

Universidad Nacional del Noroeste de la Provincia de Buenos Aires

Título: La factura electrónica y su automatización

Carrera: Ingeniería en informática

Alumno: Lucas Ezequiel Ormazabal

Supervisor Docente: Lic. Carlos Di Cicco

Tutor de Empresa: Edgardo Lacquaniti

Fecha de presentación: 16 de marzo de 2023

Contenido

1. Introducción:.....	3
2. Objetivos:	3
2.1 Objetivo general:.....	4
2.2 Objetivos específicos:.....	4
3. Plan de trabajo y carga horaria:	4
4. Descripción de la práctica profesional efectuada:.....	5
4.1 Análisis de requerimientos:	5
4.2 Modelado de backend y frontend:	6
4.3 Investigación - Análisis de documentación:.....	7
4.4 Desarrollo módulo backend:.....	7
4.4.1 Web Service de Autorización y Autenticación:	8
4.4.2 Web Service de Factura electrónica:	15
4.5 Desarrollo módulos frontend:.....	21
4.5.1 Módulo Front Office:.....	22
4.5.2 Módulo Back Office:.....	23
4.6 Análisis y desarrollo de nuevos requerimientos:	26
5. Conclusiones:	31
6. Bibliografía:	31

Imágenes

Figura 1 – Cronograma de tareas.....	4
Figura 2 - Registro	6
Figura 3 - Tabla de operaciones	7
Figura 4 - Menú	7
Figura 5 - Flujo entre servicios	8
Figura 6 - SoapUI - Request.....	12
Figura 7 - SoapUI - Response	13

Figura 8 - XML SOAP para número de factura	16
Figura 9 - XML SOAP para facturación	17
Figura 10 - XML SOAP Response	19
Figura 11 - Tabla de Facturas pendientes - BackOffice	23
Figura 12 - Modal de confirmación de facturación.....	24
Figura 13 - Modal de facturación exitosa	24
Figura 14 - Tabla de Facturas pendientes y clientes con cuit no válido – BackOffice.....	25
Figura 15 - Toastr de Error - BackOffice	25
Figura 16 - Tooltip Editar cliente - BackOffice.....	25
Figura 17 - Tabla de Operaciones por cliente - BackOffice	26
Figura 18 - Detalles de operación - BackOffice	26
Figura 19 - Tabla de Facturas emitidas- BackOffice	27
Figura 20 - Tabla de Facturas emitidas filtradas- BackOffice.....	28
Figura 21 - Documento .csv exportado - BackOffice.....	28
Figura 22 - Tabla de información de un cliente - BackOffice	29
Figura 23 - Tabla de Facturas por cliente- BackOffice	29
Figura 24 - Factura	30

Presentación del Informe

1. Introducción:

El presente informe fue redactado con supervisión del Licenciado en Sistemas Carlos Di Cicco (Profesor supervisor), y resume las tareas que fueron realizadas en el marco del plan de trabajo presentado, en el rol de Desarrollador de software Full-Stack en la empresa Espin Labs S.R.L, para la plataforma de la empresa Decrypto S.A.S, con el objetivo de que dichas tareas sean consideradas válidas y suficientes para acreditar las 200hs de Práctica Profesional Supervisada y así completar los requisitos académicos de la carrera Ingeniería en informática.

Decrypto S.A.S se autodefine como "*un broker y billetera de inversiones crypto, que tiene como principal objetivo que cada vez más personas puedan conectar el mundo tradicional con el crypto*", fundada en 2018, siendo uno de los principales clientes de la empresa Espin Labs S.R.L.

2. Objetivos:

Con la ejecución de la presente PPS se buscó poder satisfacer la necesidad del cliente para automatizar el proceso de facturación de la empresa, el cual, al realizarse de forma manual, encontraba desventajas de las cuales se pueden destacar: la imposibilidad de llevar el control absoluto de las operaciones facturadas; la cantidad de tiempo que esto llevaba; como así también

Análisis de requerimientos: se determinó la claridad, validez, consistencia y completitud de los requisitos.

Modelado de backend y frontend: en esta etapa, se definen tanto los flujos de datos que tendrá el sistema, como también las entidades y tablas que se utilizarán para su construcción.

Investigación – Análisis de documentación: se realizó un estudio exhaustivo de la documentación proporcionada por el organismo, a fin de tener en claro conocimiento los recursos necesarios para poder consumir su servicio.

Desarrollo módulo backend: esta etapa abarca desde las pruebas con herramientas externas, creación de certificados y registro en el organismo hasta el propio desarrollo del facturador.

Desarrollo módulos frontend: se diseñó una plantilla de factura, se agregaron botones en el Front Office y Back Office, tablas y se conectaron los módulos.

Corrección de errores y deploy en servidores de prueba: se realizan ajustes al sistema y se generan los recursos necesarios para utilizar el servicio en los servidores de prueba.

Análisis y desarrollo de nuevos requerimientos: se analizan y desarrollan los nuevos requerimientos del sistema.

Testing: se realizaron pruebas exhaustivas con el único propósito de analizar el comportamiento del sistema.

4. Descripción de la práctica profesional efectuada:

4.1 Análisis de requerimientos:

El primer paso fue analizar las funcionalidades requeridas por Decrypto S.A.S. Luego de una reunión con el líder de equipo y nuestro contacto con dicha empresa, se pudieron determinar los siguientes requerimientos:

Para los clientes:

- Cada usuario de la plataforma que resida en Argentina, debe poder generar una factura electrónica en formato PDF para las operaciones de depósito y retiro.
- Se debe programar un procedimiento que realice el proceso de facturación, para dichas operaciones, cada cierto tiempo a definir (no en tiempo real).

Para los usuarios del Back office:

- Se deben poder facturar las operaciones en tiempo real.
- En caso de que un proceso de facturación presente algún fallo, se deben poder visualizar en una tabla indicando, entre otras cosas, el motivo por el cual se produjo dicho fallo.
- Se debe permitir intentar refacturar, en caso de que se desee, las operaciones que presenten algún fallo
- En caso de que se encuentren clientes con cuit/cuil no válidos para AFIP, se deben poder visualizar en una tabla.
- Cada factura de la tabla se puede descargar en su formato original (factura para el cliente).

4.2 Modelado de backend y frontend:

Durante esta etapa, se tomaron decisiones sobre cómo se representará en el sistema la funcionalidad requerida por la empresa.

En la plataforma del cliente o Front Office, se decidió agregar en el registro los nuevos campos requeridos para el proceso de facturación. Para esto se agregaron dos combos con la información necesaria correspondiente, cuyo resultado final se puede observar en la figura 2.

Figura 2 - Registro

Personales

Tipo de Usuario

Persona física

Datos Personales

Lucas Ormazabal

10/07/1992
e.g "dd/mm/yyyy"

Masculino

ARGENTINA

Consumidor Final

Situación IVA *

Consumidor Final

IVA Responsable Inscripto

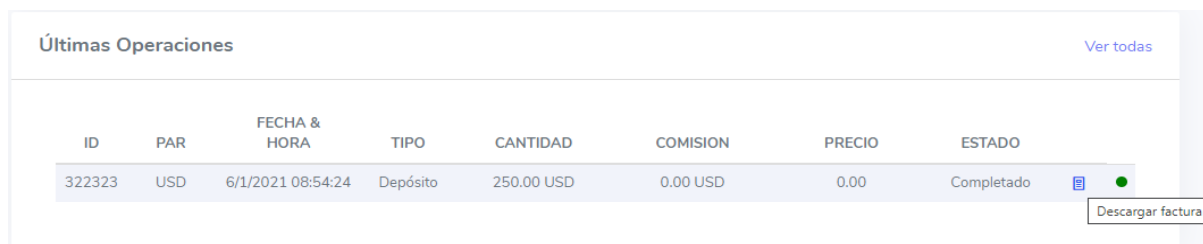
Responsable Monotributo



Soporte

En el dashboard, se puede observar una tabla que posee una lista con las operaciones realizadas por cada cliente, las cuales pueden ser depósito, retiro, trading-compra, trading-venta, transferencia, inversión o acreditación de referidos.

Al realizar operaciones de depósito o retiro, se ven reflejadas en dicha tabla con la particularidad que ambas poseen un botón para poder descargar la factura, como muestra la figura 3. Cabe destacar que dicha operación se realizó en un ambiente de testing.

