Universidad de Las Palmas de Gran Canaria

Escuela de Ingeniería Informática

Plataforma de datos de una aplicación FinTech

FinBook

Trabajo de Fin de Grado

Autor:

Raúl Lozano Ponce

Tutor:

José Juan Hernández Cabrera

Fecha de defensa:

23 de julio de 2020

Las Palmas de Gran Canaria, 8 de julio de 2020

Dedicado a mis padres, y a
mis hermanas por el apoyo incondicional

Resumen

En la actualidad, vivimos en una era tecnológica en la que diferentes elementos y aspectos que usamos en nuestro día a día, siguen una clara tendencia a digitalizarse, como es el caso de los libros o la música. Sin embargo, la digitalización de la factura no ha llegado con fuerza, al menos en Europa.

En muchos países como es el caso de México, Chile o Brasil, llevan dos décadas usando facturas electrónicas, y una de las razones por las que se ha llevado a cabo esta digitalización es para reducir el fraude. Actualmente, en España, el fraude es bastante alto, y es por eso, entre otros aspectos, que es inminente la digitalización de la factura.

Para hacer real esta digitalización, se requiere de aplicaciones que permitan manejar una gran cantidad de facturas, y que sean capaces de almacenar y distribuir las facturas relativas a cualquier periodo temporal. Además, cabe destacar la importancia de la seguridad en este tipo de aplicaciones, ya que se manejan datos que en ocasiones, disponen de cierta confidencialidad.

Sin embargo, pese a que esta digitalización esté por venir, las tecnologías utilizadas actualmente con un uso habitual, han quedado desfasadas para cumplir estos objetivos, dando así paso a una arquitectura muy conocida, pero que realmente aún no se ha explotado todo su potencial: *Big Data*.

Es por eso que, haciendo uso de una arquitectura *Big Data*, se ha decidido estudiar qué tan viable es el desarrollo y el uso de una aplicación con estas características y, de ser viable, qué especificaciones técnicas tendría.

Abstract

Nowadays, we live in a technological era in which different elements and aspects that we use in our day to day, follow a clear tendency to be digitized, as it is the case of books or music. However, the digitisation of the bill has not arrived with any force, at least in Europe.

In many countries, such as Mexico, Chile or Brazil, electronic invoices have been used for two decades, and one of the reasons why this digitalization has been carried out is to reduce fraud. Currently, in Spain, fraud is quite high, and that is why, among other aspects, the digitalization of the invoice is imminent.

To make this digitalization a reality, applications are required that allow for the handling of a large number of invoices, and that are capable of storing and distributing invoices related to any time period. In addition, it is important to emphasize the importance of security in this type of application, since it handles data that sometimes have a certain degree of confidentiality.

However, despite the fact that this digitalization is still to come, the technologies currently used with a common use, have become outdated to meet these objectives, thus giving way to a very well known architecture, but that really has not yet exploited its full potential: Big Data.

That is why, making use of a Big Data architecture, it has been decided to study how viable the use of an application with these characteristics is and, if it is viable, what technical specifications it would have.

Índice general

Índice de figuras

Índice de Códigos

Introducción

Vivimos en una época en la que existe una clara tendencia a digitalizar distintos elementos de casi cualquier ámbito: la música, los libros, etc. Esto nos lleva a pensar en la situación actual de la factura, ya que es un elemento presente de manera abundante en nuestras vidas.

Cuando se habla de una factura, lo habitual es pensar en un documento físico, y en una primera instancia, no parece resultar algo extraño. Sin embargo, con la constante globalización, automatización e informatización en la que estamos sumergidos, deja bastante claro que el ámbito de la gestión de facturas está bastante anticuado respecto a otros, aún disponiendo ya de múltiples herramientas que automatizan procesos y centralizan los datos. Además, no se trata únicamente de una cuestión de velocidad o automatización: una gestión tradicional genera una gran cantidad de gastos asociados.

Teniendo en cuenta estos detalles, no es descabellado pensar que en unos años la factura pueda estar completamente digitalizada, lo cual implicaría su uso por parte de agencias tributarias. Está claro que estas facturas deben ser manejadas con programas informáticos, pero en el caso de las agencias tributarias, el número de facturas que se manejan es demasiado grande para las tecnologías actuales de uso habitual. La principal cuestión a plantear es la siguiente: ¿Qué tan lejos estamos de poder desarrollar un sistema capaz de manejar semejante cantidad de facturas con la tecnología actual?

Capítulo 1

Hacia la digitalización de la factura

"La factura electrónica (e-factura) consiste en una modalidad de factura en la que no se emplea el papel como soporte para demostrar su autenticidad, sino un soporte electrónico en el que se recogerá la información relativa a una transacción comercial y sus obligaciones de pago y de liquidación de impuestos, además de otros requisitos en función de las obligaciones concretas del país en que utilice" (Sanz, 2008).

La digitalización de las facturas es un proceso en el que simplemente se sustituyen las facturas físicas tradicionales por facturas electrónicas, sin que esto afecte a la validez legal de las mismas. En este proceso, se generan documentos tributarios que contienen la misma información que una factura tradicional. Estos documentos, a diferencia de las facturas originales, pueden ser enviados de forma telemática, es decir, utilizando técnicas de la telecomunicación y la informática. Estos medios de comunicación deben asegurar la autenticidad e integridad del documento, y, para disponer de una mayor seguridad, las facturas se suelen firmar digitalmente haciendo uso de la firma digital para asegurar al destinatario de la factura que esta no ha sido modificado su contenido. Las facturas, tanto electrónicas como en papel, permiten justificar la compra o la venta de un producto o servicio, siempre cumpliendo los requisitos legales impuestos por las autoridades tributarias de cada país (Bartholomew, 2005). La idea de la digitalización es agilizar los procesos de tramitación (Foryszewski, 2006; Rombel, 2007), entre otras muchas ventajas, y, aunque actualmente no hay un estándar acerca del formato de estas facturas digitales, se hace mucho uso de formatos *Portable Document Format*

(PDF), si las facturas son simples escaneos de las originales; o incluso en *Extensible Markup Language* (XML), en las cuales la información de la factura está escrita mediante un formato estandarizado (Ortega y Cinca, 2009).

El uso de estas facturas dispone de una serie de ventajas, como son el aumento de velocidad a la hora de tramitar, la disminución de la posibilidad de cometer un error humano, disminución del tiempo de espera en el envío de la factura, ahorro económico significativo (El Hani, 2001; Berez y Sheth, 2007; Haq, 2007), e incluso poder realizar un cambio en la factura sin que suponga un gran coste adicional (Ortega y Cinca, 2009).

Pese a que con los años, la digitalización ha llegado a muchos países y en muchos otros es inminente, este cambio en sus inicios no fue rápidamente adoptado por las empresas. Sobre el año 2008, en España, la factura electrónica tan solo ha sido adoptada por un 8 % de las empresas españolas, hecho que impide un ahorro en costes de 15.000 millones de euros al año, importe equivalente a un 1,5 % del Producto Interior Bruto (PIB) (Molina, 2008). Este dato resulta extraño, ya que como se ha comentado, incluso en el año 2008 ya vivíamos en un mundo completamente globalizado en el que cada vez más se disponían de nuevos servicios, y se aceptó el uso de estos vía Web con la aparición de la Web 2.0 (Ortega y Cinca, 2009).

1.1. Beneficios de la digitalización de las facturas

La digitalización de las facturas conlleva un gran número de beneficios que deberían ser considerados para así, que el estado de esta digitalización sea global:

- Ahorro económico: tal vez es uno de los beneficios más claros. Está claro que el hecho de sustituir las facturas tradicionales de papel por las digitalizadas supondría un ahorro masivo de gastos en papel, tinta para imprimirlas, o incluso los gastos de envío. "Los costos relacionados con la impresión, el franqueo, las facturas correspondientes con las órdenes de compra y la presentación de facturas salientes ascienden a un promedio de $11. No usar papel le permite reducir estos costos en casi un 60 %, que hasta ahora no se podía reducir" (Christophe, 2017).
- Ahorro de espacio físico: tal vez una factura individual no ocupe mucho espacio, pero con el paso de los años, si una empresa realiza muchas

transacciones, las facturas las tienen que ir almacenando, llegando incluso a tener que guardarlas en carpetas o archivadores en grandes armarios para tenerlas bien ordenadas. Sin embargo, si las facturas son digitales, sí que supondría un gasto en espacio de almacenamiento en una unidad de almacenamiento de datos, pero no de espacio físico.

- Ahorro de tiempo: existen diferentes fuentes de ahorro de tiempo dadas por la digitalización de las facturas. Una de ellas, que está relacionada con el punto anterior, es el tiempo ahorrado en la búsqueda de alguna factura. El tener que buscar una factura específica entre muchos archivadores llenos de ellas es evidente que supone un costo mayor de tiempo que buscarla en un sistema de almacenamiento digital haciendo uso de filtros, y eso sin tener en cuenta que en el almacenamiento físico se puede traspapelar alguna factura, lo cual supondría un costo de tiempo aún mayor. Otra fuente de ahorro de tiempo es el relativo al tiempo de espera desde el momento en el que se genera la factura hasta que le llega al receptor, lo cual supone un retraso en el cobro de los productos o servicios vendidos.

- Validez legal: para que una factura electrónica sea válida, deben estar aprobadas como tal por las autoridades tributarias de cada país. Una vez en un país, este tipo de facturas han sido aprobadas, disponen de la misma validez legal que una factura tradicional impresa en papel. Es por eso que en este sentido, no supondría un problema el uso de este tipo de facturas.

- Aumento de la productividad: una vez obtenido un sistema capaz de tratar estas facturas digitales y que sean capaces de realizar las operaciones requeridas por cada empresa, la productividad de la empresa aumentaría significativamente, ya que el trabajo que antes realizaba parte del personal de forma manual y repetitiva en la que ocupaban toda su jornada laboral, ahora es realizada por este sistema en muy poco tiempo. De esta manera, se puede aprovechar este personal para realizar otras tareas importantes dentro de la empresa.

- Disminución de errores: está claro que los seres humanos no somos perfectos, y que tendemos a cometer errores, y con más frecuencia si se trata de la realización una tarea tediosa y repetitiva. Estos fallos son traducidos a gastos económicos por parte de la empresa.

- Sostenibilidad ecológica: la cantidad de papel que se ahorraría si se digitalizaran todas las facturas, supondría una clara reducción de la

fabricación del mismo, lo cual se traduce en un reducción de la contaminación atmosférica y de la deforestación.

1.2. Tipos de facturas

Existen dos posibles soluciones a la hora de digitalizar facturas: las facturas con un formato estructurado, que son aquellas que están representadas en ficheros XML, y que permiten su tratamiento y búsqueda de información de forma automatizada. Debido a esta facilidad, son el formato que permite una gestión de las mismas más rápida y eficiente. El otro tipo de facturas son las que están diseñadas con un formato no estructurado, que son aquellas compuestas por una imagen de la factura original escaneada en formato PDF. La automatización en este tipo de facturas es complicada, ya que la disposición de los campos en la imagen pueden ser distintos entre facturas, y pese a que existan maquinarias capaces de extraer información de estos escaneos, puede inducir a producir más errores (Cecarm, 2015).

