciones que usamos actualmente en invictus: app-sellers y app-admin. También esta fase consiste en integrar el micro frontend en un único paquete durante el proceso de construcción. Este método exige una coordinación meticulosa y el cumplimiento de principios de diseño y acuerdos comunes compartidos.

Gestionar el enrutamiento y la navegación en una arquitectura de micro frontends requiere una consideración detallada. Estrategias como el enrutamiento basado en URLs, enrutamiento basado en rutas, o utilizar un enrutador centralizado, pueden contribuir a preservar una experiencia de usuario coherente a través de los diversos micro frontends.

Es crucial gestionar el estado compartido dentro de una arquitectura de micro frontends. Para manejar eficazmente el estado, se pueden utilizar técnicas como la gestión de estado local, la gestión de estado global con bibliotecas como Pinia (VueX no está soportado), o la comunicación basada en eventos.

Tecnología: Micro Frontends utilizando frameworks modernos (Vue 3)

- Un micro frontend específico para el dominio de biometria.
- Integración con otros micro frontends a través de un orquestador (Invictus).

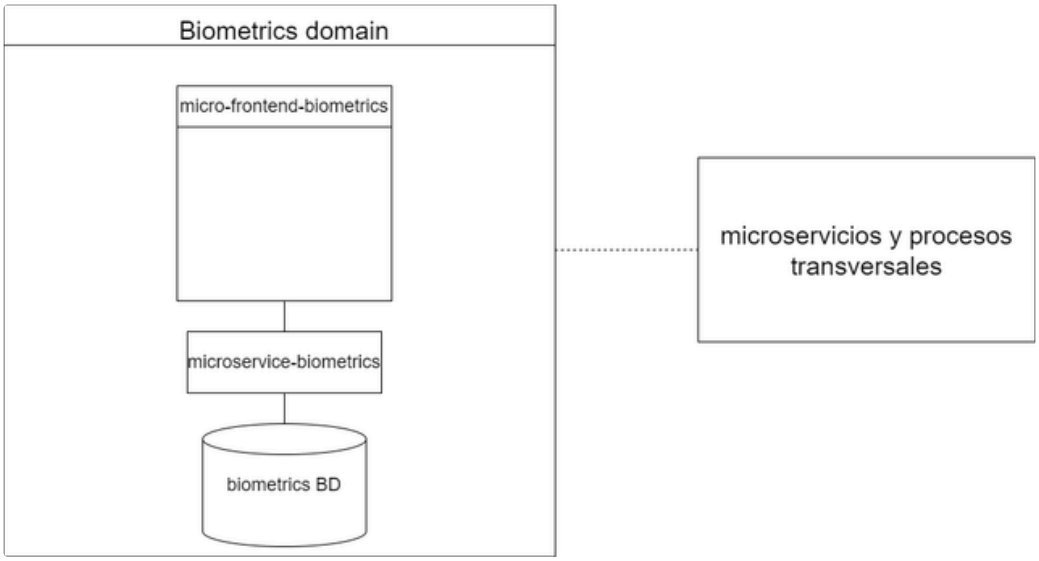


Esquema de estructura de los micro-front detallado

Creación y definición del microbackend de biometria

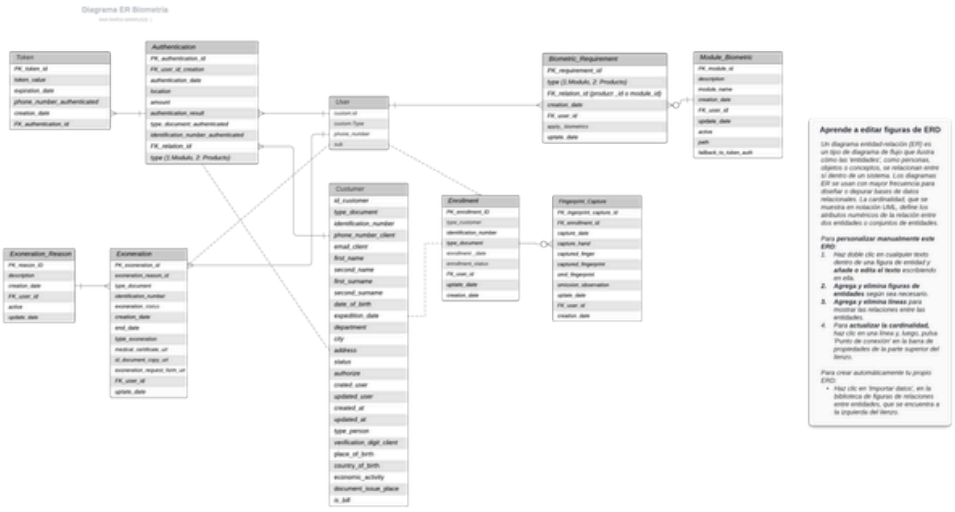
Se crea un nuevo microservicio (microservice-biometrics) dentro del nuevo dominio de biometría, el cual va responder y conectarse al micro frontend de dicho dominio.

Microservice-biometrics: Gestiona la biometria, exoneracion y validaciones de datos biometricos.



Modelo de datos:

MODELO ER:



La tabla encargada de almacenar la información de enrolamiento (**Enrollment**)

Representa los registros de enrolamiento de los clientes o usuarios en el sistema.

Contiene información sobre el tipo de cliente, número de identificación, tipo de documento, fechas de inscripción y actualización, y el estado de la inscripción.

Incluye referencias al usuario que creó o actualizó el registro.

Campo	Descripción	Detalles
PK_enrollment_ID	identificador único de la tabla	INT PK