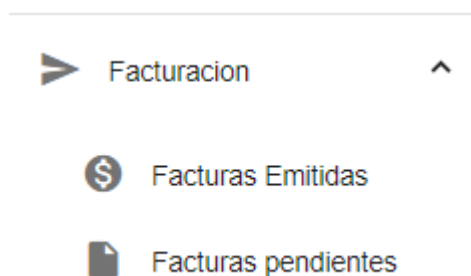
En la plataforma de los administradores o Back Office, se prosiguió agregando un menú en la barra lateral *Figura 3 - Tabla de operaciones*



ID	PAR	FECHA & HORA	TIPO	CANTIDAD	COMISION	PRECIO	ESTADO	
322323	USD	6/1/2021 08:54:24	Depósito	250.00 USD	0.00 USD	0.00	Completado	  Descargar factura

con la leyenda “Facturación” y un desplegable, con dos opciones que se especificarán más adelante en este mismo informe, como muestra la figura 4.

Figura 4 - Menú



4.3 Investigación - Análisis de documentación:

Esta etapa comprende las actividades de investigación y análisis de documentación [1], [2], [3], [4], [5], [6], [7] proporcionada por la entidad AFIP. A lo largo de dos días, se estudió detalladamente el funcionamiento del Web Service que utiliza dicha entidad, el cual utiliza un servicio que basa su comunicación bajo el protocolo SOAP, el flujo de datos, los requerimientos previos para su funcionamiento, la información necesaria y los trámites necesarios en la plataforma de dicha entidad.

Cabe resaltar que, aunque el tiempo dedicado al estudio exhaustivo de la documentación fue aproximadamente dos días, la misma fue consultada a lo largo del desarrollo completo del módulo requerido por el cliente.

4.4 Desarrollo módulo backend:

Como primera medida, se analizó la estructura de la arquitectura general del sistema de AFIP. El intercambio de información entre la entidad y los Entes Externos (EE) se lleva a cabo a través de web services SOAP sobre el protocolo HTTPS.

La entidad posee varios Web Services de Negocio (WSN) los cuales son accesibles sin el establecimiento de canales especiales de comunicación ni VPNs.

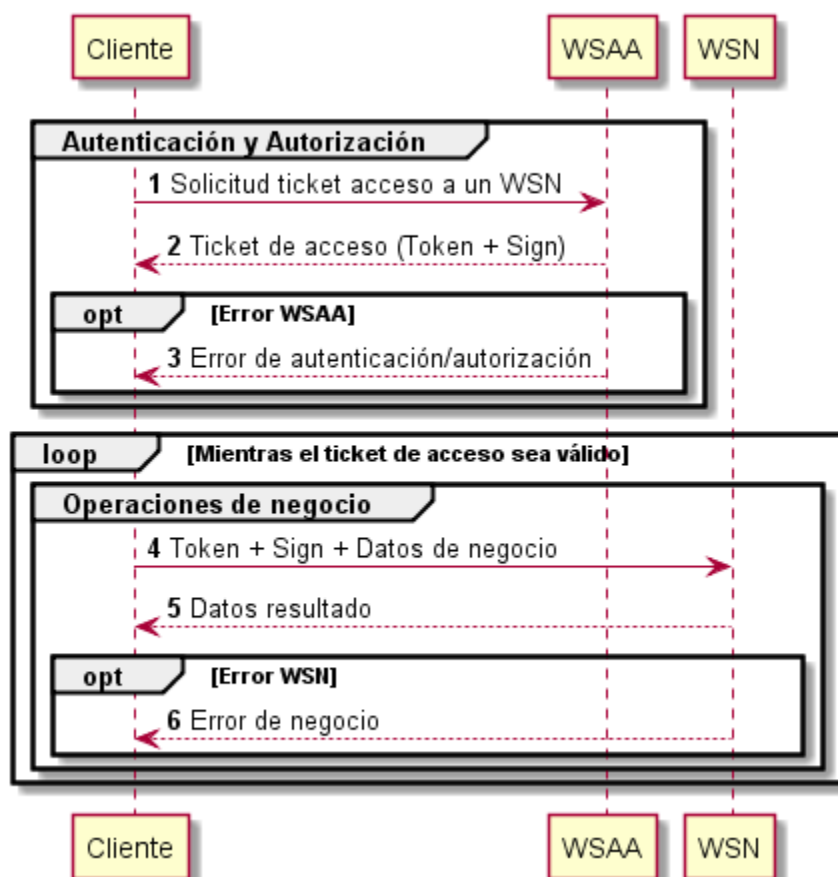
El acceso a estos WSN está regulado por otro web service denominado Web Service de Autenticación y Autorización (WSAA) que es el encargado de autenticar los EE y conceder los permisos de acceso a cada WSN a través de un Ticket de Acceso (TA). Cada TA es válido únicamente para un WSN en particular y tiene una validez limitada en el tiempo, el cual en la actualidad es de 12 horas.

En cada petición a los WSN, se debe presentar el TA obtenido por el WSAA, de lo contrario se negará el acceso a dichos servicios.

Todos estos WSN se encuentran replicados en dos entornos, uno de Producción y otro de Testing/Homologación.

El siguiente diagrama representa el flujo de datos entre los servicios, tanto WSAA como WSN.

Figura 5 - Flujo entre servicios



4.4.1 Web Service de Autorización y Autenticación:

La autenticación del cliente se realiza utilizando criptografía de clave pública basada en certificados

digitales .509. donde AFIP actúa como Autoridad Certificante y emite estos certificados.

Una vez obtenido el certificado digital el EE llevará a cabo los trámites requeridos en cada caso para obtener la autorización inicial y así acceder a un WSN que lo relacionará con un certificado digital.

El WSAA autenticará al cliente mediante la verificación de su firma digital y la comprobación en su base de datos. Previo a esto, se espera que el EE haya completado los trámites de autorización inicial para acceder al WSN; si estos controles son superados, entonces el WSAA contestará devolviendo un TA, de lo contrario, devolverá un mensaje de error explicativo.

El cliente del WSN extraerá del TA dos componentes, Token y Sign, y los enviará junto con los datos de negocio en cada solicitud que le envíe al WSN.

Para solicitar el acceso al WSAA y WSN, se debe pedir por medio del "Administrador de Relaciones de Clave Fiscal" en www.afip.gob.ar autenticándose con la clave fiscal de una persona física.

Para obtener el certificado digital .pem, vamos a necesitar generar dos archivos, utilizando OpenSSL por consola. Primero habrá que generar una clave privada (.key); y luego un archivo CRS (.csr).

Para generar la clave, utilizamos el siguiente comando:

```
openssl genrsa -out MiClavePrivada.key 2048
```

donde MiClavePrivada es el nombre del archivo con la clave privada que obtenemos. Es necesario guardar este archivo en un lugar seguro, ya que será necesario, junto con el certificado creado para firmar los mensajes.

Una vez obtenida la misma, se procede con la obtención del archivo CSR, utilizando el siguiente comando:

```
openssl req -new -key MiClavePrivada.key -subj "/C=AR/O=Empresa/CN=Sistema/serialNumber=CUIT  
nnnnnnnnnn" -out MiPedidoCSR.csr
```

MiClavePrivada.key es el archivo obtenido en el paso anterior; Empresa es el nombre de la empresa; Sistema por el nombre del sistema cliente; nnnnnnnnnn es el CUIT y MiPedidoCSR es el nombre del archivo CSR que se va a crear.

El certificado obtenido será similar al siguiente:

```
-----BEGIN CERTIFICATE REQUEST-----  
MIICITCCAQAwUDELMAkGA1UEBhMCQVlxEjAQBgNVBAoTCVNPUE9SVEVXUzES  
MBAGA1UEAxMJU09QT1JURVdTMRkwFwYDVQQFExBDVUUIUIDIwMTkwMTc4MTU0MIIIB  
ljANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAYu7TzTjKEdXEZPR4wlhFOW1S ...  
QMNYAgJ/J2Zyy2JzxtsHzZDN1oM1SjB9G9KyA5Rp02QZMMMsMWIYKQLsoM5QAPQH  
DWxkDa8dm7tBylM3u5+XT5G+w1xH4WjRHViAmUH2U+hTmaVAj7qSxsQzR/5HWoKX  
cnMXsTGvi/5Y9q9kuOw6csm43z5XvxT4BBBKZn6FqWqSwUKxVuaJU8Q=  
-----END CERTIFICATE REQUEST-----
```

Una vez obtenido el certificado digital .csr para el entorno de homologación, hay que asociarlo al Web Service de negocio al que se va a acceder. Esta asociación se realiza en la aplicación WSASS, para el entorno

de testing; o en la aplicación “Administrador de Relaciones de Clave Fiscal”, para el entorno de producción.