Código 1.1: Ejemplo de Factura

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<cfdi:Voucher
        Concept="Bill of a eating"
        Currency="euro"
        Date="Tue Apr 14 00:26:05 BST 2020"
        Location="Calle Las Pitas 6"
        SubTotal="128.67024903648075"
        TaxRate="1.0"
        Total="128.67024903648075"
        Type="income"
>
        <cfdi:Transmitter
                IssuerName="Restaurante Nito's"
                IssuerRFC="1000002"
        />
        <cfdi:Receiver
                ReceiverName="Liam Coates"
                ReceiverRFC="3000423"
        />
</cfdi:Voucher>
```

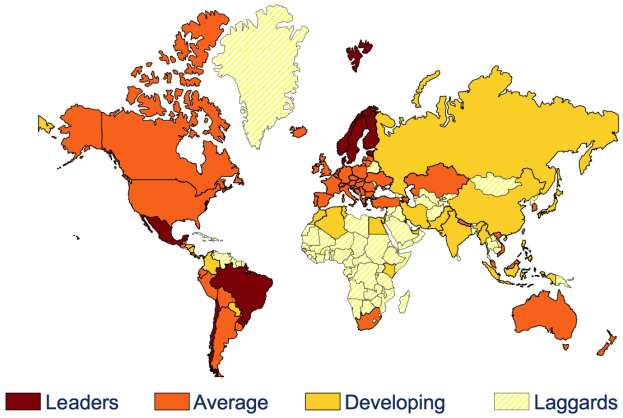

Figura 1.1: Estado de la digitalización de la factura en el mundo. Koch, Bruno. (2013). International E-Invoicing Market 2013. [Figura]. Recuperado de https://www.billentis.com/einvoicing_ebilling_market_overview_2013.pdf.

El código visualizado en el Código 1.1 es un ejemplo de una factura muy sencilla con un formato estructurado, ya que está representada en un formato XML y, como se puede apreciar, la lectura y obtención de datos es muy rápida ya que sencillamente se trata de un conjunto de información etiquetado y dispuesto en forma de texto. Como se puede apreciar, se diferencia claramente el apartado de la información de la propia factura, la información acerca del emisor, y la información del receptor.

1.3. Estado de la digitalización

Como se puede ver en la Figura 1.1, pese a la avanzada era tecnológica que estamos viviendo, la digitalización de la factura no está muy extendida por el mundo. Se puede apreciar claramente un mayor avance en el continente Americano, debido al alto porcentaje de fraude fiscal que se produce en estos países. Cabe destacar también de la presencia de los países europeos, liderados por los países de la península escandinava. A diferencia de los países Americanos, que han tomado la iniciativa de esta digitalización en la década pasada, Europa lo está haciendo en esta presente década.

1.3.1. México

Cabe destacar el caso de México, ya que es el modelo que se ha tomado como referencia. El Servicio de Administración Tributaria (SAT) ha destacado en hacer uso de las tecnologías para innovar en distintos procesos tributarios con el fin de ofrecer respuestas oportunas a los contribuyentes y aumentar la cobertura de servicios, y es el caso del uso de la factura electrónica (Barreix y cols., 2018).

Sobre el año 2004, comenzó la Comprobación Fiscal Digital (CFD) a través de proveedores de servicios de generación y envío de comprobantes fiscales digitales en México. Esto supuso que el país fuera uno de los pioneros en el uso de este tipo de facturas, lo que lo hace un referente en ese sentido. Las dificultades operativas que presentaban la facturación en papel tanto para el SAT como para los contribuyentes, motivaron el cambio de la factura tradicional, a la que se le denominó: Comprobante Fiscal Digital. Esta conversión supuso el poder asignar ciertos atributos de seguridad que en papel no se podía. Se decidió que estas facturas debían estar en formato XML, que desde aquel entonces se entendía como un lenguaje universal. Además, gracias al uso de firmas electrónicas y certificados de sello digital, haciendo uso de un firmado electrónico, se podía dotar de la misma integridad y autenticidad que una factura tradicional a estos archivos. En esta etapa, existían dos modos para emitir los comprobantes fiscales: uno en el que se haga con medios propios, es decir, el contribuyente emitía sus comprobantes fiscales sin ayuda de un tercero; y la otra consistía en utilizar los servicios de un proveedor de servicios de generación y envío de CFD (Barreix y cols., 2018).

Fue en el año 2011 cuando llegó a México el Comprobante Fiscal Digital por Internet (CFDI) sellado y entregado al SAT por un proveedor de certificación y a partir de 2014, pasó a ser el esquema de comprobación sobre el que opera el SAT, y además de uso obligatorio para todo aquel que quiera deducir gastos y acreditar impuestos. Los principales cambios producidos con la creación del CFDI fueron en el proceso de emisión y certificación: una vez el contribuyente genera la factura electrónica y la firma digitalmente, la debe remitir a un proveedor de certificación de CFDI para que valide el documento, le asigne el folio único, y lo timbre con la firma digital del SAT, y una vez se ha realizado este proceso, se devuelve la factura al contribuyente. los proveedores de certificación de CFDI son empresas autorizadas por el SAT que cuentan con una infraestructura tecnológica capaz de certificar las facturas digitales (Barreix y cols., 2018).

1.3.2. Unión Europea

Desde hace años, en la Unión Europea se promueve la digitalización de las facturas (e-factura). El primer paso llegó en 2014, que se estableció a nivel continental que toda factura superior a 5.000 euros que emitiesen las empresas al sector público, deberían llevar formato electrónico. En el caso de España, esta medida se transpondría a la legislación española al año siguiente. Esta obligación se extiende a terceras compañías subcontratadas por una principal que también tuviera tratos con el sector público como ayuntamientos, Gobierno Central, entre otros. En estos casos, la cuantía de la factura también debía ser superior a 5.000 euros. Como consecuencia de este cambio, la factura digital comenzó a generalizarse al menos entre los proveedores y el sector público. En 2018 se emitieron solamente en España casi 182 millones de facturas digitales, un 15 % más que en 2017 (Cabrera, 2019a).

Esta obligación de emitir facturas electrónicas no es obligatoria entre empresas o entre empresas y particulares. De hecho, para que una empresa privada pueda emitir una factura digitalizada a otra necesita "un consentimiento expreso" de la firma receptora de la misma. No obstante, la Unión Europea sigue interesada en promover este cambio en el ámbito entre empresas, y el movimiento de algunos países acelerará este proceso. En el caso de Italia, desde el 1 de enero de 2019, el Gobierno ha obligado a emitir en formato electrónico a todas las empresas del país, y en cuanto a Portugal, por ahora ha marcado un calendario de transición (Cabrera, 2019a).

En resumen, la globalización de la factura electrónica es inminente ya que con más frecuencia, los países que aún no hagan uso de la factura digital no tendrán más remedio que pedir facturas electrónicas a empresas de otros países que sí se hayan adaptado a este cambio, ya que así está preparado su circuito administrativo. Es por esto que para facilitar estos trámites, se han aprobado varios formatos para su uso estandarizado de obligatoria aceptación para las administraciones europeas (Cabrera, 2019b).

1.3.3. España

En el caso de España, cabe destacar el caso de País Vasco, que desde 2019 impone la obligación de emitir facturas digitales a todo tipo de operaciones entre compañías, para así poder acabar con el fraude en el comercio minorista y hacer visible el rastro de las operaciones. De cara a 2021, se prevee la implantación de un control de la facturación por parte de las administraciones

tributarias, y esta medida estaba prevista también en el ámbito estatal en el anteproyecto de ley de medidas de prevención y lucha contra el fraude fiscal. Este control tiene dos objetivos: evitar el fraude de facturación aprovechando el software que permite que las facturas cumplan una serie de requisitos; y, en el proceso de facturación, registrar un identificador único de la factura que deberá ir impreso en la misma y que permitirá su seguimiento por parte de la administración, e incluso la capacidad de autentificar la factura por parte del receptor. Además, las facturas deberán enviarse a las administraciones tributarias telemáticamente. Esto no implica necesariamente que las facturas sean electrónicas estructuradas, pero esta medida, de manera indirecta, hará que se incremente el uso de este tipo de facturas. Pese a las intenciones mencionadas, aún no ha sido publicado un calendario oficial para la transición para las empresas privadas: tan solo lo ha publicado País Vasco, cuya fecha objetivo es mediados de 2020. Todos están de acuerdo en comenzar por las personas jurídicas, ya que disponen de una mayor adaptación de sus sistemas de facturación, y tal vez se pueda dejar un mayor plazo a los autónomos (Cabrera, 2019b).

En conclusión, son tantos los beneficios que aporta la digitalización de las facturas y la cantidad de países que se están sumando a esta digitalización, que es inminente su uso en todo el mundo y probablemente, en los próximos 10 años, su uso esté completamente extendido al menos en Europa.

1.4. Aplicaciones

Con la llegada de la digitalización de las facturas, uno de los grandes beneficios es la fácil automatización de una gran cantidad de procesos costosos y repetitivos que actualmente realizan seres humanos. El hecho de que los responsables de estas tareas sean humanos conlleva a que la realización de los mismos suponga un gran coste temporal, e incluso que la cantidad de errores cometidos sea enorme.

1.4.1. Liquidación del IVA

Existen tareas de esta naturaleza que son esenciales en el mundo de la economía, y una de ellas es la liquidación del Impuesto de Valor Añadido (IVA). Según la Real Academia Española (RAE) (*REAL ACADEMIA ESPAÑOLA: Diccionario de la lengua española*, 2008), el IVA es el "impuesto

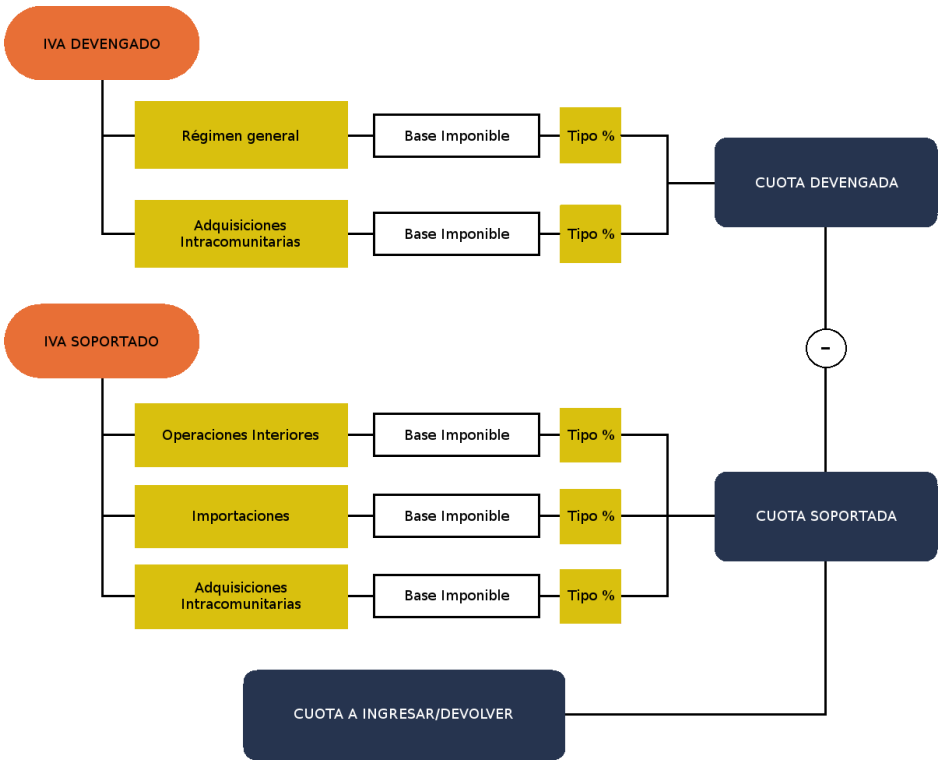

Figura 1.2: Cálculo del IVA Devengado y Soportado.
INEAF. (2013). IVA devengado e IVA soportado. [Figura]. Recuperado de https://www.ineaf.es/divulgativo/sistema-tributario/iva/liquidacion-y-contabilizacion-del-iva.

indirecto sobre el consumo y la prestación de servicios empresariales o profesionales, repercutible en cada una de las fases del proceso económico". La liquidación del IVA consiste en recaudar el impuesto para su ingreso en la Agencia Tributaria. Para llevarla a cabo, hay que calcular la diferencia entre el IVA devengado, que es el que se aplica en las facturas emitidas; y el soportado, que es el que paga en las facturas de gastos necesarios para la actividad. Si la suma del IVA devengado es superior al soportado, se generará un importe a pagar en un plazo establecido. Por otro lado, si el IVA de gastos es superior al de ingresos, se generará un IVA negativo a compensar. En el caso de ser negativo, se puede optar por acumularlo hasta durante 4 años (Fernández, 2014).

Como se puede ver en la Figura 1.2, la liquidación del IVA se calcula restando el IVA devengado al soportado, calculados obteniendo los impuestos aplicados sobre la base imponible.

1.4.2. Lucha contra el fraude fiscal

Como ha sido explicado antes, la liquidación del IVA es un proceso bastante importante y sobretodo laborioso, porque para realizar dicho proceso, se debe tener en cuenta cada una de las empresas del país, cada una de las facturas que manejan dichas empresas, los impuestos aplicados a cada una, etc. No debemos olvidarnos de la gran cantidad de fraude fiscal que se lleva a cabo anualmente debido a esta complejidad, debido a que, como es de esperar, muchos intentan pagar la menor cantidad de impuestos posibles, ya que consideran que es injusto tener que pagar una cantidad del dinero que han ganado esas empresas por "mérito propio".

"El modelo de facturación electrónica en el mundo busca la optimizar los recursos con que cuenta las empresas bajando costos de facturación y agilizando los procesos internos de las administraciones tributarias, pero el mayor impacto que se espera de este modelo es la reducción del flagelo de la evasión". La implantación de esta tecnología ha permitido a los gobiernos "mejorar la recolección de información sobre las transacciones económicas, retener impuestos y cruzar datos para fortalecer la fiscalización" (Mogollón y Bedoya López, 2019).

La temprana implantación de la factura digital en países latinoamericanos como México o Chile ha sido en parte para reducir este fraude fiscal, y es que gracias al uso de esta tecnología, se ha pasado de una tasa del 35 % al 20 %, aumentando el recaudo de impuesto de renta y del IVA. Cabe destacar que los países mencionados, también realizan "programas y campañas fiscales para fomentar el cumplimiento voluntario y tener actualizado el registro federal de contribuyentes como un mecanismo para combatir la evasión fiscal", ya que "estas campañas buscan que el contribuyente vaya de a mano con la administración tributaria en la asesoría que requieran para el pago de los tributos" (Mogollón y Bedoya López, 2019).

En cuanto al caso de España, el fraude fiscal "genera una bolsa gigantesca de dinero negro que cada año escapa al control de las autoridades y no contribuye al sostenimiento de muchos servicios públicos indispensables, como escuelas y hospitales. El consenso de los analistas dice que la economía sumergida en España supone un 20 % de su PIB, es decir, más de 240.000 millones de euros en estos momentos. Mientras tanto, algunos expertos dicen que el fraude fiscal podría llegar al 5 % del PIB, unos 60.000 millones de euros anuales". Estas cifras no dejan dudas de que se debe actuar lo antes posible, e implementar un método para controlar este fraude (Cabrera, 2019a).

"Una de las formas de reducir esta lacra es hacer más transparente la información financiera. Para ello, el Gobierno español ha promovido proyectos como el Suministro Inmediato de Información (SII), el llamado "IVA online", que obliga a las grandes y medianas empresas (a partir de seis millones de euros de ingresos anuales) a comunicar de forma telemática a Hacienda la información fiscal de las facturas emitidas y recibidas." (Cabrera, 2019a).

El siguiente paso es llevar esta digitalización a todo el tejido empresarial, lo que supondría otro golpe al fraude y a la competencia desleal, además de la reducción de falsificaciones de documentos, debido a que cada factura puede contener sellos y certificados digitales que la hacen única (Cabrera, 2019a).

Pese a la existencia del mencionado "IVA online", aun falta un gran camino por recorrer para reducir este fraude. El sistema actual depende de que las empresas liquiden correctamente el importe del IVA y lo envíen a las autoridades fiscales. El IVA se liquida durante un período fijo (cada mes o cada trimestre), y existe una desalineación en las fechas que cuentan (fecha de la factura o de la contabilización) aumentan en gran medida la complejidad. Esto hace que el control de los datos del IVA sea problemático, ya que cada empresa mantiene sus propios libros de contabilidad y liquidan el IVA de manera diferente, de una forma no centralizada. Está claro que hay que mejorar el sistema, y una posible solución que va de la mano de la digitalización de las facturas es la liquidación automatizada del IVA. Gracias a esta medida, se podrá reducir la rigidez, la complejidad y la resistencia al fraude a la hora de reclamar impuestos (Søgaard, 2018).

Para automatizar la liquidación del IVA, se hará uso de una plataforma digital de facturas como única fuente fiable para la liquidación y presentación de informes de IVA. Es evidente que al liquidar el IVA, las facturas seguirán cobrando un papel esencial aún con esta automatización. En resumen, la idea es poder almacenar todas las facturas en un sistema compartido por todas las empresas. Sobre este conjunto de datos, se ejecutan periódicamente procesos que automaticen la liquidación de este impuesto, y que además, lo enrutan con las autoridades fiscales (Søgaard, 2018).

Este sistema puede ayudar a evitar situaciones en las que: los informes del IVA se basan en negocios individuales para liquidar correctamente los importes del IVA, las declaraciones del IVA se envían a tiempo discreto, o en las que se producen envíos de IVA incorrectos o fraudulentos (Søgaard, 2018).

Como se puede ver en la Figura 1.4, que muestra el proceso actual de la liquidación del IVA, las autoridades fiscales reciben la información sobre las

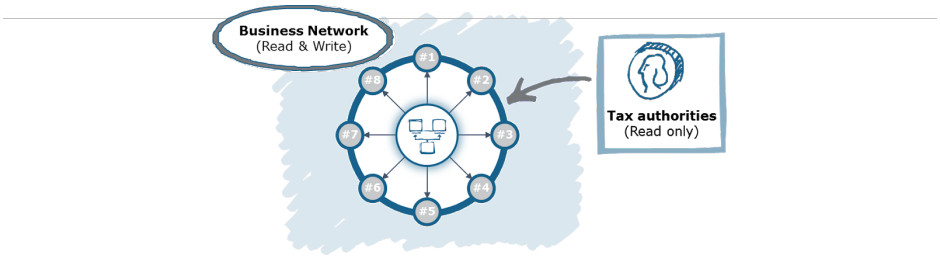

Figura 1.3: Las redes empresariales funcionan como entornos semicerrados donde los participantes comparten y validan datos.
Søgaard, Jonas Sveistrup. (2018). [Figura]. Recuperado de https://www.researchgate.net/publication/328190371_How_business_blockchain_can_enable_automatic_and_near_real-time_VAT_settlement.

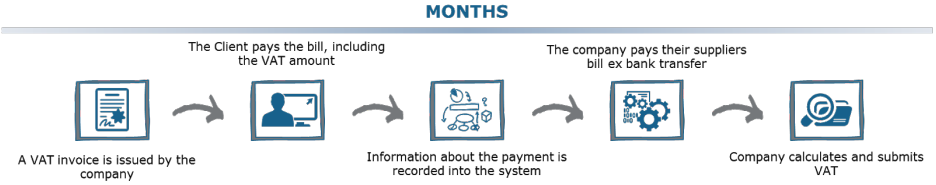

Figura 1.4: Proceso actual de la liquidación del IVA.
Søgaard, Jonas Sveistrup. (2018). [Figura]. Recuperado de https://www.researchgate.net/publication/328190371_How_business_blockchain_can_enable_automatic_and_near_real-time_VAT_settlement.

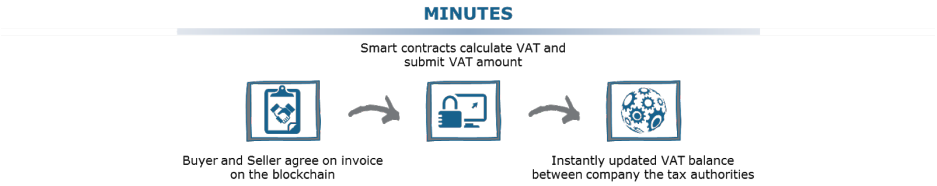

Figura 1.5: Solución propuesta para la liquidación del IVA. Søgaard, Jonas Sveistrup. (2018). [Figura]. Recuperado de https://www.researchgate.net/publication/328190371_How_business_blockchain_can_enable_automatic_and_near_real-time_VAT_settlement.

transacciones semanas o meses después de su creación y este retraso, afecta a la presentación de informes y la detección del mencionado fraude. Esto produce que para las empresas, las ganancias reembolsables deben calcularse y documentarse antes de presentarlas a las autoridades fiscales. Esto produce que en muchos casos, las empresas renuncien a dichas ganancias solo por el hecho del gran costo que supone poder obtenerlo (Søgaard, 2018).

Sin embargo, en la solución propuesta en la Figura 1.5, tanto el comprador como el vendedor informarán la misma información del IVA, basado en facturas validadas alineadas y de dos puntos. Todos los procesos, desde la última factura subida hasta el cálculo de la liquidación del IVA, se realiza automáticamente y se almacena digitalmente con un seguimiento claro y transparente (Søgaard, 2018).

En resumen, una única fuente fiable de datos tiene un alto potencial para una optimización del proceso, y produce una gran reducción en los costos, y este sistema permite a las empresas:

- Crear un seguimiento de auditoría verificado: un registro verificado y compartido puede servir como un seguimiento de auditoría confiable que puede beneficiar y automatizar la contabilidad, la reconciliación y la auditoría.
- Configurar una red que pueda interconectarse con los sistemas *Enterprise Resource Planning* (ERP) actuales: el sistema funcionaría como una solución compartida que conecta los sistemas actuales dentro de cada empresa.
- Crear un conocimiento adicional del proceso: análisis de posibles creaciones de valor mediante la operación de los procesos financieros actuales a través de esta solución.

- Formar una plataforma de financiamiento comercial: explorar el potencial para usar la red como plataforma comercial para la oferta y la demanda de bienes y servicios o incluso, activos y pasivos. (Søgaard, 2018)

Capítulo 2

Finbook

Para la realización de este Trabajo de Fin de Grado se ha realizado FinBook, una aplicación que pueda explotar el futuro auge de la digitalización de las facturas. La idea general del proyecto consiste en que las distintas empresas que hagan uso de esta aplicación, dispongan de una plataforma online muy sencilla en la que fácilmente puedan subir las facturas que manejen diariamente para que, posteriormente, se utilicen como fuentes de datos para mostrar información relevante a las empresas en relación a su actividad económica como por ejemplo, el resultado de la liquidación del IVA calculado de manera automática.

La realización de este Trabajo de Fin de Grado está situado en un escenario en el que todas las facturas están digitalizadas, y se desea analizar el potencial que tiene realizar una aplicación de este tipo.

Con la realización de FinBook, el objetivo ha sido proporcionar al cliente una cierta comodidad en la realización de trámites relacionados con un mundo

Figura 2.1: Logo de Finbook

tan engorroso como es el de la economía y con la que desde un punto de vista legal, con un mínimo fallo se puede manchar enormemente la imagen de los usuarios. Además, se ha perseguido el propósito de dar un paso más en el mundo de la economía, y proporcionar una plataforma en la que el Estado pueda sacar provecho para controlar el gran fraude fiscal que se produce anualmente. Es por ello que se ha hecho hincapié en una serie de propuestas de valor únicas, que pueden marcar la diferencia con otras aplicaciones similares: poder realizar diferentes acciones con el mínimo coste temporal posible, la disposición de una gran cantidad de datos, y aportar al usuario fiabilidad con cada una de las acciones realizadas en la aplicación.

En cuanto al bajo coste temporal, cabe destacar la realización de una interfaz simple e intuitiva para la plataforma a la que se conectan los usuarios, que supone un gran ahorro de tiempo en primer lugar porque el usuario siente en todo momento que está haciendo uso de cualquier otra aplicación relacionada con el ámbito económico ya establecida en el mercado, y porque no requerirá de un gran uso de memoria para aprender series de pasos complicados para poder realizar acciones básicas. Esto sobretodo se consigue disponiendo desde que el usuario inicia sesión opciones como subir las facturas, que es una acción muy recurrente, y esto agilizaría los procesos y por consiguiente, supondrían un ahorro de tiempo.

En lo referido a la disposición de una gran cantidad de datos, ha sido posible debido a que se ha optado por una arquitectura *Big Data*, que permite cumplir fácilmente este requisito. La disposición de esta cantidad masiva de datos es esencial en una aplicación en la que se manejan elementos económicos que deben perdurar con el tiempo, ya que en cualquier momento, las empresas podrían querer visualizar datos muy antiguos en cualquier momento, por ejemplo. En relación a la propuesta de valor expuesta en el párrafo anterior, es cierto que una arquitectura *Big Data* no asegura un ritmo de transmisión de datos tan alto como es una base de datos, por ejemplo, pero está compensado por la gran cantidad de datos que puede almacenar.

Por último, se debe destacar la fiabilidad. Una aplicación financiera en la que los usuarios tengan que subir sus facturas, y aparezca cualquier otra información confidencial acerca de la empresa y que no sea fiable, probablemente no tenga éxito en el mercado. Es por eso que se ha optado por hacer uso de las firmas digitales como base de la seguridad en FinBook. Las facturas deberán estar certificadas, como ya ha sido mencionado, ya que es la manera en la que se puede asegurar digitalmente que la factura pertenece a una empresa, y que no ha sido un extraño el que haya manipulado esa factura o subido en nombre de dicha empresa. Además, el acceso a la plataforma

no sigue el tradicional sistema de nombre de usuario y contraseña, sino que también se hará uso del certificado para entrar y asegurar que el acceso está restringido.

En relación al tipo de facturas a utilizar, serán facturas del tipo estructuradas ya que como bien se ha mencionado con anterioridad, el manejo de datos en este tipo de facturas es mejor, se puede estandarizar fácilmente, y es la dirección que está tomando la digitalización en el mundo. El tipo de fichero utilizado es XML, y las facturas contendrán datos como la empresa emisora, la receptora, el importe, la fecha, etc. Por otro lado, en cuanto al tipo de facturación, evidentemente se llevará a cabo una digitalización fiscal, en la que las facturas deberán estar firmadas digitalmente.

En resumen, FinBook es una aplicación que se encarga de almacenar una cantidad masiva de facturas a tiempo real, al cual cualquier empresa tiene acceso. Para poder acceder a esta aplicación, se realiza mediante la firma digital de un texto enviado desde el servidor por parte de una pequeña aplicación de escritorio también desarrollada (Firma), que posteriormente, este texto firmado es devuelto al servidor y verificado por el mismo. Una vez dentro, se dispone de funcionalidades como subir facturas, que con un simple sistema *drag-and-drop*, estas facturas son almacenadas en un sistema de almacenamiento de datos. Este sistema, a su vez, extrae la información de dichas facturas y esta información útil es devuelta a la plataforma para representar diferentes datos útiles, estadísticas, liquidación del IVA, facturas subidas, etc.