type_customer	Tipo de cliente. Indica el tipo de cliente.	VARCHAR(50)
identification_number	El número de identificación del cliente.	VARCHAR(20)
type_document	Tipo de documento del cliente.	VARCHAR(50)
enrollment_date	Fecha y hora en que se realizó la inscripción.	TIMESTAMP
enrollment_status	Estado actual de la inscripción.	VARCHAR(20)
FK_user_id	Clave foránea que referencia al usuario que creó o actualiza el registro.	INT
update_date	Fecha y hora en que se actualizó el registro.	TIMESTAMP
creation_date	Fecha y hora en que se creó el registro.	TIMESTAMP

La tabla encargada de almacenar la información la captura de huella dactilar (**Fingerprint_Capture**)

Almacena los datos de las capturas de huellas dactilares de los clientes.

Incluye la fecha de captura, mano y dedo capturado, la huella capturada (opcional), y detalles de omisión de huella si es necesario.

Contiene referencias a la enrolacion correspondiente y al usuario que creó o actualizó el registro.

Campo	Descripción	Detalles
PK_fingerprint_capture_id	identificador único de la tabla	INT PK
FK_enrollment_id	Clave foránea que referencia a la inscripción.	INT
captured_date	Fecha y hora en que se capturó la huella.	TIMESTAMP
captured_hand	Indica si la huella capturada es de la mano izquierda o derecha.	VARCHAR(10)
captured_finger	Indica cuál dedo fue capturado.	VARCHAR(10)

captured_fingerprint	La imagen de la huella dactilar capturada.	BLOB
FK_user_id	Clave foránea que referencia al usuario que creó o actualiza el registro.	INT
update_date	Fecha y hora en que se actualizó el registro.	TIMESTAMP
creation_date	Fecha y hora en que se creó el registro.	TIMESTAMP
omit_fingerprint	Indica si se debe omitir la huella de un dedo específico.	BOOLEAN
omission_observation	Comentarios o razones para omitir la huella de un dedo.	VARCHAR(255)

La tabla encargada de almacenar la información de los motivos de exoneración (**Exoneration_Reason**)

Define los diferentes motivos por los cuales se puede otorgar una exoneración.

Contiene descripciones, fechas de creación y actualización, y un indicador de estado activo.

Incluye una referencia al usuario que creó o actualizó el registro.