Para poder acceder a un servicio en ambiente de testing, la aplicación a programar debe utilizar el certificado generado en el WSASS. Entre otras cosas, el certificado contiene un Distinguished Name (DN) que incluye un CUIT del contribuyente. Cada DN será identificado por un “alias” o “nombre simbólico”, que actúa como una abreviación.

Cabe destacar que el CUIT que se utilice debe ser el mismo CUIT que se incluyó en la solicitud del certificado CSR (en el campo serialNumber del CSR).

Una vez que se complete el formulario que nos solicita el DN, CUIT y CSR, si no hay errores, el sistema nos devolverá un certificado x509 en formato PEM, similar al que se encuentra a continuación:

-----BEGIN CERTIFICATE-----

```
MIIDSzCCAjOgAwIBAgII57JlkcfpQhswDQYJKoZIhvcNAQENBQAwMzEVMBMGGA1UEAwwMQ29tcHV0
YWRvcmlvZmQ0wCwYDVQQKDARBRklQMzswCCQYDVQQGEwJBUjAeFw0xNjA5MzAxMTUzMjhaFw0xODA
5MzAxMTUzMjhaMDYxGTAXBgNVBAMMEENlcnRfZ2ZxcnJpZlZlXzExZG90YXN0YXN0YXN0YXN0YXN0
OTAxNzgxNTQwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDK7tNIOMoR1cRk9HjAiEU7 ...
z2JEQXFPkcxG6DE9v4Q/8WSaiPRxbzjOkC5+cEuEtqxlwGsCDeFjSRco05XFWmzXg8jLrJIEk8l
jFRvrlR9Shm/p6RxU6ZeBkyeNAu3c5ShTyL0tntIEoXDpx4wqZ2ZkA4tinmbbl7LnnCyaniKw27h
bkQkLybJPc1LlZJb9tRzFQUu9PreRj1ggnhzzo/qtCicv6oVoi7nZW05lO5mG8JxJQhEfnVgZqw
0QRQQb4/Oueqfgw84C1/dD2TKH9Rsjj8tiVvsCuz5tjkb727tH/ZW3ce2bG9t4QQ==
```

-----END CERTIFICATE-----

Llegado a este punto, solo resta copiar el certificado obtenido, pegarlo en un editor de texto y guardarlo, por ejemplo, en un archivo MiCertificado.pem.

En este momento, vamos a tener 3 archivos:

- MiClavePrivada.key
- MiPedidoCSR.csr
- MiCertificado.pem

Para poder probar que los archivos fueron generados de manera correcta, vamos a crear un CMS firmado para luego intentar obtener el Ticket de Acceso (TA) haciendo uso de una herramienta llamada SoapUI.

La solicitud de ticket de acceso es un documento XML indicando el identificador del web service de negocio al que se solicita acceder, el cual, en este caso es el “wsfe”, que significa Web Service de Factura Electrónica.

El archivo XML debe tener el siguiente formato:

```
<loginTicketRequest>
  <header>
    <uniqueId>190926</uniqueId>
    <generationTime>2020-08-19T10:47:20</generationTime>
    <expirationTime>2020-08-19T20:17:20</expirationTime>
```

```

</header>
<service>wsfe</service>
</loginTicketRequest>

```

uniqueId: Entero de 32 bits sin signo que junto con “generationTime” identifica el requerimiento.
generationTime: Momento en que fue generado el requerimiento. La tolerancia de aceptación será de hasta 24 horas previas al requerimiento de acceso.
expirationTime: Momento en el que expira la solicitud. La tolerancia de aceptación será de 12 horas posteriores al requerimiento de acceso (aproximado).
service: Identificación del WSN para el cual se solicita el TA.

Aquí, la idea es que nuestro sistema obtenga un TA y lo pueda utilizar múltiples veces, siempre y cuando el mismo continúe vigente.

Por último, solo resta crear el CMS firmado, para el cual necesitaremos los siguientes tres archivos:

- MiCertificado2019.pem (certificado para homologación ya generado)
- MiPrivada2019.key (clave privada ya generada)
- MiLoginTicketRequest.xml (la solicitud de ticket de acceso).

Utilizaremos el siguiente comando:

```

openssl cms -sign -in MiLoginTicketRequest.xml -out MiLoginTicketRequest.xml.cms -signer
MiCertificado2019.pem -inkey MiPrivada2019.key -nodetach -outform PEM

```

Una vez ejecutado, nos devolverá el archivo MiLoginTicketRequest.xml.cms, que fue como llamamos al CMS firmado, el cual es similar al siguiente:

```

-----BEGIN CMS-----
MIIG2AYJKoZlIhvcNAQcCoIIgYTCCBsUCAQExDTALBglghkgBZQMEAgEwgf8GCSqG
S1b3DQEHAaCB8QSB7jxsb2dpbIRpY2tldFJlcXVlc3Q+PGhIYWRIcj48dW5pcXVI
SWQ+MTkwOTI2PC91bmlxdWVJZD48Z2VuZXJhdGlvbIRpbWU+MjAxOS0wOS0yNIQx
MDoyOTxNTWwZ2VuZXJhdGlvbIRpbWU+PGV4cGlyYXRpb25UaW1lPjJlMTktMDkt
MjZUMTA6NDk6MTU8L2V4cGlyYXRpb25UaW1lPjwvaGVhZGVyPjxzZXJ2aWNlPndz
...
AIAwDQYIKoZlIhvcNAwICAUAwBwYFKw4DAgcwDQYIKoZlIhvcNAwICASgwDQYJKoZI
hvcNAQEBAEggEADDXstf5M89wT9IZnAz2N/Yn0r7aGhCQ30U9UAb6BkLfXta6F
dbCqDbqO7ekUyO/GjSR9R6hMGsxn7lg9U+JZ+yCVfPENaZItOAOPnfJ3gWih7U5c
loPLAIC1WvjSxoKcN81NG5ZkxNGmik9K2pKv86BdtN/1BABDFak5QO8WcTo2Wptc
VTzDyJsvs2eWsGeTqbm+AOuNfGqn4w6ITYJRuQgEoHPAC0Y2S3tZCLJKHcK5t9mQ
Kle47uzLLjP4pN/QAx0hOxbUU7yvHC/2rWrzuiu/iEXbITgu7imyxp2h4Dn+RRY1
wzR9q/SH/VoTI63TbqNDxjQvpb3+bNewHwd8Hw==
-----END CMS-----

```

Ahora probamos la validez de los certificados utilizando SoapUI. Para ello, creamos un nuevo proyecto

SOAP, le damos un nombre y en el campo WSDL, especificamos el servicio web que queremos utilizar.

Para utilizar el servicio web de WSAA de homologación, utilizaremos el siguiente WSDL:

<https://wsaahomo.afip.gov.ar/ws/services/LoginCms?wsdl>

Dentro de nuestro proyecto, buscamos una pestaña que dice Request 1, dentro de loginCmsSoapBinding > loginCms y reemplazamos “?” por el contenido de nuestro archivo CMS.