2.1. Estado del arte

Para poder analizar el estado de este tipo de aplicaciones, debemos tener en cuenta do escenarios a analizar. Por un lado, tenemos un entorno más profesional y cercano al escenario que se intenta simular con el desarrollo de FinBook, compuesto por aplicaciones reales utilizadas por entidades que manejan una gran cantidad de facturas. Por otro lado, tenemos el entorno más orientado a la investigación, que en este caso, serían las aplicaciones que han surgido como lo ha hecho FinBook.

En cuanto al caso del escenario profesional, han hecho uso de una arquitectura *Big Data*, y lo han hecho de una manera como mínimo eficiente, ya que países como Chile, México o Brasil han podido potenciar y aprovechar al máximo la digitalización de la factura desde hace casi dos décadas. No

obstante, de cara al análisis de este tipo de aplicaciones, es muy difícil poder realizar uno, ya que las técnicas utilizadas para sus respectivo desarrollos no han sido compartidos.

Por otro lado, tras buscar acerca de otras aplicaciones similares desarrolladas en ámbitos de investigación, lo que se encuentran son aplicaciones muy pequeñas que facilitan la gestión económica de empresas, pero que no llegan a la magnitud que plantea inicialmente FinBook. Muchas de las investigaciones realizadas se quedan en el punto de plantear una solución, y analizar en detalle el problema, como es el caso de (Chen, Wu, y Miau, 2015). Entre todo este tipo de artículos, tenemos algunos que van un poco más allá. Uno de ellos es (Palacios de la Flor, s.f.), que podemos ver como claramente llega incluso a implementar alguna de las funcionalidades propuestas para FinBook, sin llegar del todo a plantear el mismo problema. No obstante, presenta problemas de cara a cumplir las expectativas de FinBook, como es la imposibilidad de reestructurar los datos o de almacenar miles y millones de ellos, ya que las bases de datos limitan en ese aspecto. Sin embargo, también nos podemos encontrar investigaciones o tesis que plantean el mismo problema, como es el caso de (Cayambe Chicaiza, 2015), que pese a que su solución esté muy bien planteada, tiene el mismo problema que la tesis anterior.

Tras este pequeño análisis, podemos darnos cuenta que el sector está muy poco explotado, y que lo poco explorado en el mismo, consta de soluciones basadas en el uso de técnicas convencionales.

2.2. Aplicaciones FinTech

FinBook pertenece a un grupo de aplicaciones que se les denomina comúnmente: aplicaciones *FinTech*. Este tipo de aplicaciones está revolucionando actualmente el sector financiero. Ya han invadido múltiples mercados, convirtiéndose en tendencia inmediata. Para entender el concepto *FinTech*, nos podemos remontar a la etimología de la palabra, la cual viene de los términos ingleses *finance* (finanzas) y *technology* (tecnología). Si unimos estas dos palabras, queda claro el propósito de este tipo de aplicaciones, y es utilizar la tecnología para ofrecer productos financieros acorde a los tiempos actuales. Es decir, un producto *FinTech* es uno tal que gracias al uso de una fuerte base tecnológica, permite optimizar, agilizar y abaratar procesos financieros (Zamora, 2016).

Debido al gran tamaño que tiene el sector financiero, es complicado que este tipo de aplicaciones haya abarcado aún todos los ámbitos de dicho sector. Los principales ámbitos que están cubiertos son el de pagos, distribución de productos financieros, préstamos y créditos, inversión, gestorías, entre otros. Como se puede observar, son muchos los ámbitos y créditos que se pueden cubrir, y las aplicaciones *FinTech* aportan una gran cantidad de beneficios, así que es inminente que estas aplicaciones cubran todos los ámbitos posibles del sector económico (Zamora, 2016).

El éxito de este tipo de aplicaciones está dado por una serie de factores. El primero de ellos es que son capaces de cubrir ciertas necesidades que el sector financiero tradicional no pudo, o cubrir mercados que carecían de interés. Otro punto clave es la adaptabilidad de estas aplicaciones, ya que se basan en la recopilación masiva de datos de distintos usuarios y de esta forma, consiguen ahorrar tiempo al usuario final. Tal vez en una primera instancia pueda parecer que hay una lucha entre este tipo de aplicaciones y los sectores financieros tradicionales, pero la realidad es que han llegado a acuerdos que permiten que los usuarios accedan a estos servicios tradicionales pero con un valor añadido considerable para dichos usuarios. Además, cabe destacar que los usuarios siempre necesitarán productos financieros, es decir, no es un mercado para un sector concreto de usuarios, o uno del cual se pueda prescindir (Zamora, 2016).

2.2.1. Financiación alternativa

La financiación alternativa es un tipo de financiación que proviene directamente de las soluciones *FinTech*, y cada vez está tomando más relevancia. De manera genérica, se entiende por "alternativas" el conjunto de fuentes de financiación independientes de los bancos o mercados tradicionales aunque de manera específica, alude especialmente a las soluciones *FinTech* (Circulantis, s.f.).

Los modelos de financiación alternativa más relevantes son:

- *Crowdlending*: se diseña un *marketplace* en el que un grupo de inversores registrados eligen los proyectos que más les interesen, aportando diferentes cantidades económicas para cubrir el montante capital solicitado.

- Anticipo de facturas o *Invoice trading*: son propuestas que ayudan a los autónomos y *pymes* a anticipar el dinero de sus facturas de forma ágil y sin exigencias.

- *Business angels*: un socio "protector" acepta participar en el negocio aportando dinero, experiencia y contactos. La relación se define en un contacto privado y es conveniente dejar bien especificadas las obligaciones del inversor y evaluar si puede cumplir las expectativas.

- Sociedades de capital de riesgo: se financia a las empresas a través de participaciones en el capital de forma temporal. Es perfecto para *start-ups* innovadoras y de gran potencial porque los inversores de este tipo aceptan proyectos de más riesgo si pueden obtener un retorno rápido y de mayor rentabilidad. El capital riesgo es una modalidad similar, no societaria.

- *Direct lending*: la inversión proviene de inversores institucionales, fundamentalmente en formas de fondos. Se puede concretar en formato de deuda (acciones) o de financiación con capital riesgo. Es más adecuada para sociedades más grandes y asentadas que necesitan impulsar su crecimiento o internacionalización, adaptándose a cada caso.

- Préstamos participativos: los inversores se agrupan de forma sindicada para financiar unidades de negocio o proyectos concretos a cambio de una participación en el resultado. Los prestamistas se involucran de forma personal y es esencial para el solicitante establecer bien la relación contractual y la forma de reembolso.

- Mercados alternativos de renta: el Mercado Alternativo de Renta Fija (MARF) y el Mercado alternativo Bursátil (MaB) se crearon para facilitar la negociación de títulos emitidos por *pymes*. Sin embargo, su impacto no ha sido grande y solo funcionan, relativamente, para las *pymes* más grandes.

- *Initial Coin Offering* (ICO): los recursos se captan a través del lanzamiento al mercado de una moneda virtual o un servicio, referenciada a algún valor o activo del que se espera un alto crecimiento. Es el equivalente a una salida a bolsa con acciones. (Circulantis, s.f.)

2.2.2. Estado de la financiación alternativa en Europa

Tras haber aterrizado en España en 2016, el impacto ha sido tan grande, que este país se llegó a situar como el sexto mayor mercado europeo de la financiación alternativa, manteniéndose durante años en los primeros puestos y solo por detrás de Reino Unido, Francia, Alemania, Holanda y Finlandia (Álvarez, 2018). Ya en 2016 se superaban en España los 213 millones de euros de financiación, lo que supone acumular el 10 % de los recursos del país. El mayor mercado estaba en las aplicaciones de pagos, seguidas de las de préstamos y herramientas de inversión (Zamora, 2016).

El mercado europeo de la financiación alternativa creció un 41 % en 2016, hasta los 7.671 millones de euros. Reino Unido es el principal líder de mercado, con un 73 % del total, y excluyendo a este, la industria de la financiación alternativa creció en Europa un 101 %. La regulación es uno de los grandes retos que esta industria tiene por delante en la Unión Europea (UE). No obstante, cabe destacar que la mayoría de los países europeos, entre ellos España, ha implementado en los últimos años algún tipo de regulación nacional. Asimismo, la Unión Europea está contemplando medidas de armonización de la normativa para que las empresas puedan operar a nivel Europeo (Álvarez, 2018).

Capítulo 3

Situación de partida

3.1. Motivaciones

A la hora de decidir que quería trabajar en este Trabajo de Fin de Grado, han entrado en juego varias motivaciones.

Una de ellas es poder conocer más de cerca el mundo económico. Está claro que el sector económico es una base importante para el funcionamiento del mundo actualmente. Además, a nivel personal, conocer más este sector no solo sirve para mejorar mi cultura general, sino que ese conocimiento puede ayudarme a defenderme ante situaciones de estafa y otro tipo de fraudes que, con una falta de conocimiento, pueden pasar desapercibidos. Es por eso, junto a mi poca experiencia en el mundo económico debido a mi corta interacción con el mismo, que me ha servido como motivación a la hora de lanzarme a realizarlo. Además, es un proyecto interesante porque es una oportunidad única de poder trabajar en un ámbito como es el de la digitalización de la economía, que en los próximos años estará en auge y que probablemente, como futuro ingeniero informático, probablemente me vea inmerso en el desarrollo de este tipo de proyectos.

Otra motivación importante es el poder trabajar en la realización de una aplicación *FinTech*. Tras investigar acerca del estado del mundo financiero en el ámbito de la informática, no he dudado en decidirme por realizar este proyecto, ya que existe un claro retraso respecto a otros sectores. Ser en cierta medida pionero en realizar una aplicación de este estilo en España es algo que tuve en cuenta a la hora de decidir, ya que cuando se produzca el "boom" de las aplicaciones *FinTech*, podré aportar todos los conocimientos adquiridos en

la realización de este proyecto y poder sumarme a esta ola de manera activa y aportar experiencia para sacar lo mejor de estas aplicaciones. Además, la propia realización de este proyecto me iba a proporcionar una visión de cómo funciona una aplicación *FinTech*, conocimiento que se puede aprovechar a la hora de tener que utilizar otras aplicaciones de este tipo, ya que, en base a la experiencia obtenida, puedo detectar de manera más fácil si una aplicación es segura, entre otros aspectos.

Sin embargo, mi mayor motivación a la hora de realizar este Trabajo de Fin de Grado es el poder realizar de primera mano una aplicación con una arquitectura *Big Data*. Debido a lo reciente que es esta tecnología, no he tenido la oportunidad de poder estudiar ni trabajar en un proyecto relacionado con la misma, y es por eso que he valorado mucho el poder realizar este proyecto, ya que por la naturaleza del mismo, debe estar construido en base a esta tecnología. Sabía que la oportunidad de poder tener contacto con *Big Data* era única y de no aprovecharla, entraría en el mundo laboral sin tener conocimientos de este aspecto, y la realidad es que con la globalización, informatización y aparición de muchas redes sociales nuevas, las empresas se centran cada vez más en recopilar esta información, y necesitan cada vez más de personas que sepan como tratar estos datos. Es por eso que este proyecto me podría brindar una competencia extra que pueda ayudar a esas empresas y a distinguirme en el mundo laboral, además de solventar mis curiosidades acerca de esta tecnología.

Por último y no menos importante, destacar que otro punto muy a favor fue saber que, pese a que el proyecto es completamente individual y que no debe depender del de otros compañeros, me ha llamado mucho la atención el saber que realmente este proyecto formará un todo con el de otros cuatro compañeros más y al fin y al cabo, en el mundo laboral el trabajo en equipo es fundamental, y es por eso que el trabajar esa competencia me parece muy atractivo.

3.2. Objetivos iniciales

Los objetivos planteados inicialmente en la realización de este proyecto son:

- Identificar los componentes necesarios y como podría ser el aspecto final del trabajo, ya que es necesario tener una visión global de cómo va a ser el proyecto y así evitar posibles imprevistos.

- Creación de una plataforma de datos que contendrá una gran cantidad de datos en bruto obtenidos por los generadores de eventos, y que posteriormente serán tratados.
- Establecer una estrategia de privacidad y seguridad sobre los datos, ya que muchos de los datos podrían ser confidenciales.
- Crear un sistema de transferencia de datos, con el que los posibles usuarios puedan ingresar sus datos correctamente en el repositorio, y con el que se puedan obtener dichos datos.

3.3. Organización

Este proyecto tiene una serie de especificaciones que hay que tener en cuenta a la hora de organizarlo. En primer lugar, puede ser lo suficientemente grande como para no poder ser desarrollado por una sola persona. No obstante, la propia arquitectura permite dividir este proyecto en módulos, ya que claramente se pudo ver que hace falta una plataforma para almacenar y digerir los datos, un módulo de visualización de los mismos, y algún tipo de estructura que genere e importe los datos en la plataforma de datos.

Es por esto que se decidió que el proyecto podía ser dividido en distintas partes, en la que cada integrante realizaría una distinta de manera individual y, posteriormente, que puedan ponerse en común y ver su funcionamiento en conjunto. Cabe destacar que como se ha mencionado, cada parte se ha realizado individualmente y de manera independiente. Es decir, cada parte funciona por sí sola sin depender del trabajo realizado por otros compañeros, por lo que si alguna de las partes falla, no habría ningún problema. Las tareas que sí requerían la disposición de dos o más miembros son las de integración, destacando las discusiones sobre cómo debía ser el formato de las facturas, ya que es un aspecto crítico que se debe tener en cuenta al almacenarlas o al mostrar información de las mismas.

Los cuatro integrantes del grupo son: Gerardo Santana Franco, que realizará un módulo de visualización de información general de las facturas, además de un sistema que a través de una interacción con el usuario *drag and drop*, se pueden subir facturas manualmente a la plataforma de datos; Juan Kevin Trujillo Rodríguez, que desarrollará una plataforma que visualiza información específica acerca del IVA y su liquidación; Juan Alberto Ureña Rodríguez, que realizará un simulador para generar facturas pudiendo

cambiar parámetros de las mismas; y yo, Raúl Lozano Ponce, realizaré la plataforma de datos, y me encargaré de dar soporte al sistema de *login*, creando una pequeña aplicación de escritorio para firmar textos, y una librería que disponga de funciones para comprobar la validez de esta firma.

3.4. Planificación

- Estudio del estado del arte de la arquitectura *Big Data* y de los *Datahubs*. Aprendizaje de la herramienta Ïntino"(https://www.intino.io/), y de otros aspectos necesarios para la elaboración del trabajo como aspectos financieros, autenticación con firmas digitales, etc.
- Planificación de trabajo con los compañeros que realizarán los otros módulos, así como la creación de repositorios, recolección de información y datos de interés, etc.
- Desarrollo de un módulo de autenticación mediante firma digital, compuesto por una pequeña de aplicación de escritorio que firma un texto de forma segura, y una librería integrada con el Entorno de Usuario que compruebe la validez de la firma.
- Desarrollo de una ontología de facturas, creando un sistema en el que se reciben facturas de una empresa determinada, y generando datos para la misma útiles (como el IVA devengado y repercutido) tratando los datos de la propia factura.
- Desarrollo de una ontología de nóminas, haciendo uso del sistema para recibir facturas indicado en la tarea anterior, obteniendo datos de estas para generar nuevos relacionados con las nóminas.
- Testeo y búsqueda de errores.
- Integración de la parte desarrollada con la de los compañeros.
- Realización de la documentación correspondiente.
- Preparación de la presentación del trabajo.

Capítulo 4

Arquitectura

El proyecto consta de tres partes bien diferenciadas: los sensores de datos, el sistema de almacenamiento de datos, y los *Datamarts* o visualizadores de datos. En mi caso concreto, me encargaré de realizar principalmente un sistema de almacenamiento de datos para este proyecto. Además, hará falta un pequeño sistema para el *login* en la aplicación, basado en las firmas digitales para aumentar la seguridad, ya que en una aplicación *FinTech* se manejan muchos datos confidenciales.

Tras pensarlo bien, se ha decidido realizar el proyecto sobre una arquitectura *Big Data*, ya que los módulos que lo componen coinciden perfectamente con los que podemos encontrar en cualquier otra aplicación basada en una arquitectura *Big Data*: claramente está formado por un conjunto de sensores que generan eventos, que son las personas subiendo facturas a la plataforma; un sistema de almacenamiento de datos, en el que se almacenan los eventos de las facturas subidas y las posteriormente procesadas; y un consumidor de esos datos, que en este caso es la plataforma de visualización de datos acerca de las facturas subidas. No obstante, el factor clave por el cual se ha decidido utilizar esta arquitectura es la forma en la que se almacenan los datos, la cual está clara por la naturaleza de la aplicación en la que cada día se subirán miles de facturas nuevas y en la que interesa que haya un registro temporal de las mismas: justamente dos de los factores necesarios para decantarse por esta arquitectura.

Por otro lado, cabe destacar la forma de conectar estos módulos. Para el correcto funcionamiento de esta aplicación, necesitábamos una forma en la que cada módulo simplemente se encargara de realizar su función, y enviar los datos necesarios a otro módulo o conjunto de módulos, preferiblemente sin

conocer a estos módulos para así asegurar la escalabilidad de la aplicación. Es por eso que se ha elegido como método de comunicación entre módulos hacer uso del patrón de diseño *Publisher/Subscriber*. En resumen, el sensor publicaría eventos en el sistema de almacenamiento de datos, y el consumidor de estos datos estaría suscrito a dicho sistema de almacenamiento. Esta arquitectura permite mantener una comunicación asíncrona entre los módulos, y permite montarlo de tal manera que se haga uso de un *buffer* para así evitar que estos datos se pierdan debido a un problema de saturación del sistema.

4.1. Big Data

En el día a día se comparte una gran cantidad de datos que contienen información muy útil para el análisis desde un punto de vista empresarial. De hecho, la información útil obtenida de estos datos supone cada vez más la principal fuente de información de las empresas acerca de los clientes o potenciales clientes, por encima incluso de las tradicionales encuestas que con la digitalización de los últimos años se están quedando desfasadas.

La realidad es que pese a que haya una gran comunidad en contra de esta forma de obtener información, siendo etiquetada de poco ética está más que demostrado que es la forma más efectiva de obtener grandes cantidades de información útil, ya que comparándolo con el caso de las encuestas, en cuanto a la relación de cantidad de datos obtenidos por tiempo, es infinitamente mayor, además que no todos los usuarios disponen del tiempo de realizar una larga encuesta, sin olvidarse de que aunque estas sean anónimas, la sinceridad está normalmente alterada.

Antes de hablar de *Big Data*, cabe matizar en algunos términos. Según la RAE (*REAL ACADEMIA ESPAÑOLA: Diccionario de la lengua española*, 2008), un dato desde el punto de vista de un ámbito informático es una información dispuesta de manera adecuada para su tratamiento por una computadora. Cuando se habla de tratamiento, se incluye la lectura, transmisión y escritura del mismo.

Una aplicación con una arquitectura *Big Data* es aquella que realiza un proceso de recolección y análisis de grandes cantidades de información. La complejidad y el gran volumen de datos impiden que estos puedan ser analizados por los medios tradicionales (Jiménez, 2019). Un claro ejemplo de un ámbito en el que se utilizan aplicaciones *Big Data* es en aquellas que capturan

información de las redes sociales. Más de 500 terabytes de datos nuevos se ingieren de redes sociales como Facebook todos los días. Estos datos se generan principalmente con la subida de vídeos, fotos, intercambio de mensajes, comentarios, etc (Guru99, 2014).

Existen tres maneras en las que se pueden encontrar dispuestos los datos almacenados en una aplicación *Big Data*:

- Estructurados.
- Desestructurados.
- Semi-estructurados. (Guru99, 2014)

En cuanto a la forma estructurada, los datos, llamados datos estructurados, se pueden almacenar, acceder a ellos, y procesarse en forma de formato fijo. Con el paso del tiempo, empresas dedicadas a *computer science* han logrado un mayor éxito en el desarrollo de técnicas tanto para trabajar con este tipo de datos, en los que el formato es bien conocido, como para obtener valor de los mismos. No obstante, hoy en día se prevee que pueden dar problemas cuando el tamaño de estos datos crece en gran medida, ya que los tamaños típicos rondan el orden de los zettabytes (1021 bytes). En ejemplo de datos estructurados son los datos almacenados en sistemas de gestión de bases de datos relacionales (Guru99, 2014).

Cualquier dato con forma o estructura desconocida se clasifica como dato no estructurado. Además de que el tamaño de estos datos es bastante grande, los datos no estructurados plantean muchos desafíos en términos de so procesamiento para extraer valor de ellos. Un ejemplo de este tipo de datos son los obtenidos de una fuente de datos heterogénea que contienen una combinación de archivos de texto simples, imágenes, videos, etc. Hoy en día, muchas organizaciones disponen de una gran cantidad de datos disponibles pero, desafortunadamente, no saben cómo obtener valor de ellos y es por eso que los tienen en su forma cruda o formato no estructurado (Guru99, 2014).

Los datos semi-estructurados pueden contener ambas formas de datos. Podemos ver datos semi-estructurados como una forma estructurada, pero que en realidad no están definidos con una definición de tabla en un sistema de gestión de bases de datos relacional. Un ejemplo de datos semi-estructurados son los datos representados en archivos XML (Guru99, 2014).

Existen 5 principios, conocidos como las 5 Vs, que son características de las aplicaciones *Big Data*, que hacen de estas aplicaciones diferentes al resto:

- Volumen: es la característica más asociada al concepto de *Big Data*. La cantidad de datos generados a día de hoy por mediación de la tecnología, es enorme. Los avances tecnológicos han permitido no solo generar todos estos datos, sino también, almacenarlos en *clusters* o en la nube.