Campo	**Descripción**	**Detalles**
PK_reason_ID	Clave primaria del motivo de exoneración.	INT PK
description	Descripción detallada del motivo de exoneración.	VARCHAR(255)
FK_user_id	Clave foránea que referencia al usuario que creó o actualiza el registro.	INT
update_date	Fecha y hora en que se actualizó el registro.	TIMESTAMP
creation_date	Fecha y hora en que se creó el registro.	TIMESTAMP
active	Indica si el motivo de exoneración está activo.	BOOLEAN

La tabla encargada de almacenar la información de la exoneración (**Exoneration**)

Gestiona los registros de exoneración para los clientes.

Contiene información sobre el motivo de la exoneración, tipo y número de documento, estado de la exoneración, fechas de finalización, y tipos de exoneración.

Almacena URLs de documentos relevantes, como el certificado médico y la fotocopia del documento de identificación.

Incluye una referencia al usuario que creó el registro.

Campo	Descripción	Detalles
PK_exoneration_id	identificador único de la tabla	INT PK
exoneration_reason_id	Referencia al motivo específico de la exoneración.	INT
type_document	Tipo de documento del individuo exonerado.	VARCHAR(50)
identification_number	Número de identificación del individuo exonerado.	VARCHAR(20)
status	Indica el estado actual de la exoneración (aprobado, pendiente, rechazado).	VARCHAR(20)
end_date	Fecha y hora en que finaliza la exoneración.	TIMESTAMP
type_exoneration	Indica el tipo específico de exoneración.	VARCHAR(50)
medical_certificate_url	URL de la dirección del almacenamiento del Certificado médico en S3.	VARCHAR(255)
id_document_copy_url	URL de la fotocopia del documento de identificación.	VARCHAR(255)
exoneration_request_form_url	URL del Formato de Solicitud de Exoneración.	VARCHAR(255)
FK_user_id_creation	Clave foránea que referencia al usuario que creó el registro de exoneración.	INT
creation_date	Fecha y hora en que se creó el registro de exoneración.	TIMESTAMP
update_date	Fecha y hora en que se actualizó el registro.	TIMESTAMP

La tabla encargada de almacenar la información de si se requiere biometria (**Biometric_Requirement**)

Define los módulos o productos específicos que requieren biometría .

Contiene información sobre el tipo módulo o producto, fechas de creación y actualización, y si se debe aplicar biometría.

Incluye una referencia al usuario que creó el registro.

Campo	Descripción	Detalles
PK_requirement_id	identificador único de la tabla	INT PK
type	Tipo (1: Módulo, 2: Producto).	TINYINT(1)
FK_relation_id	Clave foránea que referencia al producto o módulo asociado.	INT
creation_date	Fecha y hora en que se creó el requisito biométrico.	TIMESTAMP
FK_user_id	Clave foránea que referencia al usuario que creó el registro del requisito.	INT
apply_biometrics	Indica si se debe aplicar biometría.	BOOLEAN
update_date	Fecha y hora en que se actualizó el registro.	TIMESTAMP

La tabla encargada de almacenar la información de los módulos de biometría (**Module_Biometric**) :

Representa los módulos del sistema que requieren autenticación biométrica.

Contiene descripciones, nombres de módulos, rutas, y un indicador de si se debe pasar a autenticación por token cuando no hay un lector conectado.

Incluye referencias a la creación y actualización del registro

Campo	Descripción	Detalles
PK_module_id	identificador único de la tabla	INT PK
description	Descripción del módulo al que se le aplicara biometría.	VARCHAR(255)
module_name	El nombre del módulo al que se le aplicara biométrica.	VARCHAR(100)
creation_date	Fecha y hora en que se creó el módulo al que se le aplicara biometría.	TIMESTAMP
active	Indica si el módulo al que se le aplicara biometría está activo.	BOOLEAN
path	La ruta o ubicación del módulo al que se le aplicara biometría.	VARCHAR(255)
fallback_to_token_auth	Indica si se debe pasar a la autenticación por token cuando no hay un lector conectado.	BOOLEAN

FK_user_id	Clave foránea que referencia al usuario que creó el registro del requisito.	INT
update_date	Fecha y hora en que se actualizó el registro.	TIMESTAMP

La tabla encargada de almacenar la información histórica de la autenticación (**Authentication**) :

Registra las autenticaciones realizadas en el sistema.