Figura 6 - SoapUI - Request

The screenshot shows the SoapUI 5.6.0 interface. The main window displays the configuration for 'Request 1' in the 'Request 1' tab. The URL is set to 'https://wsaahomo.afip.gov.ar/ws/services/LoginCms'. The SOAP body is configured with the following XML structure:

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <wsaa:loginCms>
      <!-- Content of the loginCms element -->
    </wsaa:loginCms>
  </soapenv:Body>
</soapenv:Envelope>

```

The 'Request Properties' table is visible in the bottom left corner:

Property	Value
Name	Request 1
Description	
Message Size	3084

The bottom status bar shows 'SoapUI log http log jetty log error log wsrm... memory...'.

Una vez enviada la solicitud, la misma nos devuelve un token y un sign, dentro de la solapa “credentials”, que usaremos más adelante, cuando solicitemos generar una factura. Es importante que guardemos los datos que recibimos, ya que el servicio de AFIP no nos otorgará otro ticket de acceso debido a la existencia de un TA que continúa vigente.

Figura 7 - SoapUI - Response

The screenshot shows the SoapUI 5.6.0 interface. The main window displays the raw XML response for a SOAP request to the endpoint `https://wsaahomo.afip.gov.ar/ws/services/LoginCms`. The response is a `loginCmsResponse` containing a `loginTicketResponse` and a `loginCmsReturn`. The `loginTicketResponse` includes a `header` with source, destination, uniqueId, generationTime, and expirationTime, and a `credentials` section with token and sign values. The `loginCmsReturn` is an empty `CDATA` block.

Property	Value
Name	Request 1
Description	
Message Size	2666

Se pudo comprobar que los certificados fueron generados correctamente, por lo que procedemos a desarrollar el módulo de autenticación y autorización en Java.

Al ser una aplicación Spring de tipo Maven, buscamos en la documentación de Spring y encontramos que, para utilizar los servicios web SOAP, es necesario agregar dependencias en nuestro archivo pom.xml del proyecto, lo que generará los recursos necesarios para utilizar dicho servicio, como por ejemplo clases y métodos.

A esto hemos añadido dos perfiles en nuestro POM, uno para homologación y otro para producción, ya que no queremos utilizar el servicio de homologación en el ambiente de producción y vice versa.

Para utilizar correctamente los servicios, creamos los archivos WSDL de homologación y producción, en un archivo WSDL en la carpeta correspondiente para cada perfil y se lo enviamos por parámetro al plugin añadido (campo `schemaDirectory`).

```
<profiles>
  <profile>
    <id>test</id>
    <activation>
      <activeByDefault>true</activeByDefault>
    </activation>
  </profile>
</profiles>
```

```

</activation>
<properties>
  <wsdl>src/main/resources/wsdl/afip/test</wsdl>
  <jar.finalName>api_test</jar.finalName>
</properties>
</profile>
<profile>
  <id>production</id>
  <properties>
    <wsdl>src/main/resources/wsdl/afip/prod</wsdl>
    <jar.finalName>api_prod</jar.finalName>
  </properties>
</profile>
</profiles>
...

<plugin>
  <groupId>org.jvnet.jaxb2.maven2</groupId>
  <artifactId>maven-jaxb2-plugin</artifactId>
  <version>0.14.0</version>
  <executions>
    <execution>
      <goals>
        <goal>generate</goal>
      </goals>
    </execution>
  </executions>
  <configuration>

    <schemaLanguage>WSDL</schemaLanguage>
    <generatePackage>com.afip</generatePackage>
    <schemaDirectory>${wsdl}</schemaDirectory>
    <schemasIncludes>
      <schemasInclude>*.wsdl</schemasInclude>
    </schemasIncludes>
  </configuration>
</plugin>

```

De este modo, le indicamos a nuestra aplicación qué recursos debe generar y le enviamos la especificación del WSDL por parámetro, dependiendo del perfil que se encuentre activado en el momento de correr la misma.

Una vez agregado el plugin y los perfiles en el POM, debemos hacer Maven Install para que se generen dichos recursos.

Para poder utilizar los certificados en nuestra aplicación Java, es necesario cargar un archivo PKCS12.

El formato PFX o PKCS#12 es un formato binario para almacenar el certificado, los certificados

intermedios y la clave privada, todo en un único archivo encriptado. Los archivos PFX usualmente tienen la extensión .pfx o .p12.

Al estar trabajando con Java, es necesario cargar el certificado en un almacén de claves, también conocido como keystore.

Para establecer una conexión a un punto final SSL, se debe confiar en un certificado público, el cual debe estar en el almacén de claves. El mismo puede ser propio o de un emisor.

Para generar dicho archivo, utilizaremos el siguiente comando:

```
openssl pkcs12 -export -in MiCertificado.crt -inkey MiClavePrivada.key -out MiCertificadoP12.p12 -passin pass:nuestroPass -passout pass:nuestroPass
```

donde MiClavePrivada.key es la clave privada usada cuando se generó el CSR; MiCertificado.crt es el certificado .pem en formato CRT obtenido usando el WSASS y MiCertificadoP12.p12 será el nuevo archivo PFX (conteniendo al certificado y la clave privada).

Una vez obtenido el mensaje en formato CMS, a diferencia de lo que se probó con SoapUI, tenemos que codificarlo en Base64. Luego, procedemos a consumir el servicio de autenticación WSAA. Si todo es correcto, nos devolverá un objeto LoginCmsResponse, el que contiene las credenciales para poder consumir el WSN (token y sign).

A fines prácticos y de rendimiento, se optó por utilizar Redis para almacenar las credenciales, el cual es un motor de base de datos en memoria, NoSQL, basado en el almacenamiento en tablas de hashes.

Los datos que necesitamos almacenar, además del token y sign, es el tiempo de expiración de nuestro ticket de acceso. De este modo, sabremos si nuestras credenciales aún tienen validez al momento de usar el TA. Si no es el caso, se generará un nuevo mensaje CMS con los valores de fecha y hora correspondientes y se solicitará un nuevo ticket de acceso.

Para el caso de la autenticación, el WSDL de WSAA no nos generó las clases de los objetos que retorna el servicio, sino que devuelve todo en un único String, por lo que tuvimos que hacer uso de herramientas de Java para obtener los datos que necesitamos almacenar.

Estos datos los almacenamos instanciando un objeto, creado con anterioridad, que tiene los tres atributos previamente mencionados, persistiendo en Redis.

4.4.2 Web Service de Factura electrónica:

Llegado a este punto, ya podemos hacer uso del WSN de AFIP para generar facturas electrónicas, para lo cual, crearemos una clase que reciba como parámetro el objeto con las credenciales de autenticación y la operación de depósito o retiro, según corresponda. Utilizaremos los recursos generados por el WSDL de este servicio, el cual nos crea múltiples clases y métodos.

Antes de poder facturar, es necesario conocer el número de la factura que intentaremos generar. Para esto, hacemos uso de otro WSN destinado únicamente a consultar el número de la última factura generada. Este servicio se utiliza enviando un request SOAP con el siguiente contenido:

Figura 8 - XML SOAP para número de factura

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ar="http://ar.gov.afip.dif.FEV1/">
  <soapenv:Header/>
  <soapenv:Body>
    <ar:FECmpUltimoAutorizado>
      <ar:Auth>
        <ar:Token>string</ar:Token>
        <ar:Sign>string</ar:Sign>
        <ar:Cuit>long</ar:Cuit>
      </ar:Auth>
      <ar:PtoVta>int</ar:PtoVta>
      <ar:CbteTipo>int</ar:CbteTipo>
    </ar:FECmpUltimoAutorizado>
  </soapenv:Body>
</soapenv:Envelope>
```

El objeto Auth está compuesto por el token y sign devueltos por el WSAA; luego, indicamos el cuit contribuyente, el punto de venta y el tipo de comprobante.

Con esta información, podemos armar el objeto con todos los datos necesarios para generar la factura electrónica. Cabe aclarar que nuestro plugin, explicándolo resumidamente, mapea las clases generadas a los XML correspondientes para cada solicitud SOAP.