- Velocidad: los datos se generan de manera continua e instantánea. Esto supone que muchas veces, los datos tengan ciclos de vida cortos, haciendo obsoletos los datos que instantes antes eran válidos.
- Variedad: los datos obtenidos son diversos en cuanto a tipología y fuentes de obtención. Esta diversidad es clave para la riqueza de posibilidades del *Big Data*. Esto a su vez aumenta el grado de complejidad tanto en su almacenamiento como en su procesamiento y análisis.
- Veracidad: la veracidad puede entenderse como el grado de confianza que se establece sobre los datos a utilizar. La veracidad de los datos determinará la calidad de los resultados y la confianza en los mismos.
- Valor: supone el conocimiento e información útil que se puede extraer de los datos. (de Alcalá, 2018)

Analizada la propuesta de valor que ofrece *Big Data*, ¿qué beneficios conlleva el uso de esta tecnología? Para empezar, las empresas u organizaciones pueden utilizar la inteligencia externa mientras ellos toman decisiones, ya que el acceso a los datos sociales desde los motores de búsqueda y desde sitios como Facebook o Twitter, están permitiendo a las organizaciones ajustar sus estrategias comerciales. El servicio al cliente queda mejorado, ya que los sistemas tradicionales de retroalimentación de los clientes están siendo reemplazados por nuevos sistemas diseñados con tecnologías *Big Data*. En estos nuevos sistemas, las tecnologías de procesamiento de *Big Data* y lenguaje natural se están utilizando para leer y evaluar las respuestas de los consumidores. Además, se puede identificar de manera temprana el riesgo para el producto/servicio si existe, y mejorar la eficiencia operacional, ya que las tecnologías *Big Data* se pueden usar para crear un área de preparación o zona de aterrizaje para nuevos datos antes de identificar qué datos se deben mover al sistema de almacenamiento de datos. Dicha integración de las tecnologías de *Big Data* y el almacén de datos, ayuda a una organización a descargar datos a los que se accede con poca frecuencia (Guru99, 2014).

La estructura general que siguen las arquitecturas de *Big Data* se puede resumir en el diagrama dato por la Figura 4.1.

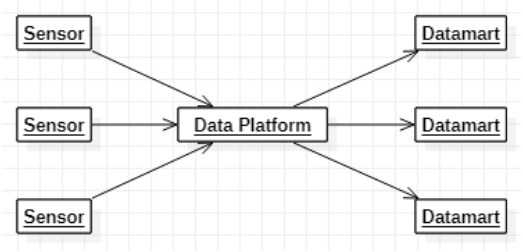

Figura 4.1: Estructura seguida por una aplicación Big Data.

El primer elemento a destacar, es el sensor. Los sensores son todos aquellos componentes que de una manera u otra, generan eventos que contienen los datos a almacenar. Un evento es un mensaje que notifica a otro componente (en este caso al *Datahub*) que el estado de lo que se esté sensorizando, ha cambiado. El mensaje del evento puede contener cualquier tipo de información, y es por eso que existen muchos tipos de sensores diferentes. Algunos ejemplos de sensores son los teléfonos móviles, que pueden ser usados por ejemplo para detectar cuándo las personas se conectan a ciertas redes *Wi-fi*, los datos generados por la Agencia Estatal de Meteorología en su página web, los *tweets* publicados en Twitter, , y como es importante nombrar, las facturas generadas por las empresas. Cabe destacar que los campos de los mensajes dependen del tipo de sensor que los capture, como es evidente. Sin embargo, es importante que contenga un registro de qué sensor es el que generó el mensaje y sobretodo, el instante en el que lo hizo, para poder llevar un control temporal sobre los mismos.

El siguiente elemento representado en el diagrama es el *Storage System*, es decir, el sistema que se utiliza para almacenar los datos. Cada sistema tiene su forma peculiar de almacenar datos, que está relacionado con el tipo de datos manejado, ya explicado con anterioridad. Existen diferentes sistemas que si bien se diferencian claramente si se comparan, normalmente son confundidos entre sí. Es por eso que se explicará cada uno a continuación.

Los *Datalakes* son sistemas de almacenamiento de datos en los que los mismos se almacenan en su forma natural, es decir, sin procesar, y en los que normalmente el propósito de estos datos aún no se ha definido. La información almacenada en un *Datalake* es altamente accesible y rápida de actualizar, lo que hace que su uso sea ideal en la realización de tareas como el almacenamiento de informes, visualización, análisis y *machine learning*. La naturaleza bruta de los datos requiere una gran capacidad de almacenamiento en comparación con los *Datahubs* o *Datawarehouses*. No obstante, en muchos entornos se suelen utilizar cuando se necesita un sistema maleable que sea

Figura 4.2: Amazon S3, un ejemplo de Datalake.
Gopisetty, D. (2019). A Story of Simple Storage Service (AWS S3). [Figura]. Recuperado de https://medium.com/@dolicag/a-story-of-simple-storage-service-aws-s3-3f4f8cbc2dd7.

Figura 4.3: BigQuery, un ejemplo de Datawarehouse.
Grynko, K. (2019). Google BigQuery: A Tutorial for Marketers. [Figura]. Recuperado de https://cxl.com/blog/bigquery/.

rápido de analizar y más adecuado para el *machine learning*. Un *Datalake* se puede implementar de varias maneras, incluyendo datos estructurados en bases de datos relacionales, datos semi-estructurados, datos no estructurados, incluso datos binarios sin procesar. Existen servicios que ya utilizan este método de almacenamiento de datos de rápido crecimiento, como Azure Data Lake, Amazon S3, y el mencionado Apache Hadoop (Ivanovic, 2019).

Un *Datawarehouse* es un sistema de almacenamiento de datos utilizado sobretodo en ámbitos de informes y análisis de datos. Este tipo de sistemas de repositorio se ocupa de los datos que se han cargado directamente desde los sistemas operativos. A diferencia de los *Datalakes*, los *Datawarehouses* solo se encargan de datos procesados, lo que ofrece una ventaja en términos de espacio de almacenamiento y accesibilidad a un público más amplio. Un *Datawarehouse* se utiliza para crear informes analíticos en curso. La mayoría de los *Datawarehouses* se basan en un sistema *Extract, Transform and Load* (ETL). Se debe elegir un *Datawarehouse* cuando necesiten datos de sistemas operativos para estar listos y a la espera de un análisis. Existen diferentes métodos para organizar un *Datawarehouse*, con una funcionalidad general definida por el *hardware*, el *software*, y los recursos de datos que definen la

Figura 4.4: Hadoop, un ejemplo de Datahub.
Redacción Byte TI (2016). Apache Hadoop cumple 10 años. [Figura].
https://revistabyte.es/actualidad-it/hadoop-cumple-10-anos/.

arquitectura específica del *Datawarehouse*. Se puede hacer uso de un enfoque dimensional, o un enfoque normalizado: con el enfoque dimensional, se dividen los datos en *facts* y *dimensions*; en el enfoque normalizado, se crean segmentos de datos de acuerdo con las reglas convencionales de normalización de bases de datos. Existen soluciones populares como Amazon Redshift, Panoply y BigQuery (Ivanovic, 2019).

Los *Datahub* son colecciones de eventos procedentes de uno o más sensores, y normalmente organizados. El uso de un *Datahub* es importante a la hora de distribuir datos de manera eficiente y con un formato deseado, como bien es requerido en una arquitectura *Big Data*. Aunque los *Datahubs* comparten muchas similitudes con los Datawarehouse, no se limitan a almacenar datos operativos. Los *Datahubs*, a diferencia de los *Datalakes*, proporcionan acceso a datos homogeneizados en formatos deseados. Además, en lugar de almacenar datos en un punto específico para garantizar un acceso fácil, los *Datahubs* utilizan múltiples ubicaciones y formatos para permitir una fácil duplicación, calidad, seguridad y estandarización. Cabe destacar que permiten que los nodos intermedios almacenen y ejecuten una gran variedad de información. Esto en términos de almacenamiento, es poco eficiente. Sin embargo, este enfoque ofrece eficiencias operativas reales, e impulsa la agilidad. Los *Datahubs* no almacenan información de transacciones, y son pequeños en comparación con los *Datalakes* o *Datawarehouses*. El uso de este sistema es una buena opción si se requiere de una buena calidad de datos con un uso compartido de los mismos, escalabilidad y un mayor acceso a servicios de consulta más eficientes. Existen productos de almacenamiento que funcionan a modo de *Datahub* que además como es el caso de Apache Hadoop, Google MapReduce, entre otros (Ivanovic, 2019).

Como se puede observar en la Figura 4.5, puede verse resumido en una tabla la diferencia entre las tres estructuras de almacenamiento de datos.

El tercer y último elemento que se aprecia en el diagrama son los *Datamarts*. Los *Datamarts* son subconjuntos de datos almacenados en cualquiera

	Data Lake	Data Warehouse	Data Hub
Data	Structured to unstructured, relational to non-relational	Structured, relational	Structured to unstructured
Schema	Schema-on-read	Schema-on-write	Schema-on-write
Price	Low cost storage	Higher cost storage	Higher cost storage
Data Quality	Raw, may not be curated	Highly curated	Highly curated
Users	Data scientists and developers	Business analysts	Diverse business users
Architecture	Centralised	Centralised	Hub and spoke

Figura 4.5: Comparación de los sistemas de almacenamiento.
Ivanovic L. (2019). Data Lake vs. Data Warehouse vs. Data Hub – What's the Difference? [Figura]. https://miktysh.com.au/data-lake-vs-data-warehouse-vs-data-hub-whats-the-difference/

de los sistemas de almacenamiento explicados, que pueden ser utilizados para las necesidades de un equipo, sección o departamento de una empresa, entre otras utilidades. Los *Datamarts* hacen que acceder a un tipo concreto de información sea mucho más fácil y más rápido, y la separación de los distintos tipos de datos en varios *Datamarts* según las necesidades requeridas por cada funcionalidad, evitan que las funcionalidades de la aplicación puedan acceder a información fuera de su contexto, y así no interferir en estos datos. Además, los *Datamarts* permiten generar, recopilar, y analizar datos, un proceso conocido como computación perimetral, que ayuda a preservar el ancho de banda y evitar problemas de latencia (Sisense, 2017).

En el caso de FinBook, la estructura a seguir en relación a lo visto de *Big Data* será la siguiente:

- Constará de dos sensores, los cuales se encargarán de alimentar la estructura de almacenamiento de datos a través de facturas en ficheros XML.
    - El primero de los sensores constará de una plataforma web en la que se podrá subir las facturas manualmente, debiendo estar estas

firmadas digitalmente. Este sensor correspondería al sensor real del cual dispondría la aplicación FinBook.

- Se dispone de un segundo sensor que si bien no es un sensor real, para poder realizar pruebas sobre la aplicación es esencial. Se trata de un simulador de generación de facturas que, a través de datos recopilados de Trip Advisor, genera una gran cantidad de facturas realistas en cuestión de segundos, con un formato similar al tipo de facturas que se utilizan en el primer sensor mencionado.

- Las facturas que se suben/generan en los sensores, serán almacenadas en un sistema de almacenamiento de datos. Los datos utilizados en esta aplicación son semi-estructurados, ya que están almacenados en formatos XML. En cuanto a la naturaleza del sistema de almacenamiento, se ha descartado la opción de utilizar un *Datawarehouse*, ya que no era necesario realizar ningún tipo de estudio con los datos. La verdadera cuestión es, para este caso, ¿se debe usar un *Datahub* o un *Datalake*? Recordemos que la mayor diferencia entre ambos es que en el *Datalake*, los datos se almacenan en bruto, mientras que en el *Datahub*, hay un pequeño procesamiento sobre los mismos y se distribuyen con unos formatos específicos. En el caso de la realización de FinBook, se ha optado por el uso de un *Datahub*, ya que el propósito de esta aplicación está más relacionado al ámbito empresarial que al de *Data Science*. Además, la escalabilidad es muy alta, hablando desde el punto de vista de los diferentes *Datamarts* que se pueden enganchar a este sistema de almacenamiento y responsabilizar a los *Datamarts* del filtrado de datos, además que no es correcto ya que la única responsabilidad de este módulo debe ser obtener los datos filtrados y mostrarlos, haría muy ineficiente la realización de esta responsabilidad.

- FinBook cuenta con dos *Datamarts*:

  - El primero de ellos muestra información acerca de la liquidación del IVA de cada empresa en base a los datos recopilados de las facturas obtenidas del *Datahub* que están relacionadas directamente con la empresa a analizar.
  - El segundo muestra más información acerca de la empresa, como las facturas subidas, factura que involucran a dos empresas específicas, etc, pero desde un punto de vista más general.

4.2. Patrón Publisher/Subscriber

Como se ha explicado en el apartado anterior, las distintas partes que componen una arquitectura *Big Data* deben comunicarse entre sí. Para resolver esta necesidad, se ha optado por usar el patrón *Publisher/Subscriber*.

Publisher/Subscriber es un patrón que permite que una aplicación o módulo de una anuncie evento de forma asíncrona a varios consumidores interesados, sin necesidad que los remitentes conozcan a los receptores (Microsoft, 2018).

En las aplicaciones distribuidas y basadas en la nube, como es el caso de FinBook, los componentes del sistema a menudo necesitan proporcionar información a otros componentes a medida que suceden los eventos. La mensajería asíncrona es una forma eficaz de desacoplar a los remitentes de los consumidores y evitar bloquear al remitente para que espere una respuesta. Sin embargo, crear una cola de mensajes para cada consumidor no es efectivo, y cada consumidor puede estar interesado en un subconjunto de la información (Microsoft, 2018).

Una solución a este problema es crear un sistema de mensajería asíncrona que incluya:

- Un canal de mensajería en el que el remitente o *publisher* empaquete los eventos en mensajes mediante un formato de mensaje conocido y los envíe a través de un canal de entrada.
- Un canal de mensajería de salida por consumidor o *subscriber*.
- Un mecanismo para copiar cada mensaje del canal de entrada a los canales de salida para todos los *subscribers* interesados en ese mensaje. Esta operación suele manejarla un intermediario como un agente de mensajes o un bus de eventos. (Microsoft, 2018)

Los componentes lógicos de este patrón se muestran representados claramente en la Figura 4.6: los *publishers* publican los mensajes en un canal de entrada (*input channel*), el *message broker* (agente de mensajes o bus de eventos) se encarga de "traducir" el mensaje para posteriormente enviarlo al canal de salida (*output channel*). Al enviar el mensaje, se notifica a todos los *subscribers* que estén suscritos.

El hacer uso de este patrón supone una serie de ventajas:

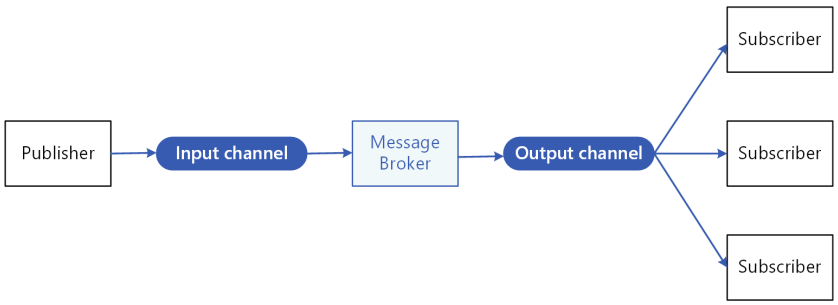

Figura 4.6: Estructura del patrón Publisher/Subscriber.
Microsoft (2018). Patrón de publicador y suscriptor. [Figura]. https://docs.microsoft.com/es-es/azure/architecture/patterns/publisher-subscriber.

- Tal vez la principal causa del uso de este patrón es el desacoplamiento de los distintos componentes que forman la arquitectura principal, ya que de esta manera, estos componentes se pueden administrar de forma independiente y los mensajes se pueden administrar correctamente incluso si algunos de los receptores están desconectados. Esta ventaja proporciona facilidad a la hora de definir las responsabilidades que se le da a cada componente, de modo que dicho componente se debe centrar únicamente en sus funcionalidades. Además, este desacoplamiento permite una integración más sencilla entre sistemas que utilizan diferentes plataformas, lenguajes de programación o protocolos de comunicación, así como entre sistemas locales y aplicaciones ejecutadas en la nube.

- Aumenta la escalabilidad y mejora la capacidad de respuesta del remitente, ya que solo se tiene que preocupar de de enviar los mensajes al canal de entrada y luego volver a sus responsabilidades de procesamiento principales. Es la infraestructura de mensajería la que se encarga de garantizar que los mensajes lleguen a los suscriptores.

- Mejora la confiabilidad, ya que la mensajería asíncrona ayuda a que las aplicaciones continúen funcionando sin problemas bajo cargas cada vez mayores y controlen los errores con mayor eficacia.

- Permite el procesamiento diferido o programado. Los suscriptores pueden esperar para recoger los mensajes hasta las horas de menor consumo e incluso los mensajes pueden ser procesados de acuerdo a un horario específico.

- Mejora la capacidad de prueba, ya que los canales pueden supervisarse y los mensajes se pueden inspeccionar o registrar como parte de las pruebas de integración. (Microsoft, 2018)

A la hora de implementar este patrón, se deben tener en cuenta una serie de consideraciones:

- Control de las suscripciones: la infraestructura de mensajería debe proporcionar mecanismos que los consumidores puedan utilizar para suscribirse o darse de baja de los canales disponibles.
- Seguridad: la conexión a cualquier canal de mensajes debe estar restringida por una directiva de seguridad para evitar que usuarios o aplicaciones no autorizados resulten interceptados.
- Subconjunto de mensajes: por lo general, los suscriptores solo están interesados en el subconjunto de los mensajes distribuidos por un publicador. Los servicios de mensajería a menudo permiten a los suscriptores reducir el conjunto de mensajes recibidos por:
  - Temas: cada tema tiene un canal de salida dedicado, y cada consumidor puede suscribirse a todos los temas pertinentes.
  - Filtrado de contenido: los mensajes se inspeccionan y se distribuyen según su contenido. Cada suscriptor puede especificar el contenido que le interesa.
- Suscripción a múltiples temas: se debe considerar que un mismo suscriptor pueda estar suscrito a múltiples temas.
- Orden de los mensajes: el orden en el que las instancias de consumidor reciben los mensajes no está garantizado y no refleja necesariamente el orden en el que se crearon los mensajes. Se debe diseñar un sistema en el que se pueda asegurar que el procesamiento de los mensajes sea idempotente para eliminar cualquier dependencia en el orden en el que se administra los mensajes.
- Mensajes venenosos: un mensaje con formato incorrecto o una tarea que requiere acceso a recursos que no están disponibles puede hacer que una instancia de servicio produzca un error. El sistema debe impedir que dichos mensajes vuelvan a la cola. En su lugar, capture y almacene los detalles de estos mensajes en otro lugar para que se puedan analizar si es necesario.

- Mensajes repetidos: el mismo mensaje se podría enviar varias veces. Por ejemplo, se puede producir un error después de que el remitente publique un mensaje. A continuación, una nueva instancia del remitente podría iniciarse y repetir el mensaje. La infraestructura de mensajería debe implementar la detección y eliminación de mensajes duplicados basada en identificadores de mensajes para proporcionarlos una sola vez.

- Expiración de mensajes: un mensaje puede tener una duración limitada. Si no se procesa dentro de este período, es posible que ya no sea pertinente y se deba descartar. Un remitente puede especificar un tiempo de expiración como parte de los datos del mensaje. Un receptor puede examinar esta información antes de decidir si desea realizar la lógica de negocios asociada con el mensaje.

- Programación de mensajes: un mensaje puede estar prohibido temporalmente y no debe procesarse hasta una fecha y hora específicas. El mensaje no debe estar disponible para los receptores hasta ese momento. (Microsoft, 2018)

Capítulo 5

Datahub de Finbook

Mi tarea principal dentro de la realización de FinBook es realizar su *Datahub*. El objetivo es simple: debo crear un módulo que permita la entrada de facturas brutas, tratar esta factura, y extraer los distintos datos que la componen, para que posteriormente puedan ser utilizados por otros módulos. Para ello, he realizado este módulo haciendo uso de Ïntino", que facilita el desarrollo de una aplicación basada en una arquitectura *Big Data*, y que de una manera muy intuitiva, permite montar los distintos elementos que componen dicha arquitectura.

Como bien se ha especificado con anterioridad, se hará uso de una arquitectura *Publisher/Subscriber*, lo cual supone que este *Datahub* no conoce realmente qué otros módulos son los que publican eventos, y qué otros módulos son los que se suscriben a ellos. Esto permite que la aplicación sea completamente escalable, y que en el momento que se precise, se pueda añadir un nuevo módulo que publique eventos o que se suscriba a los mismos.

Existe un gran problema que se puede plantear a partir de esta idea, y es acerca del formato de los eventos: y es que los eventos deberían tener la misma forma, para que posteriormente puedan ser tratados indiscriminadamente. Para ello, cabe destacar en primer lugar la existencia de los tanques. Para definir lo que es un tanque, se puede hacer mediante una analogía con tanques físicos utilizados para almacenar líquidos. Los proveedores son los que proporcionan estos líquidos y solo tienen permisos para depositar los líquidos que proveen en los tanques destinados para ello, y los compradores solo pueden acceder a los tanques que contengan los líquidos que ellos quieren. En resumen, un tanque es un repositorio que se rellena con eventos de un mismo tipo generados por los sensores, y sobre los cuales otros módulos

se suscriben para obtener estos datos. En este caso, FinBook dispone de dos tanques: uno en el que se almacenan las facturas en bruto codificadas en base 64, y otro tanque en el que se se almacenan las facturas, ya decodificadas y despiezadas en los distintos campos que la componen.

En este caso, la comunicación está dada por unos terminales. Cada tanque tiene su terminal con su formato del evento definido, y los distintos módulos, dependiendo de su función, envían los eventos por estos terminales, o están a la escucha de que llegue algún evento.

Una decisión importante ha sido saber separar las responsabilidades. Como se ha comentado con anterioridad, el objetivo de este módulo es obtener las facturas brutas, y proporcionar los distintos elementos que la componen por separado, pero se debe recordar que el objetivo del *Datahub* es únicamente almacenar los eventos, y comunicarse por los terminales, por lo que lo ideal es liberarlo de la responsabilidad de la extracción de datos. Es por eso que se ha desarrollado un módulo secundario cuya función es realizar esta extracción, y simplemente hace uso de los terminales mencionados: es suscriptor del tanque en el que llegan las facturas brutas, y es publicador del tanque en el que se depositan las facturas procesadas. Es decir, está a la escucha que de llegue una factura nueva, y cuando llega alguna, la procesa, y genera un evento nuevo en el segundo tanque.

5.1. Diseño de la API

El *Datahub* dispone de una *Application Programming Interface* (API) para conectarse con el mismo y hacer uso de los servicios que ofrece. Tanto para la conexión con el tanque de las facturas brutas en el que se debe publicar, como con el tanque de las facturas procesadas sobre el cual se debe suscribir, hay que realizar una tarea en común: establecer una conexión. Esta conexión se realiza mediante un objeto *JmsConnector*, que por detrás realiza esta conexión a través de una API *Java Message Service* (JMS).

Para realizar esta conexión, se definen cinco parámetros básicos como puede verse reflejado en el Código 5.1: la url del servidor, el usuario, contraseña e id del cliente, y el directorio del servidor en el que se va a publicar el evento. Cabe destacar que la información relativa al cliente viene dada por lo definido en el propio *Datahub*, que como se verá más adelante, permite diferenciar las posibilidades que dispone un tipo de usuario u otro del *Datahub*.

Código 5.1: Conexión con Datahub en local