Contiene información sobre la fecha, ubicación, monto (opcional), resultado de la autenticación, y los documentos del cliente o usuario autenticado.

Incluye una referencia al usuario que creó el registro y al producto o módulo asociado.

Campo	Descripción	Detalles
PK_authentication_id	identificador único de la tabla	INT PK
FK_user_id_creation	Clave foránea que referencia al usuario que creó el registro de autenticación.	INT
authentication_date	Fecha y hora en que se realizó la autenticación.	TIMESTAMP
location	Lugar donde se realizó la autenticación.	VARCHAR(100)
amount	El monto asociado a la transacción.	DECIMAL(10,2)
authentication_result	Indica el resultado de la autenticación (éxito, fallo, etc.).	VARCHAR(20)
type_document_authenticated	Tipo de documento del cliente o usuario autenticado.	VARCHAR(50)
identification_number_authenticated	Número de identificación del cliente o usuario autenticado.	VARCHAR(20)
type	Tipo (1: Módulo, 2: Producto).	TINYINT(1)
FK_relation_id	Identificador de la relación (product_id o module_id).	INT

La tabla encargada de almacenar la información histórica de los tokens (**Token**) :

Gestiona los tokens utilizados para la autenticación.

Contiene el valor del token, fecha de expiración, y fecha de creación.

Incluye una referencia a la autenticación relacionada.

Campo	Descripción	Detalles
PK_token_id	identificador único de la tabla	INT PK
FK_authentication_id	Clave foránea que referencia a la entidad de autenticación relacionada con el token.	INT
token_value	El valor único del token utilizado para la autenticación.	VARCHAR(5)
expiration_date	Fecha y hora en que expira el token.	TIMESTAMP
creation_date	Fecha y hora en que se creó el token.	TIMESTAMP

Servicios del socket de biometría:

Este documento detalla la implementación de dos servicios de biometría dentro del sistema INVICTUS, específicamente relacionados con la captura y verificación de huellas digitales. Estos servicios se implementan utilizando sockets y están disponibles en los siguientes endpoints:

- `http://localhost:1024/fingerprint`
- `http://localhost:1024/fingerprint/verify`

1. Endpoint: `/fingerprint`

Método HTTP: `GET`

Descripción Funcional: El servicio `fingerprint` se utiliza para capturar la huella digital de un usuario a través de un lector de huellas. Este servicio se conecta al hardware del lector, captura la imagen de la huella digital y la devuelve en un formato específico que puede ser utilizado para almacenamiento o verificación posterior.

Descripción Técnica:

- **Conexión al Lector:** El servicio se conecta al lector de huellas digitales disponible.
- **Captura de Huella:** Una vez conectado, el servicio captura la imagen de la huella digital del usuario.
- **Formato de la Huella:** La imagen capturada se convierte en un objeto `FingerPrintDTO`, que incluye el formato de la huella y los datos de la misma.
- **Respuesta:** El servicio devuelve un objeto `ResponseMessage<FingerPrintDTO>` que contiene los datos de la huella digital capturada.

Esquema de Respuesta:

```yaml
ResponseMessageFingerPrintDTO:
  type: object
  properties:
    code:
      type: integer
      format: int32
    message:
```

```
      type: string
    data:
      $ref: '#/components/schemas/FingerPrintDTO'rPrintDTO'
```

2. Endpoint: `/fingerprint/verify`

Método HTTP: `POST`

Descripción Funcional: El servicio `verifyFingerPrint` se utiliza para verificar si una huella digital capturada coincide con otra huella digital previamente registrada. Este servicio toma los datos de una huella digital, captura una nueva huella del usuario, y las compara para determinar si pertenecen a la misma persona.