En el caso del WSN de factura electrónica, enviaremos un XML con la siguiente estructura (solo utilizamos los campos obligatorios):

Figura 9 - XML SOAP para facturación

```

<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ar="http://ar.gov.afip.dif.FEVL/">
  <soapenv:Header/>
  <soapenv:Body>
    <ar:FECAESolicitar>
      <ar:Auth>
        <ar:Token>string</ar:Token>
        <ar:Sign>string</ar:Sign>
        <ar:Cuit>long</ar:Cuit>
      </ar:Auth>
      <ar:FeCAEReq>
        <ar:FeCabReq>
          <ar:CantReg>int</ar:CantReg>
          <ar:PtoVta>int</ar:PtoVta>
          <ar:CbteTipo>int</ar:CbteTipo>
        </ar:FeCabReq>
        <ar:FeDetReq>
          <ar:FECAEDetRequest>
            <ar:Concepto>int</ar:Concepto>
            <ar:DocTipo>int</ar:DocTipo>
            <ar:DocNro>long</ar:DocNro>
            <ar:CbteDesde>long</ar:CbteDesde>
            <ar:CbteHasta>long</ar:CbteHasta>
            <ar:CbteFch>string</ar:CbteFch>
            <ar:ImpTotal>double</ar:ImpTotal>
            <ar:ImpTotConc>double</ar:ImpTotConc>
            <ar:ImpNeto>double</ar:ImpNeto>
            <ar:ImpOpEx>double</ar:ImpOpEx>
            <ar:ImpTrib>double</ar:ImpTrib>
            <ar:ImpIVA>double</ar:ImpIVA>
            <ar:FchServDesde>string</ar:FchServDesde>
            <ar:FchServHasta>string</ar:FchServHasta>
            <ar:FchVtoPago>string</ar:FchVtoPago>
            <ar:MonId>string</ar:MonId>
            <ar:MonCotis>double</ar:MonCotis>
            <ar:Iva>
              <ar:AlicIva>
                <ar:Id>short</ar:Id>
                <ar:BaseImp>double</ar:BaseImp>
                <ar:Importe>double</ar:Importe>
              </ar:AlicIva>
            </ar:Iva>
          </ar:FECAEDetRequest>
        </ar:FeDetReq>
      </ar:FeCAEReq>
    </ar:FECAESolicitar>
  </soapenv:Body>
</soapenv:Envelope>

```

Auth:

Token: identificador generado por WSAA

Sign: identificador generado por WSAA

Cuit: número de CUIT de la empresa

CantReg: cantidad de facturas, en nuestro caso, siempre será 1.

PtoVta: punto de venta

CbteTipo: tipo de comprobante (factura A o B)

Concepto:

DocTipo: tipo de documento del facturado

DocNro: número de documento del facturado

CbteDesde: número de comprobante que intentaremos generar

CbteHasta: número final de comprobante que intentaremos generar. Siempre será igual que CbteDesde

ImpTotal: importe total

ImpTotConc: importe neto no gravado

ImpNeto: importe neto gravado

ImpOpEx: importe excento

ImpTrib: importe tributario

Implva: importe IVA

MonId: identificador de moneda. Para pesos argentinos, "PES"

MonCotiz: cotizacion de la moneda informada. Para pesos argentinos, debe ser 1

AlicIva: es un objeto, el cual tiene:

id: es 5, indica alícuota de IVA 21%

BaseImp: base imponible para la alícuota indicada en Id

Importe: importe IVA liquidado

Una vez realizado todo esto, hacemos el request y obtenemos una respuesta con la siguiente estructura:

Figura 10 - XML SOAP Response

```

<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:ar="http://ar.qov.afip.dif.fev1/">
  <soap:Header/>
  <soap:Body>
    <FECAESolicitarResponse>
      <FECAESolicitarResult>
        <FeCabResp>
          <Cuit>long</Cuit>
          <PtoVta>int</PtoVta>
          <CbteTipo>int</CbteTipo>
          <FchProceso>string</FchProceso>
          <CantReq>int</CantReq>
          <Resultado>string</Resultado>
          <Reproceso>string</Reproceso >
        </FeCabResp>
        <FeDetResp>
          <FEDetResponse>
            <Concepto>int</Concepto>
            <DocTipo>int</DocTipo>
            <DocNro>long</DocNro>
            <CbteDesde>long</CbteDesde>
            <CbteHasta>long</CbteHasta>
            <Resultado>string</Resultado>
            <CAE>string</CAE>
            <CbteFch>string</CbteFch>
            <CAEFchVto>string</CAEFchVto>
            <Obs>
              <Observaciones>
                <Code>int</Code>
                <Msg>string</Msg>
              </Observaciones>
            </Obs>
          </FEDetResponse>
        </FeDetResp>
        <Events>
          <Evt>
            <Code>int</Code>
            <Msg>string</Msg>
          </Evt>
        </Events>
        <Errors>
          <Err>
            <Code>int</Code>
            <Msg>string</Msg>
          </Err>
        </Errors>
      </FECAESolicitarResult>
    </FECAESolicitarResponse>
  </soap:Body>
</soap:Envelope>

```

Dentro de FeCabResp, obtenemos el resultado de nuestra solicitud, el cual puede tomar 3 valores posibles, A de aprobado, R de rechazado y P de parcial.

Los resultados parciales se obtienen cuando se intenta generar más de una factura en una solicitud, por lo que, al generar las facturas individualmente, estaremos pendientes únicamente por resultados aprobados o rechazados.

Si la solicitud fue aprobada, guardaremos los siguientes datos:

CbteTipo: tipo de comprobante
CbteFch: fecha del comprobante
CbteDesde y CbteHasta: número del comprobante
CAE: código de autorización electrónico
CAEFchVto: fecha de vencimiento del CAE

A su vez, guardaremos también datos que enviamos al momento de realizar la solicitud, tales como importe total, importe neto gravado, importe IVA, número de documento (cuit/cuil), situación IVA del cliente, punto de venta y también el id de la operación generada por el cliente.

Acto seguido, se verifica si la operación se encontraba guardada en la base como factura pendiente y, en caso de ser necesario, se actualiza la información.

Si la solicitud fue rechazada, podemos encontrar el mensaje de error en el objeto Err dentro de Errors, o bien en el objeto Observaciones, dentro de Obs.

Para este caso, creamos un método para procesar los errores, el cual recibe el código de error, el mensaje y la operación del cliente.

Si el código de error es 10015, significa que el cliente no posee un cuit válido para AFIP, y sólo en este caso actualizamos la información del cliente y enviamos un mail a un administrador de la plataforma para que se pueda comunicar con dicho cliente. Para esto, utilizamos una librería de Spring para enviar emails, con los datos de configuración smtp proporcionados por la empresa.

Por defecto, si el código de error es distinto, sólo se loguea en la consola. Acto seguido, hacemos una llamada a otro método con el mensaje de error, la operación, un booleano "false" que indica que no se facturó dicha operación y la fecha del comprobante en null. Este método también se utiliza cuando se logra facturar de manera exitosa, con la fecha de comprobante correspondiente y el booleano mencionado en true.

En este punto, se pueden dar 4 casos:

Si la factura se encuentra en nuestra base de datos, en la tabla de facturas pendientes y fue aprobada, se obtiene dicha factura y se setea la fecha de facturación correspondiente.

Si la factura no se encuentra en nuestra base de datos y fue aprobada, se actualiza la información del cliente en caso de ser necesario.

Si la factura se encuentra en nuestra base de datos en la tabla de facturas pendientes y no fue aprobada,

significa que ya se intentó generar dicha factura, por lo que únicamente se actualiza el mensaje recibido por la petición (pueden fallar por varios motivos).

Si la factura no se encuentra en nuestra base de datos y no fue aprobada, se instancia un objeto con el id de operación, el mensaje de error, la fecha de facturación en null y se guarda en la base.

Esta aplicación utiliza una base de datos MariaDB con un manejador de migraciones Flyway, el cual se utiliza para garantizar la integridad de dichas migraciones.

Para este módulo, creamos dos migraciones. Una para generar una tabla con los datos de las facturas exitosas y otra para generar una tabla con las operaciones que están pendientes a facturar.

Una vez creadas las clases y habiendo mapeado los repositorios correspondientes, creamos los controladores que utilizaremos desde la aplicación cliente y administrador, las clases Dto y una clase para ejecutar tareas programadas.

Las clases Dto, las utilizamos para devolver la información que necesitemos a cada controlador. Por ejemplo, todos los objetos que persisten en nuestra base de datos tienen un id y timestamps de creación, modificación y eliminación lógica de los datos, los cuales, en la mayoría de los casos no queremos mostrar a los clientes o administradores, por lo que creamos una clase Dto únicamente con los atributos que queremos mostrar, utilizando un Mapper.

En la clase para ejecutar tareas programadas, creamos un método con una anotación propia de Spring, `@Scheduled`, a la cual se le indica un tiempo de ejecución, ya sea una fecha específica, franja horaria o tiempo de espera, entre otras.