```
JmsConnector connector = new JmsConnector(
        "failover:(tcp://localhost:63001)",
        "io.finbook.datamart",
        "io.finbook.datamart",
        "io.finbook.datamart",
        new File("C:\\Users\\raull\\IdeaProjects\\" +
                "finbook\\temp\\terminals")
);
connector.start();
```

Tras esta definición, simplemente habría que iniciar el conector, y crear el terminal que se va a usar asociándolo con dicho conector. Tras haber definido el terminal que se va a usar, llega la hora de publicar o suscribirse a dicho terminal, que se realiza de forma distinta pero no complicada.

En cuanto a la publicación de eventos sobre el tanque de las facturas en bruto, como se puede observar en el Código 5.2, básicamente consiste en crear un objeto Invoice, y asignarle el XML en base 64, junto al momento y autor en el que se publica el evento (estos dos últimos campos son obligatorios en todos los eventos).

Código 5.2: Publicación en el Datahub en local

```
SensorTerminal st = new SensorTerminal(connector);

st.publish(new Invoice()
        .ts(Instant.now())
        .ss("Prueba")
        .xml(TextGenerator.getBase64TextFrom("<?xml "+
        "version=\"1.0\" encoding=\"UTF-8\" "+
                "standalone=\"no\"?>"+
        "<cfdi:Voucher "+
                "Concept=\"Bill of a eating.\" "+
                "Currency=\"euro\" "+
                "Date=\"Thu May 14 08:47:40 BST "+
                        "2020\" "+
                "Location=\"39032\" "+
                "SubTotal=\"21.357588898891848\""+
                "TaxRate=\"0.32\" "+
                "Total=\"28.19201734653724\" "+
                "Type=\"income\" "+
```

```
                    "Use=\"G01\">"+
                    "<cfdi:Issuer "+
                            "IssuerName=\"el patio de "+
                                    "lola\" "+
                            "IssuerRFC=\"1000000\"/>"+
                    "<cfdi:Receiver "+
                            "ReceiverName=\"Julian "+
                                    "Bell\" " +
                            "ReceiverRFC=\"3000008\"/>"+
        "</cfdi:Voucher>"))
);
```

Por otro lado, en cuanto a la suscripción de eventos al tanque de las facturas procesadas, como podemos apreciar en el Código 5.3, la suscripción a un terminal consiste en definir una acción a realizar con la factura procesada a través de un evento. Es decir, siendo action un ejemplo, se puede hacer uso de cualquier método en ese evento, el cual tratará la factura procesada que llegue. El segundo parámetro es un identificador único que debe tener cada cliente.

Código 5.3: Suscripción al Datahub en local

```
DatamartTerminal dt = new DatamartTerminal(connector);

dt.subscribe(processedInvoice ->
        action(processedInvoice),
        "finbook-processedInvoices");
```

5.2. Diseño de los datos

Los datos utilizados son única y exclusivamente las facturas. Las facturas están compuestas por diferentes campos, los cuales ya estaban definidos en el momento en el que se ha desarrollado el *Datahub*.

Como se ha indicado con anterioridad, las facturas están en formato XML. Se ha elegido un formato y un conjunto de datos que las facturas deben tener para que puedan ser tratadas de la misma forma. Generalmente, el formato de una factura utilizada por FinBook deberá tener el aspecto mostrado en el Código 5.4.

Código 5.4: Ejemplo de Factura

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<cfdi:Voucher
        Concept="Bill of a eating"
        Currency="euro"
        Date="Tue Apr 14 00:26:05 BST 2020"
        Location="Calle Las Pitas 6"
        SubTotal="128.67024903648075"
        TaxRate="1.0"
        Total="128.67024903648075"
        Type="income"
>
        <cfdi:Transmitter
                IssuerName="Restaurante Nito's"
                IssuerRFC="1000002"
        />
        <cfdi:Receiver
                ReceiverName="Liam Coates"
                ReceiverRFC="3000423"
        />
</cfdi:Voucher>
```

Si despiezamos esta factura, tenemos los siguientes campos:

- UUID: id único de una factura.
- Concept: motivo de la factura.
- Currency: divisa con la que se ha cobrado la factura.
- Date: fecha de generación de la factura.
- Location: lugar de generación de la factura.
- SubTotal: total a pagar sin aplicar la tasa de impuesto.
- TaxRate: tasa de impuesto aplicada a la factura.
- Total: total a pagar con la tasa de impuesto aplicada.
- Type: representa el tipo de factura, que puede ser ingreso, egreso, o nómina.

- Use: es el uso que se le da a la factura, representado por los códigos proporcionados por CFDI. https://www.cfdi.org.mx/catalogos-de-cfdi/uso/
- IssuerName: nombre de la empresa que generó la factura
- IssuerRFC: Registro Federal de Contribuyentes (RFC) de la empresa que generó la factura.
- ReceiverName: nombre de la empresa que recibió la factura.
- ReceiverRFC: RFC de la empresa que recibió la factura.

Tras saber cómo es la forma de los datos utilizados en FinBook, se deben definir los eventos que representan a los mismos. Para ello, se ha utilizado un lenguaje proporcionado por Intino: Ness.

En primer lugar, como podemos apreciar en el Código 5.5, tenemos la definición del evento de facturas en bruto, que consta únicamente de un atributo en el que se almacenará el XML en base 64 de la misma.

Código 5.5: Definición de la factura en Ness

```
Event Invoice
        Attribute xml as Text
```

Por otro lado, como podemos ver en el Código 5.6, tenemos la definición del evento de las facturas procesadas, el cual dispone de todos los campos definidos en el modelo, además de un campo XML en el que se almacenará el XML original, codificado en base 64.

Código 5.6: Definición de la factura procesada en Ness

```
Event ProcessedInvoice
        Attribute UUID as Text
  Attribute date as DateTime
  Attribute PC as Integer
  Attribute type as Text
  Attribute Use as Text
  Attribute Concept as Text
  Attribute issuerName as Text
  Attribute issuerRFC as Text
  Attribute receiverName as Text
  Attribute receiverRFC as Text
  Attribute subtotal as Real
```

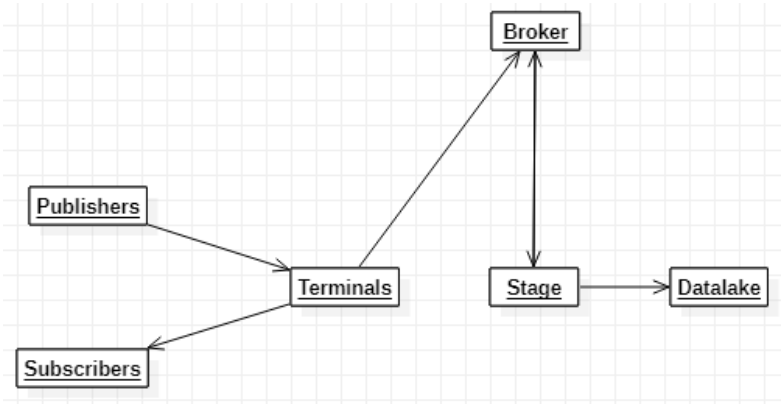

Figura 5.1: Flujo de datos producido en el Datahub.

```
Attribute taxRate as Real
Attribute total as Real
Attribute currency as Text
Attribute xml as Text
```

5.3. Diseño Modular

La estructura de almacenamiento de datos del *Datahub* está dada por el sistema de ficheros del servidor en el que se almacenen los datos. Como podemos observar en la Figura 5.1, los usuarios de este *Datahub*, es decir, los *publishers* y *subscribers*, tienen acceso únicamente a los terminales que a efectos prácticos, es un directorio en el que los *publishers* ponen los nuevos eventos y en el que los *subscribers* leen los eventos nuevos publicados que les interese.

El flujo de los eventos comienza cuando los *publishers* publican los eventos en los *terminals*. Tras esto, el *broker* se percata de la existencia de estos nuevos eventos y se los lleva al *stage*. Una vez llegan al *stage*, los *subscribers* son notificados de estos eventos a través de los terminales a los que esté suscrito. Por último, cabe destacar que los eventos del *stage* son sellados periódicamente en el *Datalake*. Este proceso es muy costoso, y es por eso que no se suele realizar en tiempo real: el sellado se realiza en las horas en las que la aplicación tiene una actividad menor.

En cuanto a la definición de estos elementos en el código del proyecto, se ven reflejados en el fichero Solutions.tara, en el que se han definido haciendo

uso del lenguaje Ness. Como se puede apreciar en el Código 5.7, se encuentra la definición del *broker*, en el que se han definido distintos tipos de usuario que disponiendo de esta información, pueden usar la aplicación. Cabe destacar que la definición del usuario está dada por el id del usuario y su contraseña, que a modo de ejemplo, he usado el mismo para ambos campos. El primer lugar, he definido un usuario sensor, que es aquel que de manera externa al *Datahub*, publicará los datos: es decir, son aquellos módulos que publican facturas brutas. El segundo usuario definido es el *Datamart*, que son aquellos que consumen los datos publicados en el tanque de las facturas procesadas. El tercer y último usuario el es *handler*, cuya información de usuario ha sido distribuido únicamente en el módulo que convierte la factura sin procesar a procesada.

Código 5.7: Definición del broker en Ness

```
Broker(path = empty, port = empty,
        secondaryPort = empty)
  User("io.finbook.sensor", "io.finbook.sensor")
  User("io.finbook.datamart", "io.finbook.datamart")
  User("io.finbook.handler", "io.finbook.handler")
```

Tras la definición del *broker*, como se puede apreciar en el Código 5.8, se ha definido la propia estructura de datos. El primer aspecto a destacar es la escala, que se ha elegido que sea mensual: es decir, los eventos se agruparán mensualmente. Tras esto, se ha definido un contexto. Este proyecto es muy pequeño y con uno bastaba, pero el definir múltiples contextos permite definir de manera más estricta quién accede a cada tanque, y es muy útil en proyectos más grandes. Tras esto, se encuentran las definiciones de los dos tanques, que constan de los siguientes campos: en primer lugar, se define qué tipo de eventos se almacenan en cada tanque, tras esto, se define el nombre del tanque, y por último, se define el contexto, que en este caso es el mismo pero podrían ser distintos si se definen múltiples contextos. Por último, se ha definido un *Cron*, que consiste en marcar los momentos según la definición dada en el *pattern* en el que se realizará el sellado de los datos en el *Datalake*.