Descripción Técnica:

- **Recepción de Datos:** El servicio recibe un objeto `FingerPrintDTO` que contiene los datos de una huella digital previamente registrada.
- **Captura de Nueva Huella:** Captura una nueva huella digital del usuario utilizando el mismo proceso que el endpoint `/fingerprint`.
- **Comparación:** Utiliza un motor de comparación biométrica para comparar la huella digital recién capturada con la huella digital proporcionada.
- **Resultado:** Devuelve un objeto `ResponseMessage<Boolean` > indicando si las huellas digitales coinciden o no.
-

Esquema de Petición y Respuesta:

Petición:

```
FingerPrintDTO:
  type: object
  properties:
    format:
      type: string
    fingerprint:
      type: array
      items:
        type: string
        format: byte
```

Respuesta:

```
ResponseMessage<Boolean>:
  type: object
  properties:
    code:
      type: integer
      format: int32
    message:
      type: string
    data:
      type: boolean
```

Modelos:

```
FingerPrintDTO(String format, byte[] fingerprint) {
       String format;
       byte[] fingerprint;
   }
```

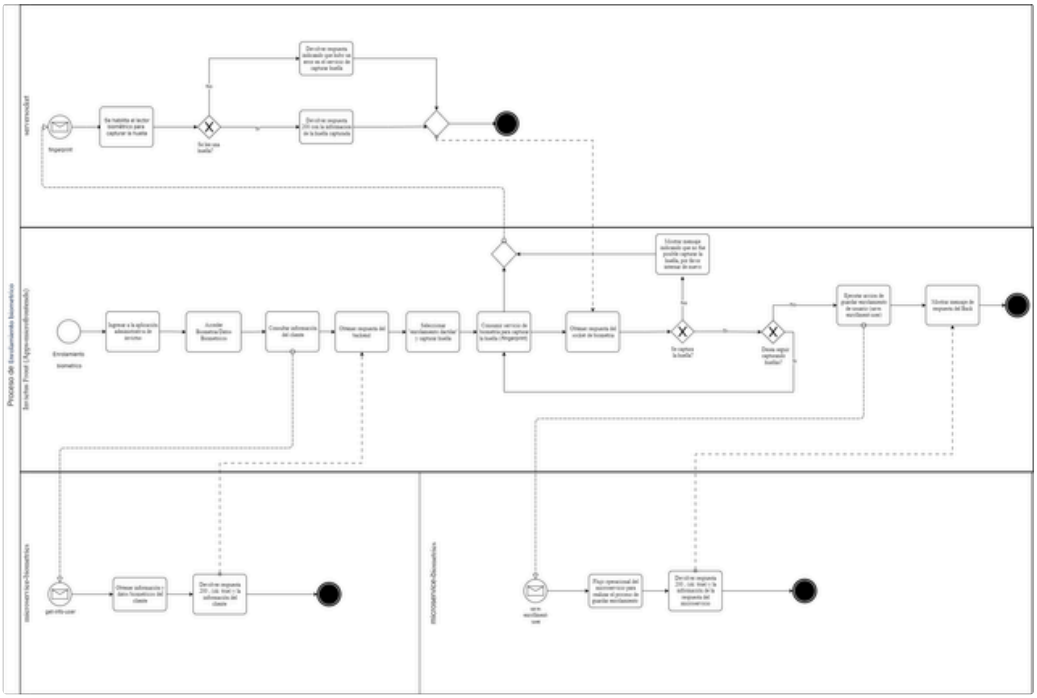
```
ResponseMessage(String format, byte[] fingerprint) {
       int code;
       String message;
       T data;
   }
```

Estos dos servicios son componentes críticos para la implementación de autenticación biométrica en el sistema INVICTUS. Proveen funcionalidades esenciales para la captura y verificación de huellas digitales, asegurando un alto nivel de seguridad y precisión en la identificación de usuarios.

Módulo de Enrolamiento biométrico:

El modulo de enrolamiento biométrico consumirá desde el app-microfront-ends el servicio de `/fingerprint` que es el encargado de capturar la huella desde el dispositivo biométrico conectado y devolver la información de la huella para posteriormente después de tener todas las huellas necesarias llamar al bff-administration para este llamar al microservice-biometrics para guardar esta información en la tabla XYZ de la BD del dominio.