Para esta clase, también utilizamos otra anotación, `@ConditionalOnProperty`, a la cual le indicamos el prefijo que debe encontrar en nuestro archivo de propiedades, el nombre de la propiedad y el valor booleano que debe encontrar para que se ejecuten las tareas de esta clase.

El método creado en dicha clase, es el que facturará las operaciones de manera automática. Para ello, escribimos una query nativa, la cual nos devuelve un Optional List de operaciones que sean depósitos o retiros, no hayan sido facturadas, sean de clientes residentes en Argentina, tengan cuit válido, la operación haya sido aprobada y que el timestamp de eliminación sea null.

Por último, si el Optional tiene una lista con operaciones, recorreremos las operaciones y, por cada una, obtenemos el ticket de acceso válido y procedemos a la facturación de la misma.

Para terminar con el desarrollo en nuestra API REST, resta crear los controladores, métodos y queries que utilizaremos para dar funcionalidad al módulo frontend, las cuales, por conveniencia y para que sea un poco más ordenado, se explicarán en el siguiente punto.

4.5 Desarrollo módulos frontend:

Este desarrollo se divide en dos módulos, Front Office y Back Office, los cuales son utilizados por los clientes y administradores de la plataforma respectivamente.

4.5.1 Módulo Front Office:

Como se aprecia en la imagen IMG-002, se agregó un botón en la tabla de Últimas operaciones, que permitiera al cliente descargar la factura. Este botón sólo aparece cuando el tipo de operación es Depósito o Retiro. Para ello, utilizamos una propiedad de Angular, evaluando cada tipo de operación de la tabla, para mostrar o no dicho botón.

Al hacer click, cambiamos el botón por un spinner de carga del mismo tamaño, propio de Bootstrap y pasamos por parámetro el id de la operación a un método creado en el archivo .ts (TypeScript), propio de Angular, que es donde se resuelve la lógica detrás del HTML de la página. Teniendo el id de la operación, buscamos la factura en una lista, aplicando un filtro; esto es para que no se generen nuevas peticiones a nuestra API REST si la factura ya fue descargada, y así hacer el proceso mucho más rápido. Para lograr esto, creamos una interfaz, que básicamente es la estructura de un objeto, el cual tiene un id y nuestra factura. Si encuentra la factura, se abre una nueva pestaña con nuestra factura en .pdf, caso contrario, se hace una petición a nuestra API REST, solicitando el documento. Si la petición es exitosa, se agrega en la lista y luego se imprime en pantalla.

Para la resolución de la petición, armamos un controlador que recibe el id de la operación de la cual queremos obtener la factura. Definimos una clase Business que se utilizará para manejar todo lo referido a las reglas de negocio de las facturas. En esta clase, creamos un método el cual recibe el id de la operación en cuestión y devuelve un objeto ResponseEntity de tipo InputStreamResource, siendo el tipo de dato de nuestra factura.

Como medida de seguridad, para resguardar la información de los clientes, obtenemos el cliente del Token enviado en la petición y luego buscamos la operación y la factura en la base con la información de dicho cliente, devolviendo un mensaje de error en caso de que la operación no exista (lo cual sería un indicativo de que se está intentando obtener una factura de otro cliente o la persona que hizo la petición no está logueada).

Si se encuentra la factura en nuestra tabla de facturas, enviamos la información a otro método. Este método, instancia un objeto de tipo `Map<String, Object>`, el cual tendrá un String como clave y un Object con la información para cada clave. Este objeto, es el que utilizaremos para mapear todos los datos de nuestra factura, haciendo uso de un template en formato .ftl y una librería para generar salidas de texto basado en plantillas, llamada Freemaker. El template, no es más que una plantilla de nuestra factura, la cual se completa con los datos necesarios, siendo estos la información del cliente, de la empresa y de la factura en cuestión.

Acto seguido, utilizamos una librería para convertir nuestra plantilla en PDF, llamada HtmlConverter, instanciamos el objeto InputStreamSource, el cual es un array de bytes; luego, instanciamos el objeto ResponseEntity con nuestro array de bytes, información de los headers de la petición y el estado de la petición HTTP (para peticiones exitosas, toma el valor 200), para después retornar el objeto.

Para que esto se lleve a cabo de manera exitosa, debe existir la factura en nuestra base de datos, la cual se genera en una tarea programada o manualmente a través del Back Office (este fue un nuevo requerimiento).

Si la factura no se encuentra, puede deberse a tres motivos: uno de ellos, es que, posterior a una determinada fecha, las facturas se hacían a través de un Ente externo; el segundo motivo, es que la factura

aún no se generó porque el proceso que las genera aún no se ejecutó; y por último, el tercer motivo corresponde a que la factura no se generó porque la petición a AFIP presenta errores, ya sea que el cliente no posee un cuit registrado en los padrones de la entidad o, aunque menos común, se produjo un error interno en el servidor de la entidad, por lo que hay que volver a hacer la petición.

Si la facturación falla, independientemente del motivo, la misma se cargará en la tabla de facturas pendientes, y habrá que realizarla de forma manual. Para lograr esto, se desarrolló una funcionalidad en una tabla creada en el Back Office.

4.5.2 Módulo Back Office:

En la imagen IMG-003, se puede ver un menú con la leyenda “Facturación”, el cual tiene un desplegable con dos opciones. Al acceder en el segundo menú, Facturas Pendientes, se redirecciona a la siguiente pantalla, la cual consta de dos tablas paginadas: una con las facturas pendientes, que posee id de factura pendiente, id de operación, id de cliente, mensaje de error que devuelve el servidor de AFIP, fecha de creación; y un botón con dos opciones: Facturar y Ver Saldos.

Figura 11 - Tabla de Facturas pendientes - BackOffice

Facturas pendientes				
Id	Operacion Id	Cliente Id	Mensaje	Fecha creación
446	338739	61540	Error interno de base de datos - Metodo FECAESolicitar	11/1/2021 11:53:34
445	338773	59199	Error interno de base de datos - Metodo FECAESolicitar	11/1/2021 11:53:33
444	338759	57999	Error interno de base de datos - Metodo FECAESolicitar	11/1/2021 11:53:33
443	338764	57200	Error interno de base de datos - Metodo FECAESolicitar	11/1/2021 11:53:32
441	338754	56707	Error interno de base de datos - Metodo FECAESolicitar	11/1/2021 11:53:31

Cuit no válido				
Id	Nombre de usuario	Nombre	Apellido	Cuit/cuil
57719	[REDACTED]	nombre_57719	apellido_57719	[REDACTED]

Si hacemos click en Facturar, el sistema intentará facturar esa operación nuevamente.

Figura 12 - Modal de confirmación de facturación

The screenshot shows a mobile application interface with a dark blue header and a light gray background. The header contains a menu icon, the text "Facturas pendientes", and a share icon. Below the header is a navigation bar with a back arrow and the text "Atrás". The main content area is titled "Facturas pendientes" and contains a table with the following columns: Id, Operacion Id, Cliente Id, Mensaje, and Fecha creación. The table lists five rows of data, all with "Error interno de base de datos - Metodo FECAESolicitar" in the message column. A modal dialog is overlaid on the table, titled "Facturar", with the text "¿Está seguro que desea facturar la operacion id: 338739?". The modal has two buttons: "Aceptar" (blue) and "Cancelar" (red). At the bottom right of the table, there is a pagination control showing "Filas por página: 10" and "1 - 10 of 435". Below the table is a section titled "Cuit no válido" with a table containing one row of data: Id (57719), Nombre de usuario (nombre_57719), Nombre (nombre_57719), Apellido (apellido_57719), and Cuit/cuil (57719).

Id	Operacion Id	Cliente Id	Mensaje	Fecha creación
446	338739	61540	Error interno de base de datos - Metodo FECAESolicitar	11/1/2021 11:53:34
445	338773	59199	Error interno de base de datos - Metodo FECAESolicitar	11/1/2021 11:53:33
444	338759	57999	Error interno de base de datos - Metodo FECAESolicitar	11/1/2021 11:53:33
443	338764	57200	Error interno de base de datos - Metodo FECAESolicitar	11/1/2021 11:53:32
441	338754	56707	Error interno de base de datos - Metodo FECAESolicitar	11/1/2021 11:53:31

Id	Nombre de usuario	Nombre	Apellido	Cuit/cuil
57719	nombre_57719	nombre_57719	apellido_57719	57719

Si la operación se factura correctamente, se abre un modal con algo de información de la factura, un botón para descargar la factura y otro para cerrar el mismo modal.