Código 5.8: Definición del Datalake en Ness

```
Datalake(scale=Month, path = empty) dl
  Context io
     Context finbook

  Tank(event = Invoice) invoice as
                 Event Contextual(io.finbook)
```

```
Tank(event = ProcessedInvoice) processedInvoice as
                Event Contextual(io.finbook)

Seal > Cron(pattern = "0 0/1 * 1/1 * ? *",
                timeZone = "Atlantic/Canary") cron
```

Por último, como está indicado en el Código 5.9, se han definido los terminales, los cuales, una vez exportado el proyecto, generan unos Java ARchive (JAR) que se pueden importar en cualquier proyecto para hacer uso de estos tanques. Se han definido tres terminales, y cada uno corresponde a cada uno de los usuarios: en el caso de los sensores, solo puede publicar en el tanque "invoices", en el caso de los *Datamarts*, solo se pueden suscribir al tanque "processedInvoices", y en el caso del "handler", se suscribe a "invoices", y publica en "processedInvoices".

Código 5.9: Definición de los terminales en Ness

```
Terminal sensorTerminal
  Publish(dl.invoices)

Terminal datamartTerminal
  Subscribe(dl.processedInvoices)

Terminal handlerTerminal
  Publish(dl.processedInvoices)
  Subscribe(dl.invoices)
```

En cuanto a la arquitectura utilizada, en este módulo se ha hecho uso de una arquitectura *Big Data* y del patrón de diseño *Publisher/Subscriber* para la comunicación a través de los terminales.

En el caso del módulo del propio *Datahub*, no hay clases Java más allá de la definición de un main para configurar el lanzamiento de la aplicación, las clases generadas por el propio Intino, las definiciones de los datos que hemos visto con anterioridad, y la configuración del proyecto.

Por otro lado, en el módulo que procesa las facturas, tenemos además las clases necesarias para acceder a los eventos a manejar, además de una clase "XMLHelper", la cual se encarga de extraer la información de la factura XML en datos manejables por Java, haciendo uso de la librería "dom", de "org.w3c" (https://docs.oracle.com/javase/7/docs/api/org/w3c/dom/package-summary.html).

En el desarrollo de este *Datahub*, se ha pretendido cumplir los principios *Single responsibility, Open-closed, Liskov substitution, Interface segregation and Dependency inversion* (SOLID):

- Principio de responsabilidad única: este punto suele ser analizado sobretodo en las clases y métodos. En este caso, las clases Java no son muy extensas, pero la aplicación de este punto se puede ver claramente en la división del módulo del *Datahub* con el del procesador de facturas.
- Principio de abierto-cerrado: el cumplimiento de este punto se ve sobretodo con el uso de los terminales, y es que para que otros módulos nuevos hagan uso de este *Datahub*, no es necesario modificar el *Datahub* en ningún momento.
- Principio de sustitución de Liskov: el uso de Intino hace que las clases desarrolladas sean extensiones de un conjunto de clases abstractas que genera. Esto permite que estas clases realicen las funciones estipuladas en la clase padre siguiendo una serie de criterios, y en cualquier momento se puede crear una nueva clase que herede de esta clase padre y que realice en esencia las mismas acciones.
- Principio de segregación de interfaces: en este caso en el que el programa es muy pequeño, las interfaces no están presentes.
- Principio de inversión de dependencias: el uso de Intino permite que el programa funcione a nivel de clases abstractas, sin que las clases principales conozcan las clases que heredan y las implementaciones concretas.

5.4. Diseño Arquitectónico

Para poder ejecutar este *Datahub*, en primer lugar, haría falta un Sistema Operativo: ha sido probado en un Windows 10 Pro (https://www.microsoft.com/es-es/software-download/windows10), y en un Ubuntu 18.04.3 LTS (https://lists.ubuntu.com/archives/ubuntu-announce/2019-August/000248.html). Sea cual sea la forma de ejecutar la aplicación, hace falta la Java VM 11(https://jdk.java.net/11/) o superior. En el caso de ejecutar el *Datahub* desde un *Integrated Development Environment* (IDE), debe hacerse desde Intellij, el cual en mi caso, ha sido la versión Community (https://www.jetbrains.com/es-es/idea/download): esto es porque el *framework* Intino está

preparado para aprovechar al máximo las funcionalidades que proporciona este IDE. Además, se requerirá la instalación de Intino 5.1.0. Además, requiere tener instalado Maven (https://maven.apache.org/download.cgi), para poder añadir al equipo muchas de las dependencias necesarias en este caso.

Capítulo 6

Firma

Debido al sistema de acceso implementado en la plataforma, ha sido necesario crear una pequeña aplicación de escritorio que sea capaz de firmar un texto automáticamente y traer de vuelta esta firma, y es por ello que ha surgido la necesidad de desarrollar la aplicación "Firma". Cuando un usuario desea acceder a FinBook, se requiere de cierta seguridad, y es por eso que se ha desarrollado un sistema en el que el *login* se realice a través de una firma digital, y la idea es que Firma sea la aplicación que realice esa firma.

Cuando el usuario decide "loguearse", la aplicación FinBook accede a Firma, previamente instalado en el equipo, y lo hace haciendo uso de un protocolo "finbSign", definido especialmente para ejecutar esta aplicación (como hacen muchas páginas web para abrir uTorrent, por ejemplo). Esta petición contiene el protocolo, el texto a firmar, y la *Uniform Resource Locator* (URL) del *web socket* al que debe enviar la respuesta. Firma, haciendo

Figura 6.1: Logo de Firma.

uso de la localización de la clave privada del usuario y del certificado del mismo especificado en los archivos de configuración, firma el texto, y envía este texto firmado al *web socket* mencionado.

En muchas aplicaciones en las que el acceso o la realización de una acción concreta está dado por el uso de una firma digital, puede llegar a ser algo peligroso el pedir buscar la clave privada en el equipo y seleccionarla para firmar online, además de engorroso, ya que habría que estar buscándola cada vez que se quiera acceder a la propia página o realizar dicha acción. Es por eso que en la realización de Firma se ha tenido en cuenta dos aspectos muy importantes, que son fundamentales de cara a aportar valor al usuario: la seguridad, y la agilización de la acción de acceder a la aplicación.

Por el lado de la seguridad, cabe destacar que de la forma en la que está hecho este sistema de *login*, la clave privada del usuario, la cual no debe ser compartida con nadie, no es transferida en ningún tipo de petición web: se usa exclusivamente en Firma, y simplemente se utiliza para firmar el texto. Sabiendo esto, el usuario tiene la seguridad de que ningún *sniffer* pueda acceder a la red del usuario, y hacerse con la contraseña introducida para hacer uso de la clave privada, o de la propia clave privada.

Por otro lado, en cuanto a la agilización del acceso, debemos tener en cuenta que casi todo el tiempo que se tarda en acceder en otras aplicaciones, está dado por el hecho de estar buscando la clave privada en el equipo, o incluso los certificados, que normalmente están guardados en la configuración del navegador. La idea es tener estos dos componentes almacenados en disco, y tener almacenado su localización. Esta localización es almacenada en los ficheros de configuración, y basta con que el usuario defina una sola vez la localización de la clave privada y el certificado. A partir de ese momento, a efectos prácticos, el usuario al "loguearse", simplemente debe introducir la contraseña de la clave privada pedida por Firma, y el resto lo realiza la propia aplicación.

6.1. Diseño de la interfaz

El objetivo de esta interfaz es simple: mostrar al usuario las localizaciones de los ficheros necesarios (clave privada y certificado), pedir la contraseña de la clave privada al usuario, y notificar al mismo de cualquier error sucedido en el proceso de firmado.

Figura 6.2: Interfaz de usuario de Firma.

Debido a estas necesidades, se ha optado por realizar una interfaz lo más sencilla posible, sin exceso de opciones adicionales, ya que el objetivo es no hacer complicada la experiencia al usuario.

Como se puede apreciar en la Figura 6.2, se ha optado por mostrar el logotipo de Firma relativamente grande para que se sepa en todo momento la aplicación que se ha estado usando, se ha optado por comprimir lo máximo posible la zona de texto, para no hacer la interfaz excesivamente grande, y los colores que se han utilizado están en armonía con los colores del logotipo, y es una combinación de colores agradable a la vista.

6.2. Diseño de la API

El acceso a la aplicación está preparado para recibir un solo argumento procedente desde la web. Este argumento viene dado por la siguiente forma: finbSign: textToSign wsurl, siendo finbSign el protocolo, textToSign el texto a firmar, y wsurl la URL del *web socket* al que se debe enviar la respuesta.

6.3. Diseño de los datos

En esta aplicación de Firma, se manejan varios conceptos importantes:

- El usuario que quiera usar Firma, deberá disponer de su clave privada y su correspondiente certificado. El certificado debe seguir el estándar "X.509".
- A la aplicación se le pasará como parámetro un texto cualquiera a firmar, el cual no tiene ningún formato especial.

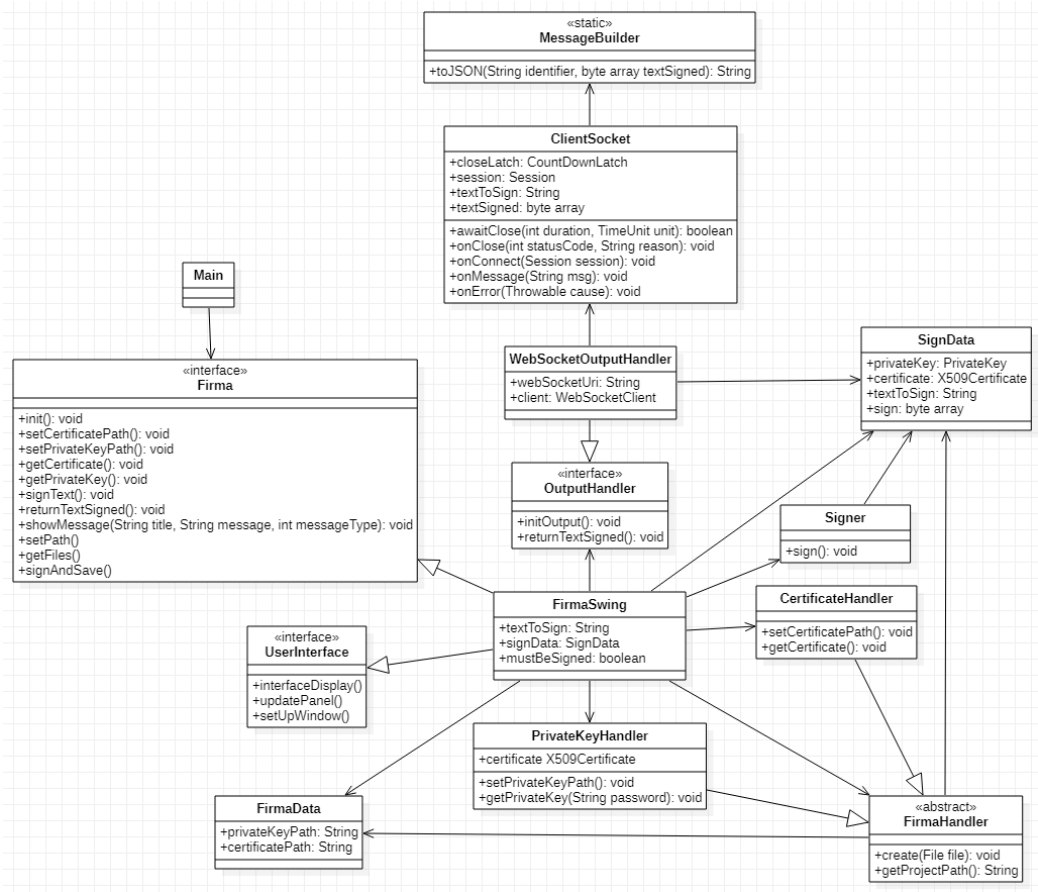

Figura 6.3: Diagrama de clases de Firma.

- Además, un segundo parámetro que necesita la aplicación es la URL del web socket, que tendrá un formato similar a "ws://echo.websocket.org".

6.4. Diseño modular

En el diseño de Firma, se ha seguido un diseño modular en el que las acciones se realizan siguiendo una serie de pasos. En resumen, los pasos a seguir son: búsqueda de la clave privada y el certificado, firmar el texto, y enviarlo mediante un *web socket* en forma de *JavaScript Object Notation* (JSON). Las clases implementadas y sus relaciones están dadas por el diagrama representado en la Figura 6.3.

El uso de interfaces y de clases abstractas ha sido abundante teniendo en cuenta lo pequeña que es la aplicación y las reducidas funcionalidades que ofrece.

Podemos ver por ejemplo el desarrollo de la interfaz "Firma" y de "UserInterface". La primera ofrece métodos que orientan a implementar las funcionalidades siguiendo el flujo de acciones que se deben realizar en Firma. Sin embargo, "UserInterface" ofrece métodos orientados a la visualización de elementos gráficos. La clase FirmaSwing implementa estos dos, y es una clase cuya función es realizar las acciones básicas del flujo de Firma, por eso implementa "Firma"; y además dispone de elementos gráficos, por lo que debe implementar "UserInterface". En este caso, se ha elegido Swing para implementar la vista, ya que las necesidades visuales de esta aplicación eran escasas, acorde al potencial de esta librería.

En el caso de la interfaz "OutputHandler", se ha diseñado de modo que en un futuro, se puedan añadir otras clases que propongan otro tipo de salidas para la firma, como podría ser un fichero. En este caso, bastaba con diseñar una en la que se haga uso de un *web socket*.

Los datos principales manejados por el programa han sido descritos en el apartado anterior. No obstante, Firma hace uso de dos clases que modelan este proceso: "FirmaData" para modelar la configuración del programa, y "SignData" para modelar la firma, ya que como podemos observar, contiene la clave privada, el certificado, el texto a firmar, y el texto ya firmado.

El flujo del programa comenzaría en "FirmaSwing". En primer lugar se debe cargar los *paths* de la clave privada y del certificado almacenadas en los ficheros de configuración, y las clases encargadas de realizar esta acción son "PrivateKeyHandler" y "CertificateHandler". Estas mismas clases son las que posteriormente, obtienen tanto la clave privada como el certificado haciendo uso de estos *paths*. Tras esto, se le pide al Signer que firme el texto inicial haciendo uso de estos dos elementos, y es en este punto en el que se le pide al usuario la contraseña de la clave privada. Esta interacción es la única que realiza el usuario en todo el proceso. El texto es firmado, y la instancia de la clase "SignData" ya está listo para extraer este texto firmado y enviarlo a través del *web socket*. Para ello se encarga "WebSocketOutputHandler" y "ClientSocket". Antes de ser enviado, "ClientSocket" hace uso de "MessageBuilder" para crear un JSON con campos bien conocidos por la plataforma web que recibe el mensaje a través del *web socket*.

En el desarrollo de Firma, se ha pretendido cumplir los principios SOLID:

- Principio de responsabilidad única: se ha decidido optar por crear todas las clases que fueran necesarias, por pequeñas que sean, para así liberar de responsabilidades extra a las mismas. Si bien es verdad que "FirmaSwing" se encarga tanto de mostrar la interfaz como de inyectar las dependencias en otras clases, hecho así para facilitar la ejecución del programa; la mayoría de clases tienen responsabilidades reducidas, como es el caso de "Signer", que únicamente debe firmar un texto.

- Principio de abierto-cerrado: con el uso de interfaces, hace que no sea estrictamente necesario modificar las clases cuando se necesite un cambio, bastaría con crear nuevas clases con los nuevos comportamientos.

- Principio de sustitución de Liskov: este principio se ve cumplido sobretodo en la interfaz "OutputHandler", que dispone de unos métodos determinados que en cualquiera de sus implementaciones, realizaría acciones similares. De hecho, en el caso en el que se tiene que enviar el texto firmado, se hace a través de la interfaz.

- Principio de segregación de interfaces: se ha procurado que las interfaces dispongan de métodos que se van a usar. Como se ha mencionado antes, se ha separado incluso las interfaces "Firma" y "UserInterface" para cumplir con este principio. De este modo, se podría realizar una interfaz, o una clase que se encargue de inyectar dependencias sin interfaz perfectamente, sin la necesidad de disponer de métodos que no van a usar.

- Principio de inversión de dependencias: en este caso, Firma no es tan grande como para hacer que interaccionen clases de alto nivel entre sí.

6.5. Diseño arquitectónico

Para poder ejecutar Firma, en primer lugar, haría falta un Sistema Operativo: ha sido probado en un Windows 10 Pro, y en un Ubuntu 18.04.3 LTS. Sea cual sea la forma de ejecutar la aplicación, hace falta la Java VM 11 o superior. En el caso de ejecutar la aplicación desde un IDE, está recomendado hacerlo desde Intellij, el cual en mi caso, ha sido la versión Community, aunque siempre se puede adaptar a otro IDE. Además, requiere tener instalado Maven, para poder añadir al equipo todas las dependencias necesarias en este caso.

Capítulo 7

SignVerifier

SignVerifier se trata de una librería desarrollada como apoyo tanto a la aplicación de FinBook como a la de Firma. El objetivo principal del desarrollo de esta librería es que los distintos módulos destinados al acceso a la plataforma de FinBook, tengan alguna manera de verificar el acceso producido a la aplicación. Es decir, dispone de las funciones necesarias para, dado un texto firmado en forma de *array* de bytes, poder extraer el certificado adjunto a ese texto, aplicarlo al mismo, y comprobar que el texto descifrado con el certificado coincide con el original. Además de esta comprobación, debe realizar otra muy importante, y es comprobar que el certificado está generado por una entidad de confianza como podría ser la Fábrica Nacional de Moneda y Timbre (FNMT) en España. Además de realizar esta función, dispone de otras funciones útiles para la aplicación FinBook, y es por ejemplo, generar un texto aleatorio, que es el que se envía a la aplicación Firma, pasar un texto a base 64, o pasar un texto que está en base 64 a texto codificado con caracteres *American Standard Code for Information Interchange* (ASCII). Las funciones para pasar un texto normal a base 64 y viceversa, serán usados tanto desde el método de generar un texto aleatorio, como en la propia aplicación de FinBook al manejar facturas, que deberán ser tratadas en base 64.