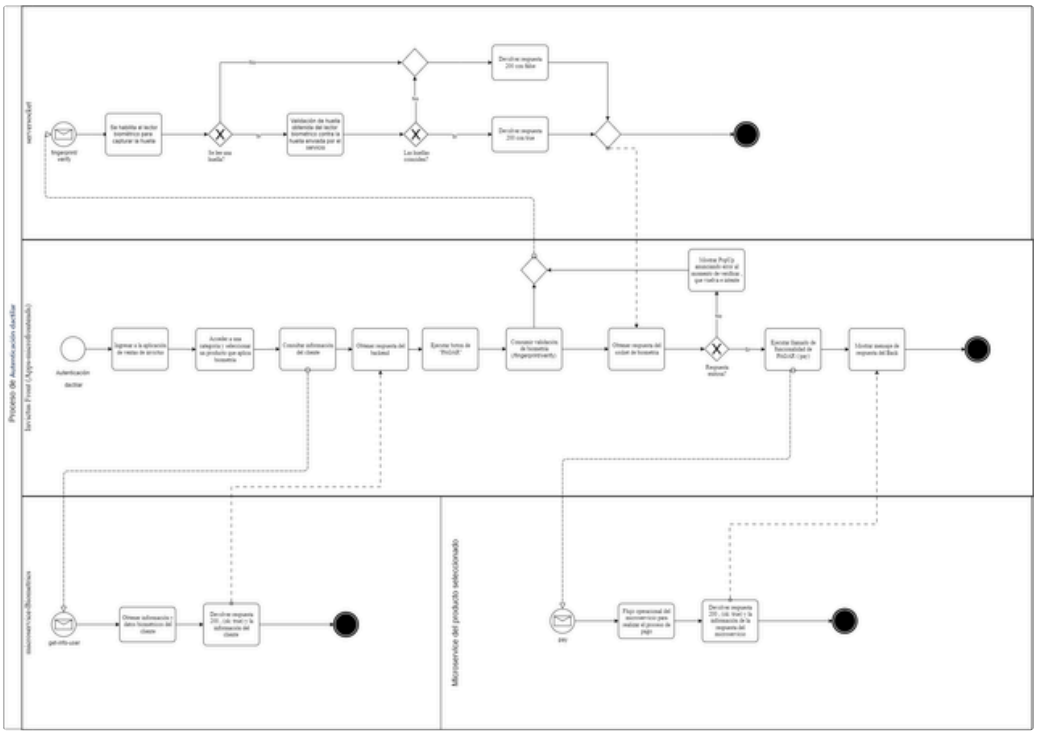
Diagrama de procesos de Enrolamiento biométrico:



Módulo de Autenticación dactilar:

El modulo de Autenticación dactilar consumirá desde el app-microfront-ends el servicio de `/fingerprint/verify` , el servicio se encargara de validar si la huella enviada del cliente coincide con la huella leída por el lector biométrico y retornara true en caso de que coincidan o false en caso de que no sean las mismas huellas.

Diagrama de procesos de Autenticación dactilar:



3. Requisitos no funcionales

Descripción

Los siguientes requerimientos no funcionales son importantes para garantizar un producto seguro y que genere mayor valor para el cliente.

- Pruebas Unitarias 100%
- Pruebas de cobertura 100%
- Pruebas de Eslint y SonarQube al 100%
- Componentes Agnósticos
- Tiempos transaccionales de máximo 3 segundos.

Conclusión

Este documento proporciona una guía estructurada para implementar un sistema de biometría en la arquitectura de invictus. Se cubren todos los aspectos desde la comunicacion del front , al microservicio del backend y comunicación con el `ServerSocket`. La estructura modular y la implementación en la nube aseguran escalabilidad, mantenibilidad y una fácil integración con todo Invictus.

Responsable de la actividad (Líder del Proyecto / Autor)	
Espacio para firma	
Nombre:	
Cargo:	
Fecha:	