Figura 13 - Modal de facturación exitosa

The screenshot shows the same mobile application interface as Figure 12. The modal dialog is now titled "Operacion facturada con éxito" and contains the following information: "Nro Factura: 34720", "Tipo factura: B", and "Punto de venta: 1". The modal has two buttons: "Descargar factura" (red) and "Aceptar" (blue). The background table and pagination controls are the same as in Figure 12.

Id	Operacion Id	Cliente Id	Mensaje	Fecha creación
446	338739	61540	Error interno de base de datos - Metodo FECAESolicitar	11/1/2021 11:53:34
445	338773	59199	Error interno de base de datos - Metodo FECAESolicitar	11/1/2021 11:53:33
444	338759	57999	Error interno de base de datos - Metodo FECAESolicitar	11/1/2021 11:53:33
443	338764	57200	Error interno de base de datos - Metodo FECAESolicitar	11/1/2021 11:53:32
441	338754	56707	Error interno de base de datos - Metodo FECAESolicitar	11/1/2021 11:53:31

Id	Nombre de usuario	Nombre	Apellido	Cuit/cuil
57719	nombre_57719	nombre_57719	apellido_57719	57719

Uno de los errores que se ha observado, y como hemos mencionado anteriormente, es el error de cuit/cuil no registrado en los padrones de AFIP. En la tabla de facturas pendientes, en el campo de Mensaje, se puede observar claramente el error. En este caso, se debe comunicar con el cliente para modificar el número de cuit/cuil en caso que sea requerido. Para ello, tenemos la segunda tabla, la cual tiene

información básica del cliente con su número de cuit/cuil.

Figura 14 - Tabla de Facturas pendientes y clientes con cuit no válido – BackOffice

Facturas pendientes					
← Atrás					
Facturas pendientes					
Id	Operacion Id	Cliente Id	Mensaje	Fecha creación	
447	338853	62879	Factura B (CbteDesde igual a CbteHasta), DocTipo: 86, DocNro 2036818646 no se encuentra registrado en los padrones de AFIP.	12/1/2021 08:22:43	⋮
445	338773	59199	Error interno de base de datos - Metodo FECAESolicitar	11/1/2021 11:53:33	⋮
444	338759	57999	Error interno de base de datos - Metodo FECAESolicitar	11/1/2021 11:53:33	⋮
443	338764	57200	Error interno de base de datos - Metodo FECAESolicitar	11/1/2021 11:53:32	⋮
441	338754	56707	Error interno de base de datos - Metodo FECAESolicitar	11/1/2021 11:53:31	⋮
				Filas por página: 10	1 - 10 of 435 < >
Cuit no válido					
Id	Nombre de usuario	Nombre	Apellido	Cuit/cuil	
62879	lespinlabs@gmail.com	Lucas	Ormazabal	2036818646	

Si se diera el caso y se intenta facturar una operación de un cliente con un cuit/cuil erróneo, el sistema mostrará el siguiente mensaje. Para esto utilizamos un servicio de Toast de notificaciones.

Figura 15 - Toastr de Error - BackOffice

Cuit no válido					
Id	Nombre de usuario	Nombre	Apellido	Cuit/cuil	
62879	lespinlabs@gmail.com	No es posible facturar esta operación.		azabal	2036818646

En la segunda tabla, se listan de forma paginada los clientes con número de cuit/cuil no válidos por AFIP. Al hacer click en la fila del cliente, nos lleva a una pantalla para su edición, donde se puede cambiar el número de cuit/cuil en caso de que el mismo llegase a ser incorrecto o la entidad le hubiese proporcionado otro número si llegara a ser extranjero.


Figura 16 - Tooltip Editar cliente - BackOffice

Cuit no válido					
Id	Nombre de usuario	Nombre	Apellido	Cuit/cuil	
57719		nombre_57719	apellido_57719		

Si la operación es facturada de manera exitosa, se actualizan ambas tablas y por ende, no serán reflejadas las operaciones que se pudieron facturar ni los clientes que ahora tienen cuit/cuil válido.

Además, en los detalles de las operaciones de depósito y retiro de cada cliente, se agregó un botón que permitiera generar/descargar la factura de cada operación por separado.

Figura 17 - Tabla de Operaciones por cliente - BackOffice

Operaciones					
Id	Fecha y Hora	Tipo de operación	Moneda	Cantidad	Estado operación
338851	12/01/2021 08:02 AM	Depósito	Dolares	250.00 USD	 Manual

Filas por página: 50 1 - 1 of 1 < >

Al clicar en la fila de cada operación, se extiende un panel con los detalles de la misma. El botón fue agregado al final de la misma.

Figura 18 - Detalles de operación - BackOffice

Address label:	
Usuario Operador:	Lucas Ormazabal
Nombre cliente:	Lucas
Apellido cliente:	Ormazabal
Mail cliente:	lespinlabs@gmail.com
Monto original:	250.00000000000000 USD
Monto neto:	250.00000000000000 USD
Porcentaje Comisión Decrypto	0 %
Valor Comisión Decrypto:	0 USD
Valor Gasto Fijo:	0 USD
Porcentaje Gasto	0 %
Valor Gasto	0 USD
Ganancia Decrypto:	0 USD
Porcentaje Descuento Total	0 %
Descuento Total del Cliente (fijo):	0 USD
Usuario aprobador:	
Fecha/Hora aprobación:	12/01/2021 08:02 AM
Detalle:	





Descargar factura < >

Para obtener los datos de todas estas tablas, seguimos un estándar de estructura; creamos un controlador para las peticiones relacionadas a facturas del Back Office, y utilizamos el mismo Business creado para resolver las peticiones del Front Office.

4.6 Análisis y desarrollo de nuevos requerimientos:

En este punto, una vez deployado en los servidores de testing y habiendo corregido errores, se continuó con el desarrollo de los nuevos requerimientos por parte del cliente:

Para los usuarios del Front Office:

- Cada usuario, en el registro, debe poder seleccionar el tipo de usuario, que puede ser Persona Física o Persona Jurídica, y también su Situación IVA.

Para los usuarios del Back Office:

- Se debe poder visualizar el total de facturas emitidas, monto total facturado, monto total IVA y monto total gravado. Además, es necesario que se puedan exportar dichas facturas en un documento .csv con una serie de datos definidos.
- Se deben poder listar el total de facturas por cada cliente.

Front Office:

Se agregaron los combos Tipo de Usuario y Situación IVA en el registro de los clientes, mostrada anteriormente en la imagen IMG-001. Previo a este desarrollo, al registrarse un nuevo cliente a la plataforma, el mismo tenía el campo Situación IVA seteado por defecto en Consumidor Final, debido a que esa información no era relevante para el correcto funcionamiento de la plataforma.

Luego, se creó una migración para agregar una tabla con los tipos de clientes y otra para modificar la tabla de clientes, agregando la columna faltante, en este caso para identificar el tipo de usuario. Dicho valor es una Foreign Key a la tabla tipos de usuarios, el cual puede ser Persona física o Persona jurídica.

Back Office:

En el primer menú, Facturas Emitidas, tenemos la información que se aprecia en la siguiente imagen:

Figura 19 - Tabla de Facturas emitidas- BackOffice

Fecha	Tipo	Nro Comprobante	Factura	Cliente	Operacion	Total	IVA	Gravado
6/1/2021	B	40379	31023	59695	322323	99.6	17.28	82.31

Como se puede apreciar en la Figura 19, aquí se encontrarán tres cards que muestran el total facturado, el total IVA y el total gravado. A su vez, encontrarán un botón para exportar la información de las facturas en un documento .csv, el cual es similar a Excel, como así también una serie de filtros. Estos últimos se podrán utilizar para seleccionar el tipo de factura, que puede ser A o B; o mismo las fechas de facturación. Por último, en esta misma opción se encontrará un listado con las últimas facturas emitidas, las cuales podrán ser impresas con el botón que se encuentra sobre el margen derecho. Este mismo botón, puede ser utilizado también para facturar una operación que aún no haya sido procesada por el sistema que se

ha creado; por lo cual la acción del mismo dependerá de las acciones previas que se hayan tomado.