7.1. Diseño de la API

En cuanto a la clase FileHandler, invocado con el *path* de un fichero, dispone de un método readByteArray() que devuelve un *array* de bytes leídos

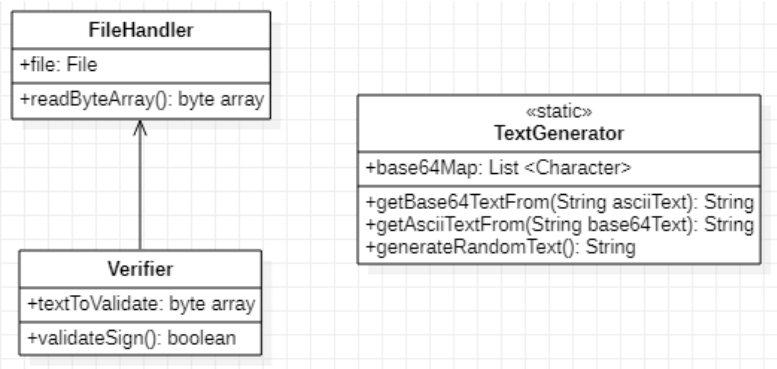

Figura 7.1: Diagrama de clases de SignVerifier.

del fichero especificado en el constructor de FileHandler. Verifier, que se instancia pasándole un *array* de bytes, dispone de un método, validateSign(), que valida la firma que contiene este *array* de bytes. Si la validación es correcta, devuelve el campo RFC del certificado con el que se firmó. Por otro lado, si no es posible validar la firma ya sea porque la propia firma no es correcta, porque el certificado no esté autorizado por una entidad autentificadora, o porque el certificado no tenga un campo RFC, el método devolverá un valor nulo. Por otro lado, TextGenerator dispone de tres métodos:

- getBase64TextFrom(String asciiText): este método codifica un texto cualquiera compuesto por caracteres ASCII a base 64.
- getAsciiTextFrom(String base64Text): su función es decodificar un texto en base 64.
- generateRandomText(): devuelve un texto que se genera a partir de un número aleatorio de entre el 0 al 100.000, que posteriormente se pasa a String, y se codifica en base 64.

7.2. Diseño modular

Para la realización de esta librería, se ha hecho uso de nuevo de las librerías de bouncycastle (https://www.bouncycastle.org/java.html), que disponen de un conjunto de clases que permiten una gestión muy sencilla de la verificación de la firma. Al tratarse de una librería de reducidas funcionalidades, su diagrama de clases es mínimo, el cual está representado en la Figura 7.1.

Para realizar las funciones de conversión de un texto sin base a uno de base 64 y viceversa, he decidido hacerlo por mí mismo, y no hacer uso de librerías externas, ya que de esta manera tengo un control total sobre las operaciones realizadas y reduciría las dependencias con paquetes externos, y podía ser una oportunidad para aprender cómo codificar un texto compuesto por caracteres ASCII a base 64.

Base 64 es un sistema de numeración posicional que utiliza 64 como base, es decir, un texto en base 64 puede tener hasta 64 caracteres diferentes. Sin embargo, un texto sin base está representado con caracteres disponibles en el código de caracteres ASCII, que dispone de 128 caracteres diferentes. Para realizar la conversión de un texto sin base (codificado en ASCII) a uno con base 64, en primer lugar, se debe pasar el texto a un *array* de bits: para ello se debe ir obteniendo letra por letra, y mapearla a su versión decimal según la tabla de caracteres ASCII. Esta versión decimal se pasa a bits (cada carácter está compuesto por 8 bits), y se rellena este *array*. Una vez el texto completo ha sido convertido a un *array* de bits, para obtener el texto en base 64, se deben ir tomando de 6 bits en 6 bits, y esos 6 bits, pasados a base decimal, deben equivaler a un número entre 0 y 63, el cual corresponde directamente con un carácter de la base 64, tal y como se muestra en la Figura 7.2. Finalmente, al unir todos estos caracteres obtenidos del *array* de bits en el mismo orden en el que se encuentran los bits de los que se extrajeron, se forma un texto, que es el equivalente al original, pero en base 64.

Además, hay que tener en cuenta un pequeño detalle: en ciertas ocasiones, al texto ya codificado en base 64 tiene un pequeño relleno compuesto por caracteres '='. Esto se realiza cuando el número de bits totales al pasar el texto original a *array* de bits no es múltiplo de 6. El número de '=' sigue una regla simple: si el número de bits + 2 es múltiplo de 6, el relleno se compone de un solo '='; por otro lado, cuando el número de bits - 2 es múltiplo de 6, son dos '=' los que se utilizan. Este relleno funciona a modo de sufijo, es decir, se concatena por el final del texto.

Base64 Encoding Table

Value	Char	Value	Char	Value	Char	Value	Char
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

Figura 7.2: Tabla de codificación de base 64.
Fernando, Isuru. (2019). Base64 Encoding and Decoding in SAP ABAP. [Figura]. Recuperado de https://sapintegrationhub.com/abap/base64/base64-encoding-and-decoding-in-sap-abap/.

Capítulo 8

Metodología de trabajo

El punto más importante sobre el que se basa este Trabajo de Fin de Grado es el trabajo en equipo. Como se ha explicado con anterioridad, el proyecto dispone de un conjunto de módulos bien diferenciados entre sí, y su manera de integrarse permite el desarrollo y las pruebas de forma independiente. Gracias a este factor y debido a la dimensión del proyecto, se ha permitido el desarrollo asignando cada uno de los módulos a un integrante. Como bien se ha especificado, cada módulo es independiente del resto y si bien idealmente estos deben comunicarse entre sí haciendo uso de unos terminales, son totalmente funcionales por sí solos, o haciendo uso de datos *mock*.

Para la realización del trabajo, se han llevado a cabo reuniones cada dos semanas con los demás integrantes del proyecto y con el tutor, ya que si bien cada parte funciona de manera independiente, debíamos ponernos de acuerdo en ciertos aspectos de diseño como el formato de las facturas, ya que en mi caso debía realizar la conversión en el módulo del *Datahub*, cómo se llevaría a cabo la seguridad de la aplicación, y otros aspectos relacionados con el rumbo que debía llevar *FinBook*. Esta forma de trabajo viene dada por la naturaleza del proyecto, que incitaba a trabajar de una manera iterativa e incremental, en el que se han establecido una serie de objetivos/hitos a cumplir en cada iteración.

Se ha usado una arquitectura de programación funcional, aplicando en parte un desarrollo guiado por pruebas haciendo uso de *Test-Driven Development* (TDD) para probar ciertos módulos, y se ha recurrido a la ayuda de los creadores del *framework* de Intino para solucionar problemas a la hora de desarrollar con este *framework*.

Cabe destacar el uso de Java como lenguaje de programación principal, haciendo uso de IntelliJ como IDE. Para facilitar el trabajo a la hora de realizar tanto la plataforma de datos como el sistema de mensajería entre módulos, se ha usado el framework Intino. Cabe destacar que Intino usa la tecnología JMS que consiste en una solución para el uso de colas de mensajes: es un estándar de mensajería que permite a los componentes de aplicaciones crear, enviar, recibir, y leer mensajes, y que permite una comunicación confiable de manera asíncrona.

Para el control de versiones del proyecto se utilizará la tecnología Git (https://git-scm.com/), y dada la naturaleza grupal del proyecto, se aprovechará el modelo de ramificación GitFlow para facilitar el desarrollo concurrente y organizado por parte de todos los integrantes del proyecto. Además, cabe destacar el uso de Maven como gestor de dependencias, ya que facilita mucho trabajo en este aspecto, y la inclusión de estas librerías se hace mucho más llevadero. No obstante, en los módulos en los que se ha usado Intino, también existe un archivo, llamado "artifact" que se utiliza para configurar el módulo y entre otras cosas, también permite definir dependencias.

8.1. Iteraciones

En mi caso, comencé por recordar el funcionamiento de las firmas digitales, y me informé acerca de librerías que pudieran ayudarme a realizar las operaciones necesarias en relación con este aspecto. Comencé creando un pequeño módulo en el que pudiera firmar un texto e inmediatamente se compruebe la validez de la firma. Tras lograr esto, separé la parte en la que se firma y le añadí una interfaz (Firma), de la parte en la que esta se comprueba (SignVerifier). Mientras tanto, hablaba con mis compañeros para integrar lo desarrollado: la librería SignVerifier no tenía problema, ya que bastaba con que ellos la usaran. No obstante, en cuanto a Firma, se llegó al acuerdo que la aplicación debía enviar la respuesta haciendo uso de un *web socket*, funcionando primeramente almacenando la firma en un fichero para realizar las distintas pruebas.

Tras pulir estos dos módulos y ya definido el formato de los datos, me dispuse a realizar el *Datahub*. Para ello he estudiado varios proyectos que hacen uso de Intino, y he ido adaptándolo poco a poco según las necesidades, realizando también muchas consultas conceptuales a los desarrolladores de este *framework*. Tanto las facturas como las nóminas tienen el mismo formato, así que en el aspecto de la implementación, he podido utilizar el mismo código

para tratar ambas. La razón por la cual he dejado la realización de esta plataforma de datos es para darle más tiempo a mis compañeros para pensar acerca del diseño de las facturas. Una vez realizada esta plataforma de datos y realizar sus respectivas pruebas de rendimiento, me dispondré a desplegar esta plataforma para que pueda ser usada desde cualquier equipo, y poder realizar las pruebas de integración mencionadas.

Tras la realización del proyecto, las siguientes iteraciones constaban de revisión de la memoria, o planificación para la presentación ante el tribunal.

8.2. Herramientas utilizadas

Las herramientas utilizadas durante la realización del proyecto son las siguientes:

- Windows 10 Pro: es el sistema operativo principal sobre el que se ha desarrollado los distintos módulos.
- Ubuntu 18.04.3 LTS: se ha utilizado como sistema operativo secundario para realizar pruebas de compatibilidad.
- Java 11.0.2: lenguaje de programación principal.
- Intellij Community 2019.3.4: IDE utilizado durante el desarrollo.
- Git: sistema utilizado para llevar a cabo el control de versiones.
- GitFlow: herramienta utilizada que permite de una manera sencilla el desarrollo concurrente del proyecto.
- StarUml: aplicación para la elaboración de los distintos diagramas necesarios (http://staruml.io/).
- Intino 5.1.2: *framework* principal sobre el que se ha desarrollado la plataforma de datos.
- Putty: aplicación que permite la conexión con servidores (https://www.putty.org/).

8.3. Discusión

En lo personal, este desarrollo orientado a lo iterativo e incremental, me parece muy interesante. Creo que para trabajar en equipo, la mejor técnica es aquella en la que la organización de las tareas es fundamental, además de llevar una fuerte organización temporal, ya que es muy común que subestimar las tareas a realizar, y al final se acaba desarrollando todo con prisas y en unas condiciones que no son ideales. Dejando claro este aspecto, por otro lado, creo que si hubiéramos seguido un método de trabajo más parecido a SCRUM, se hubieran retrasado muchísimo con las entregas: SCRUM es una forma en principio ideal para realizar proyectos en los que existe un usuario final que desea comprar este producto, e incluso creo que en ocasiones se puede organizar el trabajo de otra manera, sobretodo si la cantidad de funcionalidades no es excesiva. En el caso de este proyecto, no hubiera tenido sentido hacer uso de esta forma de trabajar, ya que no existe un usuario real que imponga unas funcionalidades.

Por otro lado, en cuanto al TDD, creo que es una técnica muy interesante que permite que no fallemos a la hora de desarrollar una funcionalidad. Se ha podido hacer uso de esta técnica en ciertas ocasiones en las que su uso estaba muy claro, como es el caso de probar las funciones para convertir un texto a base 64. No obstante, también estoy a favor de otros métodos de desarrollo de pruebas como son los *code reviews*, en los que otros compañeros revisan el código realizado por nosotros en busca de posibles fallos.

Por último, destacar que me he sentido muy cómodo con el desarrollo de una aplicación dividida en módulos. Creo que este desacoplamiento es la forma ideal para hacer que la arquitectura de cualquier proyecto sea lo más limpia y reutilizable posible. Además, se hace más fácil el reparto de trabajo, y de esta manera, no se producen contratiempos a la hora de unir las distintas partes desarrolladas por el equipo.

Capítulo 9

Competencias cubiertas

Las competencias que han sido cubiertas con la realización de este Trabajo de Fin de Grado son:

TFG01: Ejercicio original a realizar individualmente y presentar y defender ante un tribunal universitario, consistente en un proyecto en el ámbito de las tecnologías específicas de la Ingeniería en Informática de naturaleza profesional en el que se sinteticen e integren las competencias adquiridas en las enseñanzas.

Este requisito es común a todos los Trabajo de Fin de Grado (TFG), y por el hecho de realizar el trabajo y con su posterior presentación, la competencia quedaría cubierta.

CII01: Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.

Durante la realización de este trabajo, se ha diseñado y desarrollado distintos elementos importantes en la estructura de *FinBook*, como es su *Datahub*. Con su desarrollo, se ha procurado hacerlo de modo que cumpla los estándares necesarios si esta aplicación la utilizara una agencia tributaria.

CII05: Conocimiento, administración y mantenimiento sistemas, servicios y aplicaciones informáticas.

Se ha visto sobretodo con el control de la aplicación a lo largo de su desarrollo, sobretodo en el despliegue del *Datahub* en el servidor, el cual

posteriormente debe ser mantenido correctamente para asegurar el correcto funcionamiento y la comodidad de los usuarios.

CII07: Conocimiento, diseño y utilización de forma eficiente los tipos y estructuras de datos más adecuados a la resolución de un problema.

Esta competencia puede verse cubierta teniendo en cuenta la decisión del uso de *Big Data* como arquitectura principal de la aplicación, además del uso del patrón *Publisher/Subscriber*.

CII012: Conocimiento y aplicación de las características, funcionalidades y estructura de las bases de datos, que permitan su adecuado uso, y el diseño y el análisis e implementación de aplicaciones basadas en ellos.

El proyecto está basado casi en su totalidad en el manejo de datos en un sistema de gestor de datos de una aplicación basada en una arquitectura *Big Data*.

CII013: Conocimiento y aplicación de las herramientas necesarias para el almacenamiento, procesamiento y acceso a los Sistemas de información, incluidos los basados en web.

La aplicación Firma hace uso de *web sockets* para conectarse con la plataforma web de *FinBook*.

CII016: Conocimiento y aplicación de los principios, metodologías y ciclos de vida de la ingeniería de software.

Las distintas partes desarrolladas han seguido un ciclo de vida iterativo e incremental, realizando reuniones entre estos incrementos con los compañeros que han desarrollado otros módulos y con el tutor.

IS03: Capacidad de dar solución a problemas de integración en función de las estrategias, estándares y tecnologías disponibles.

Esta competencia se ha visto cubierta con la integración de las distintas partes desarrolladas por mis compañeros y la mía, haciendo uso de los terminales que ofrecen el *Datahub*.

IS04: Capacidad de identificar y analizar problemas y diseñar, desarrollar, implementar, verificar y documentar soluciones software sobre la base de un conocimiento adecuado de las teorías, modelos y técnicas actuales.

Se ha identificado *Big Data* como una arquitectura interesante para el diseño, y se ha podido desarrollar haciendo uso de un *framework* externo que utiliza técnicas bastante actuales.

IS06: Capacidad para diseñar soluciones apropiadas en uno o más dominios de aplicación utilizando métodos de la ingeniería del software que integren aspectos éticos, sociales, legales y económicos.

En este caso, gracias al desarrollo de una aplicación *FinTech*, se ha podido integrar el ámbito tecnológico gracias al propio desarrollo con el ámbito económico proporcionado por el manejo de facturas y variables relacionadas con la economía.

Capítulo 10

Conclusiones

10.1. Resultados

10.1.1. Del proyecto

Como se ha expuesto a lo largo del proyecto, la constante digitalización de la gran mayoría de elementos del día a día lleva a pensar acerca del estado actual de las facturas, y es que el uso de las mismas impresas en papel aún es muy común. Durante la realización de este Trabajo de Fin de Grado, se ha intentado responder a una simple pregunta: ¿Es viable a día de hoy desarrollar un sistema para el fácil manejo de estas facturas digitales?

La respuesta es un sí, y muy claro además. Como se describirá a continuación, FinBook es un sistema que permite manejar una gran cantidad de facturas por segundo, por lo que es completamente apto para ser utilizado a nivel de grandes empresas o incluso estatal, como ocurre actualmente en México. Además, cabe destacar la poca cantidad de recursos que utiliza, por lo que realmente no necesita de una nueva era tecnológica para poder llevar esta digitalización de las facturas a cabo: es perfectamente viable con la tecnología actual.

Datahub

Para empezar, he creado un módulo extra muy básico ajeno al proyecto. El objetivo de este módulo es hacer uso de los terminales que ofrece el *Datahub*

para publicar un evento en el tanque de las facturas sin procesar, y recibir eventos de facturas procesadas. Este módulo no ha sido implementado con Intino, simplemente se le ha definido Maven como gestor de dependencias, haciendo uso de los terminales de la forma indicada en el Código 10.1 y en el Código 10.2. Como se puede apreciar, también hacen uso de la librería de alexandría, perteneciente a Intino, para poder hacer uso del Connector utilizado para establecer la conexión con el *Datahub*.

Código 10.1: Ejemplo del archivo pom.xml de un publicador

```
<dependency>
    <groupId>io.finbook</groupId>
    <artifactId>sign-verify</artifactId>
    <version>1.0</version>
</dependency>
<dependency>
    <groupId>io.intino.alexandria</groupId>
    <artifactId>core-framework</artifactId>
    <version>2.0.4</version>
</dependency>
<dependency>
    <groupId>io.finbook.datahub</groupId>
    <artifactId>sensor-terminal</artifactId>
    <version>2.0.0</version>
</dependency>
```

Código 10.2: Ejemplo del archivo pom.xml de un suscriptor

```
<dependency>
    <groupId>io.intino.alexandria</groupId>
    <artifactId>core-framework</artifactId>
    <version>2.0.4</version>
</dependency>
<dependency>
    <groupId>io.finbook.datahub</groupId>
    <artifactId>datamart-terminal</artifactId>
    <version>2.0.0</version>
</dependency>
```

Se han realizado pruebas en las que se han subido al *Datahub* 200,000 facturas seguidas, lo cual no es descabellado ya que según el Instituto Nacional de Estadística (INE), "el número de empresas activas el 1 de enero de 2019 se situó en 3,36 millones" (de Estadística, 2019), lo cual supone que estamos

hablando que aproximadamente una de cada 17 empresas en España envía una factura a la vez, lo cual puede suponer un buen número, ya que hay que tener en cuenta que muchas podrían enviar estas facturas en horarios de baja demanda (después del horario laboral) y debemos tener en cuenta que además de las múltiples empresas diurnas, existen muchas otras nocturnas, por lo que este envío de facturas estará más que repartido. Paralelamente, ha estado funcionando a la par un módulo suscriptor que muestra información acerca de esta factura por consola. Cabe destacar que en un caso de uso real, el sellado de las facturas se realizará en horarios de baja demanda, ya que consume una gran cantidad de recursos, así que se ha establecido que el sellado se realice todos los días a las 4 AM.