Los totales que se muestran sobre los tres cards, corresponden a la información de la tabla debajo; es decir, que en caso de que haya algún filtro aplicado, la tabla y los totales se verán modificados.

En la siguiente imagen, se puede ver que el total corresponde con los datos de la factura que coincidió con los filtros, la cual es la factura correspondiente a la operación que realizamos en un principio.

Figura 20 - Tabla de Facturas emitidas filtradas- BackOffice

Fecha	Tipo	Nro Comprobante	Factura	Cliente	Operacion	Total	IVA	Gravado
6/1/2021	B	40379	31023	59695	322323	99.6	17.28	82.31

Para representar todos estos datos, creamos una tabla paginada con un valor por default de 50 filas por página. Esto se logró devolviendo un objeto `CollectionPaginatedDto` y como tipo utilizamos un `Dto` creado previamente con los datos que queremos mostrar, mapeando cada factura como ya explicamos anteriormente.

En la siguiente imagen, se puede ver al archivo csv importado de dicha factura.

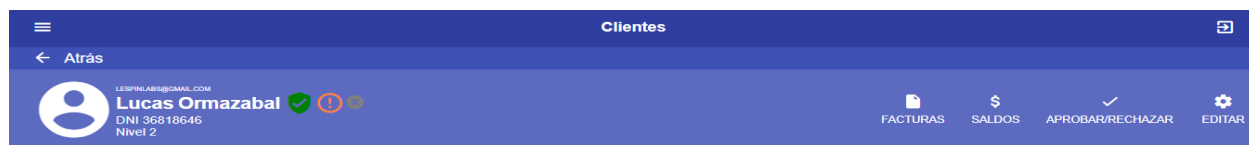
Figura 21 - Documento .csv exportado - BackOffice

	A	B	C	D	E	F	G	H	I	J	K	L
1	fechaComprobante	fechaVencimientoCae	numeroComprobante	importeTotal	tipoComprobante	importeNetoGravado	importeIva	operacion	cuitCuil	puntoVenta	id	
2	2021-01-06T03:00:00.000Z	2021-01-16T03:00:00.000Z	40379	99.6	B	82.31	17.28	{"id":322323,"cliente":{"id":59695,"situacionIva":{"description":"Consumido Final","id":5},"tipoOperacion":{"description":"Depósito","id":5}}	20368186466	1	31023	
3												

Otro de los nuevos requerimientos, fue poder listar las facturas de cada cliente.

Para ello, agregamos el botón "FACTURAS" en el header del perfil de los clientes, como se muestra en la siguiente imagen.

Figura 22 - Tabla de información de un cliente - BackOffice



Haciendo click en ese botón, nos redirecciona a una pantalla que solo tiene una tabla paginada con todas las facturas de ese cliente, con información solicitada de cada factura.

Figura 23 - Tabla de Facturas por cliente- BackOffice

Facturas										
Id	Cuit	Situacion IVA	Tipo factura	Sucursal	Numero factura	Tipo operación	Gravado	IVA	Total	Fecha
34721	20368186466	Consumidor Final	B	1	40382	Depósito	83.3	17.49	100.8	12/1/2021 24:00:00
34719	20368186466	Consumidor Final	B	1	40380	Depósito	83.3	17.49	100.8	12/1/2021 24:00:00

Filas por página: 10 1 - 2 of 2 < >

Al final, tenemos un botón que nos permite descargar cada factura de manera individual.

Por último, una imagen del formato de cada factura:

Figura 24 - Factura

ORIGINAL



de DECRYPTO S.A.S.

RESPONSABLE INSCRIPTO
DORREGO AV. 2079 Piso:4 Dpto:401
(1414) CABA
Caba

B

Cod. 006

Factura B

Número : 0040379

Fecha Emisión : 06-01-2021

CUIT : 23330005319

Ing.Brutos : 23330005319

Inicio de Actividad : 18-06-2018

Nombre: Ormazabal, Lucas **Período del Servicio :** 06-01-2021 al 06-01-2021

Domicilio : Test 1234 **Fecha Vto Pago :** 06-01-2021

Provincia : Ciudad Autónoma de Buenos Aires **Condición de Venta :** Contado

CUIL : 20368186466 **Condición de IVA:** Consumidor Final

Código	Detalle	Cantidad	Unidad	Precio \$	Bonif %	Total \$	Alic
	Servicios por cuenta y orden de cobros y pagos en criptomonedas : Volumen Negociado USD 250.00	1,00	UNIDAD	\$ 99.60	0,00 %	\$ 99.60	21%
	Servicio prestado en Ciudad de Buenos Aires con fecha de liquidación bancaria 06-01-2021						


Importe en Letras : noventa y nueve pesos con sesenta centavos

Consultas y reclamos
soporte@decrypto.la

CAE Nro : 71019906548090
Fecha Vto CAE : 16-01-2021

Página 1/1

Importe TOTAL : \$ 99.60



3371608676906000270259177473106202006302

5. Conclusiones:

A través de la implementación del facturador electrónico, la empresa puede ofrecer una factura electrónica con validez legal casi de manera instantánea, generada de forma automática, lo que conlleva a un aumento en la calidad de servicio ofrecido a sus clientes.

Puedo concluir que todo lo aprendido en la Universidad Nacional del Noroeste de la Provincia de Buenos Aires, en la carrera de Ingeniería en informática, me sirvió como una gran base para la realización de la práctica profesional y desempeño como desarrollador, el cual fue muy enriquecedor a nivel académico y personal, pudiendo incorporar abundantes conocimientos en el área de programación, bases de datos y análisis y diseño de sistemas; además, pude adquirir experiencia en encriptación, autenticación, consumo de servicios REST y SOAP, diseño web y diversos lenguajes de programación.

También puedo afirmar que Espin Labs es una empresa con una excelente cultura de trabajo y con proyectos desafiantes, conformado por personas muy profesionales que incentivan al desarrollo personal y profesional.

Como futuro profesional, espero seguir desarrollando la comunicación con profesionales de diferentes áreas, continuar con la investigación y adaptación a las nuevas tecnologías y mejorar los criterios para desarrollar soluciones mantenibles y escalables.

6. Bibliografía:

- [1] Afip. Arquitectura general [online]. Disponible en: <https://www.afip.gob.ar/ws/documentacion/arquitectura-general.asp>
- [2] Afip. Certificados [online]. Disponible en: <https://www.afip.gob.ar/ws/documentacion/certificados.asp>
- [3] Afip. Wsaa [online]. Disponible en: <https://www.afip.gob.ar/ws/documentacion/wsaa.asp>
- [4] Afip. Factura electrónica [online]. Disponible en: <https://www.afip.gob.ar/ws/documentacion/ws-factura-electronica.asp>
- [5] Afip. Wsass [online]. Disponible en: https://www.afip.gob.ar/ws/WSASS/WSASS_manual.pdf
- [6] Afip. Wsass manual developer [online]. Disponible en: <https://www.afip.gob.ar/ws/WSAA/WSAAmanualDev.pdf>
- [7] Afip. Wsaa especificación técnica [online]. Disponible en: https://www.afip.gob.ar/ws/WSAA/Especificacion_Tecnica_WSAA_1.2.2.pdf
- [8] Spring. Consuming a SOAP web service [online]. Disponible en: <https://spring.io/guides/gs/consuming-web-service/>
- [9] Baeldung. Maven profiles [online]. Disponible en: <https://www.baeldung.com/maven-profiles>
- [10] Oracle. Java Platform, Standard Edition (Java SE) 8. Disponible en: <https://docs.oracle.com/javase/8/javase-books.htm>
- [11] Oracle. MySQL 8.0 Reference Manual. Disponible en: <https://dev.mysql.com/doc/refman/8.0/en/>
- [12] SmartBear. Swagger. Disponible en: <https://swagger.io/docs/>
- [13] Redgate. Flyway. Disponible en: <https://documentation.red-gate.com/fd>
- [14] Apache. Freemarker. Disponible en: <https://freemarker.apache.org/docs/index.html>
- [15] Google. Angular. Disponible en: <https://angular.io/docs>