La prueba se ha realizado con el módulo de la plataforma de datos, un publicador, un suscriptor, y un módulo extra suscrito a los dos tanques que guardaba información en un fichero *Comma-Separated Values* (CSV). Cabe destacar que las pruebas se han realizado con un equipo con las siguientes especificaciones:

- Sistema Operativo: Windows 10 Pro
- Procesador: AMD Ryzen 5 1400 Quad-Core Processor 3.20 GHz
- RAM: 8 Gb

Tras la realización de esta prueba, tenemos los siguientes resultados:

A continuación se expondrán los distintos tiempos que ha tardado la aplicación en realizar las operaciones. Cabe destacar que hay varios, ya que cada publicación producida depende del ritmo de transmisión del bus de mensajería de Java y de su respectiva congestión.

En primer lugar, el tiempo que ha tardado la aplicación en recibir y publicar las 200,000 facturas producidas por el publicador ha sido de 21 minutos y 45 segundos. Esto en parte puede parecer bastante. Para saber cuál es la verdadera magnitud, vamos a darle la vuelta a este número y vamos a calcular cuántas facturas se han publicado por segundo: tenemos que el tiempo total es igual a 1,305 segundos. Si se han publicado 200,000 en 1,305 segundos, tendríamos que en un solo segundo se ha publicado una media de unas 153 facturas.

En segundo lugar, tenemos el tiempo entre la primera publicación de la primera factura sin procesar, y la publicación de la última factura procesada, que, para mi sorpresa, ha sido exactamente de 21 minutos y 45 segundos. Tras

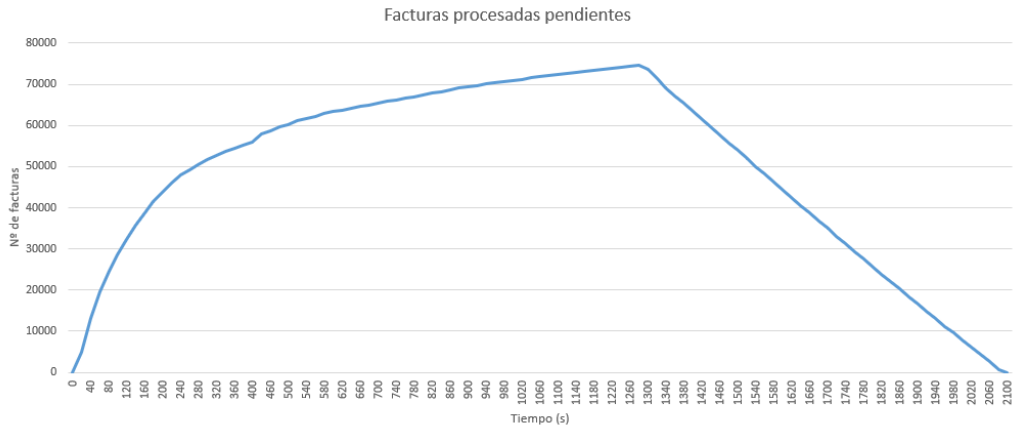

Figura 10.1: Evolución de las facturas procesadas.

ver estos resultados, se ha podido ver claramente que la congestión ha estado en el bus al publicar, ya que el ritmo al que Java llama a este evento es mucho mayor que la capacidad de tratar eventos del *bus*. No obstante, se puede ver que el procesador de facturas es bastante eficiente.

Por otro lado, en el fichero CSV, los datos nos indican que la ejecución del programa ha sido en un total de 34 minutos y 48 segundos, lo cual indica que también se ha producido una congestión en el bus de salida. Con los datos obtenidos en este CSV, he realizado dos gráficas que muestran el flujo de la aplicación. En primer lugar, como podemos observar en la Figura 10.1, podemos ver que el número de facturas pendientes sigue una tendencia logarítmica hasta que claramente, comienza a descender una vez se han introducido todas facturas. Con esta gráfica, podemos ver claramente que ese exceso de tiempo de la introducción de datos en el CSV con respecto al de introducción en la plataforma de datos, simplemente está dado por una latencia a la hora de recibir los eventos cuando está suscrito al *Datahub*, lo cual es una buena noticia, ya que lo importante es que no se produzca una gran congestión a la hora de publicar datos en la plataforma de datos. Un detalle que debemos tener en cuenta es este aspecto logarítmico de la gráfica, y esto es debido que al comienzo, el número de facturas creció de una manera muy rápida ya que el bus no estaba saturado y las aceptaba todas, pero llegó un punto en el que se empezaron a aceptar poco a poco a medida que se iba descongestionando.

En resumen, tenemos que si publicamos unas 150 facturas por segundo, si quisiéramos publicar 200,000, tendríamos que requerir de 22 minutos de ejecución, y además, para recibir los datos en los *Datamarts*, habría que

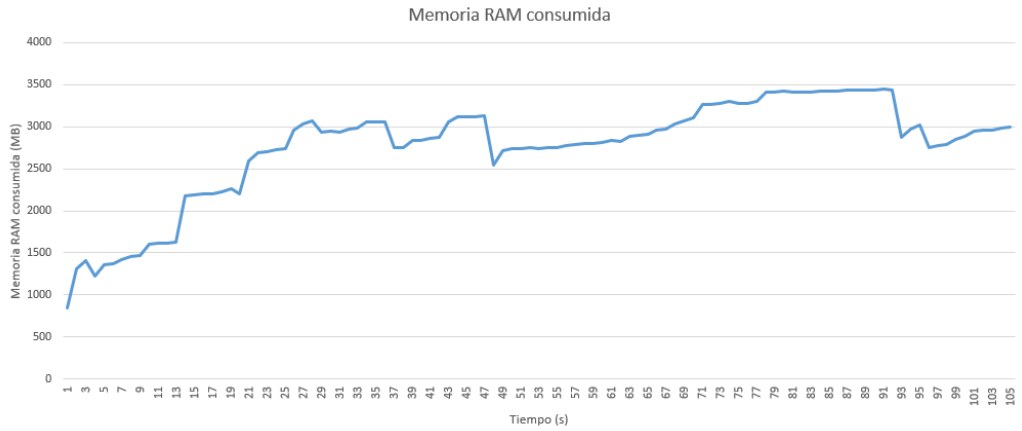

Figura 10.2: Memoria RAM consumida durante la ejecución.

esperar 13 minutos más, que no es problema ya que la visualización no es el aspecto más importante.

Por otro lado, como se ha mencionado con anterioridad, se ha medido la cantidad de memoria *Random Access Memory* (RAM) en uso en cada instante de la ejecución, como bien se puede apreciar en la Figura 10.2, salvo al comienzo de la ejecución que ha aumentado el uso de la RAM, se ha mantenido sobre los 3GB.

En cuanto a almacenamiento secundario, los eventos relativos a las 200,000 facturas ocupan un total de 260 MB, que correspondería a 1.33 KB por factura.

Firma y SignVerifier

Salvo el módulo encargado de generar textos en base 64 de SignVerifier, el cual ha sido testeado haciendo uso de TDD, tanto la aplicación de Firma como el módulo verificador de SignVerifier han sido testeados complementándose entre ellos.

Para empezar, he creado unos certificados generados con un campo extra, que simulará el RFC que tienen las empresas. Todos estos certificados generados estarán firmados por una entidad autentificadora ficticia: “finbook”. El siguiente paso es que SignVerifier de por válido todo certificado firmado por esta entidad autentificadora, y para ello, ha sido modificado el fichero

"cacerts", situado en los archivos de Java. En el caso que FinBook estuviera desplegado en un servidor, el archivo deberá estar modificado en el servidor.

De cara a probar los módulos, con Firma se ha generado firmas generadas con: un certificado firmado por la FNMT, un certificado creado por mí no firmado por "finbook", y otro creado por mí firmado por "finbook". Además, se ha generado una cuarta firma que ha sido adulterada a mano. Los resultados obtenidos son los siguientes:

- En el caso de la firma adulterada, el resultado ha sido que no se ha podido obtener el RFC, ya que el contenido de la firma no ha podido ser validado.
- En el caso de la firma en la que se ha utilizado un certificado creado a mano no firmado por "finbook", el resultado ha sido que no se ha podido obtener el RFC, debido a que el certificado no ha sido firmado por una entidad de confianza. No obstante, la validez de la firma es correcta.
- En el caso de la firma en la que se ha utilizado un certificado firmado por la FNMT, no se ha podido obtener el RFC, ya que el certificado no dispone de este campo. No obstante, la firma es válida, y la entidad que ha firmado el certificado es de fiar.
- Por último, en el caso de la firma realizada con un certificado firmado por "finbook", sí se ha podido obtener el RFC, ya que la firma es válida, la entidad es de confianza, y el certificado dispone de este campo.

Respecto al correcto envío de la firma a los *web sockets*, ha sido probado con los compañeros que han implementado el sistema de *login*, en el proceso de integración.

10.1.2. A nivel personal

Desde un punto de vista personal, este Trabajo de Fin de Grado me ha aportado una nueva visión y forma de resolver problemas en los que se manejan una gran cantidad de datos, en los que la persistencia de estos es crítica, y en los que el formato de los datos podría cambiar en cualquier momento, que son problemas en los que las tradicionales bases de datos no ofrecen una solución clara y realmente eficiente. Además, no solo me ha

aportado esta nueva visión para solucionar problemas, sino una manera de poder llevar a cabo esta solución.

10.1.3. A nivel profesional

Por otro lado, a nivel profesional, este Trabajo de Fin de Grado me ha ayudado a adquirir las disciplinas y experiencias necesarias para participar en cualquier otro proyecto en la que su arquitectura principal sea *Big Data*, lo cual para mí es muy importante, ya que es una de las arquitecturas que están y estarán en auge durante los próximos años, lo cual puede que en un futuro pueda hacerme un hueco en el desarrollo de múltiples aplicaciones de este tipo.

10.2. Aportaciones

La presencia de FinBook implica dos grandes aportaciones: una en el ámbito de la economía, y otra en el ámbito informático.

En cuanto a las aportaciones en el ámbito de la economía, podría significar un pequeño granito de arena en el inicio de la digitalización de la factura, que puede acarrear la solución de problemas con los que las agencias tributarias lidian año tras año, y es el grave problema de la evasión de impuestos. Además, de llegar esta digitalización, la realización de tareas de tramitación mejorarían muchísimo en cuanto a rendimiento, ya que muchos de los costosos procesos realizados actualmente por seres humanos pasarían a realizarlos estas aplicaciones, que además, la posibilidad de error es mucho menor. Estos trabajadores podrían ahorrarse ese tiempo que ocupan realizando estas tareas, y apoyar en otros ámbitos de las empresas para las que trabajan. Por consiguiente, la realización de este Trabajo de Fin de Grado podría ayudar a fomentar a las agencias tributarias a usar las facturas digitales.

En cuanto a las aportaciones en el ámbito de la informática, es un claro ejemplo de que el poder realizar aplicaciones con una arquitectura basada en *Big Data* no es excesivamente complejo, y podría formar parte del inicio de la verdadera revolución que el *Big Data* lleva prometiendo años, que todavía no ha llegado por la complejidad de muchas otras herramientas existentes, o desconocimiento total o parcial de esta visión de la resolución de ciertos problemas.

En resumen, FinBook ha sido un experimento realizado para entender qué productos tecnológicos existirán cuando se produzca la digitalización de la factura.

10.3. Trabajos futuros

La realización de FinBook invita a seguir ampliando la propia aplicación, ya que la escalabilidad que proporciona su propia arquitectura permite que se pueda hacer de una manera muy sencilla, y permite aprovechar al máximo las capacidades de la misma.

Por un lado, tenemos la escalabilidad externa, y es que el número de aplicaciones que hagan uso del *Datahub* de FinBook podría aumentar considerablemente. La idea es que en cualquier momento, se pueda desarrollar una aplicación que publique o consuma eventos de los diferentes tanques disponibles una vez se les haya dado la autorización necesaria para acceder a estos tanques. Esta escalabilidad no depende tanto de el *Datahub* desarrollado, pero sería muy interesante ver la evolución que tendría FinBook con un gran número de usuarios.

No obstante, una escalabilidad que sí que depende completamente del *Datahub*, es su propia escalabilidad interna. Esta escalabilidad consistiría en poder disponer en un futuro de más tanques en los que se puedan publicar otro tipo de eventos muy interesantes como es el registro de las actividades que realiza una empresa, registros para llevar la contabilidad de cada empresa, eventos para llevar a cabo otros balances económicos secundarios, eventos para controlar el inventario de cada empresa, entre otros muchos ejemplos.

Otro aspecto en el que puede mejorar FinBook es en su rendimiento. Si bien los resultados han sido mejores a los esperados, aún son muy mejorables, y puede llegar a ser interesante ver hasta donde se puede llegar en cuanto a rendimiento se refiere.

Referencias

Álvarez, C. (2018). España, quinto país de europa en financiación alternativa. *BBVA*. Descargado 2020-05-30, de https://www.bbva.com/es/espana-quinto-pais-europa-financiacion-alternativa/

Barreix, A. D., Zambrano, R., Costa, M. P., da Silva Bahia, Á. A., de Jesus, E. A., de Freitas, V. P., ... others (2018). *Factura electronica en america latina* (Vol. 595). Inter-American Development Bank.

Bartholomew, D. (2005, 02). Paper or ether? *Industry Week*, *254*, 26-33.

Berez, S., y Sheth, A. (2007). Break the paper jam in b2b payments. *Harvard Business Review*, *85*(11), 28.

Cabrera, J. (2019a). "la llegada de la factura electrónica a las empresas impulsará su digitalización y reducirá el fraude". *Hablemos de Empresas*. Descargado 2020-05-29, de https://hablemosdeempresas.com/empresa/factura-electronica-digitalizacion/

Cabrera, J. (2019b). "vamos hacia un control de la facturación por parte de las administraciones tributarias". *Channel Partner*. Descargado 2020-05-29, de https://www.channelpartner.es/pymes/entrevistas/1113693047302/vamos-hacia-control-de-facturacion-parte-de-administraciones-tributarias.1.html

Cayambe Chicaiza, E. E. (2015). *Diseño e implementación de un sistema de facturación electrónica para la universidad central del ecuador.* (B.S. thesis). Quito: UCE.

Cecarm. (2015). Tipos y formatos de factura electrónica. *Cecarm*. Descargado 2020-05-29, de https://www.cecarm.com/servlet/s.Sl?METHOD=DETALLENOTICIA&sit=c,732,m,3132&id=36107

Chen, S.-C., Wu, C.-C., y Miau, S. (2015). Constructing an integrated e-invoice system: the taiwan experience. *Transforming Government: People, Process and Policy*.

Christophe, V. (2017). Outgoing invoices: 9 reasons to go paperless. *Generix Group*. Descargado 2020-05-29, de https://www.generixgroup.com/en/blog/outgoing-invoices-9-reasons-go-paperless

Circulantis. (s.f.). Financiación alternativa. *Circulantis*. Descargado 2020-05-30, de https://circulantis.com/blog/financiacion-alternativa/

de Alcalá, U. (2018). Las cinco v que sirven para explicar el big data. *Universidad de Alcalá*. Descargado 2020-06-05, de https://www.master-bigdata.com/big-data-5-v/

de Estadística, I. N. (2019). Estructura y dinamismo del tejido empresarial en españa directorio central de empresas (dirce) a 1 de enero de 2019. *Notas de prensa - INE*. Descargado 2020-06-27, de https://www.ine.es/prensa/dirce_2019.pdf

El Hani, E. (2001). The e-invoice-a legal document? *Credit Management*, 22–23.

Fernández, E. (2014). ¿cómo se realiza la liquidación del iva? *Anfix*. Descargado 2020-05-30, de https://anfix.com/blog/la-liquidacion-de-iva/

Foryszewski, S. (2006). The evolution of b2b electronic invoicing. *Credit Control*, *27*(4/5), 35.

Guru99. (2014). What is big data? introduction, types, characteristics and example. *Guru99*. Descargado 2020-06-05, de https://www.guru99.com/what-is-big-data.html

Haq, S. (2007). Electronic invoicing gains as adoption barriers fall. *Financial Executive*, *23*(7), 61–63.

Ivanovic, L. (2019). Data lake vs. data warehouse vs. data hub – what's the difference? *Miktysh Blog.*. Descargado 2020-06-05, de https://miktysh.com.au/data-lake-vs-data-warehouse-vs-data-hub-whats-the-difference/

Jiménez, C. (2019). Big data: qué es y cómo funciona. *Forbes*. Descargado 2020-06-05, de https://forbes.es/empresas/33690/big-data-funciona/

Microsoft. (2018). Patrón de publicador y suscriptor. *Microsoft*. Descargado 2020-06-05, de https://docs.microsoft.com/es-es/azure/architecture/patterns/publisher-subscriber

Mogollón, L. C., y Bedoya López, E. (2019). Incidencia de la facturación electrónica en la reducción de la evasión fiscal.

Molina, C. (2008). La factura electrónica ahorrará 15.000 millones. *Cinco días*, 07–02. Descargado 2020-05-29, de https://cincodias.elpais.com/cincodias/2008/02/07/economia/1202367384_850215.html

Ortega, B. H., y Cinca, C. S. (2009). ¿ qué induce a las empresas a adoptar facturación electrónica? efecto de las percepciones y del entorno competitivo. *UCJC Business and Society Review (formerly known as Universia Business Review)*(24).

Palacios de la Flor, J. J. (s.f.). Plataforma para la gestión de facturas electrónicas factura-web.

Real academia española: Diccionario de la lengua española (23.ª ed. ed.). (2008). RAE.

Rombel, A. (2007). Paving the way for e-invoices. *Global Finance*, *21*(3), 21-22.

Sanz, J. R. (2008). *Gestión del cobro de las operaciones de venta internacional*. Editorial Club Universitario.

Søgaard, J. S. (2018, 09). How business blockchain can enable automatic and near real-time vat settlement.

Sisense. (2017). What is a data mart? *Sisense*. Descargado 2020-06-05, de `https://www.sisense.com/glossary/data-mart/`

Zamora, J. (2016). Aplicaciones fintech, el nuevo mercado financiero online. *El Español*. Descargado 2020-05-30, de `https://elandroidelibre.elespanol.com/2016/06/aplicaciones-fintech.html`

Agradecimientos

Me gustaría agradecer a mi tutor, José Juan, y a mis compañeros de Monentia SL por la ayuda brindada durante la realización del proyecto.

Anexo

A continuación, se adjuntarán los respectivos links de FinBook, Firma y SignVerifier. Cabe destacar que en FinBook, los módulos realizados por mí son “datahub” e “invoice-handler”.

- FinBook: https://github.com/finbook-io/finbook
- Firma: https://github.com/finbook-io/Firma
- SignVerifier: https://github.com/finbook-io/sign-verifier