PROYECTO DE GRADO

Presentado ante la ilustre UNIVERSIDAD DE LOS ANDES como requisito final para obtener el Título de INGENIERO DE SISTEMAS

Desarrollo del subsistema de creación, exportación y envió de facturas electrónicas en la aplicación Datalog DWEB.

Por

Br. Alex Manuel Romero Barroeta

Tutor: Dr. Rafael Rivas

Septiembre 2019

Desarrollo del subsistema de creación y exportación de facturas electrónicas en la aplicación Datalog DWEB.

Br. Alex Manuel Romero Barroeta

Proyecto de Grado — Sistemas de Computación, 12 páginas

Resumen: La factura electrónica es un documento digital de índole fiscal, que tiene su origen en las legislaciones Latinoamericanas que surgieron entre los años 2000 a 2005. Actualmente la factura electrónica es empleada de forma obligatoria u optativa en distintos países alrededor del mundo. La factura electrónica cuenta con al menos dos elementos básicos que son el Mensaje de datos basado en el estándar universal, abierto, no propietario (XML) y el uso de firmas electrónicas basadas en infraestructura de clave pública, la cual es una combinación de hardware y software con políticas y procedimientos de seguridad, que permiten la ejecución con garantías de operaciones criptográficas y el no repudio de transacciones electrónicas. En los sistemas fiscales digitales más maduros, un tercer elemento caracteriza también a la factura electrónica, la certificación, la cual consiste en la validación de la sintaxis y el certificado digital del emisor que realiza la administración tributaria o un Tercero en Confianza, para garantizar su coherencia con el estándar definido por la autoridad fiscal correspondiente y la validez de la firma electrónica del emisor. Cuando ambas validaciones son exitosas, se adiciona al documento un sello digital que certifica la validez de dicha factura y otorga efectos fiscales a la misma.

En este trabajo se presenta el desarrollo de un subsistema de facturación integrado en la aplicación web Datalog DWEB, sistema encargado de realizar diversas funciones de índole fiscal para la realización de facturas electrónicas basadas en la normativa fiscal por la cual se rige Italia.

El desarrollo se basa en la aplicación de diversas tecnologías con base en ASP .NET MVC 4 mezclado con JQuery, lo que permite la integración del subsistema de la facturación electrónica con otros subsistemas que se desarrollaron en paralelo (haciendo uso de la técnica ágil Scrum) que pertenecen a Datalog DWEB, y un sistema ya existente llamado Datalog DHUB, mediante el uso de una API Web que permite validar el uso de la infraestructura de clave pública y la certificación respectivamente, es decir todos los procedimientos necesarios para la realización de la factura electrónica desde su creación hasta su emisión, pasando por diversos estados y ciclos de validación internos y externos, permitiendo obtener como producto una factura electrónica válida para su posterior exportación y registro fiscal mediante envíos realizados tanto a clientes como a las entidades fiscales correspondientes según el régimen fiscal de la empresa que usa el sistema.

Palabras Clave: Facturación Electrónica, Aplicación Web, ASP .NET MVC 4, Scrum, JQuery, Api Web.

Índice

Índice de figuras

Índice de tablas

CAPITULO 1

Introducción

Los últimos avances en tecnología han dado lugar a nuevas ideas, nuevas aplicaciones y nuevos desarrollos a nivel informático, los cuales sin duda se encuentra destinados principalmente a la necesidad de generar mediante soluciones tecnológicas la comodidad y el confort para los clientes de una compañía o comercio, teniendo en cuenta los aspectos históricos, políticos, culturales, científicos y técnicos pero sobre todo destacando las actividades comerciales, que apuntan en dirección a un proceso de una globalización inminente, entre una de las principales y destacadas necesidades se encuentra el uso de la Facturación Electrónica.

La factura electrónica es un documento digital de índole fiscal, que tiene su origen en las legislaciones Latinoamericanas que surgieron entre los años 2000 a 2005. Actualmente la factura electrónica es empleada de forma obligatoria u optativa en distintos países alrededor del mundo. La factura electrónica cuenta con al menos dos elementos básicos que son el Mensaje de datos basado en el estándar universal, abierto, no propietario (XML) y el uso de firmas electrónicas basadas en infraestructura de clave pública, la cual es una combinación de hardware y software con políticas y procedimientos de seguridad, que permiten la ejecución con garantías de operaciones criptográficas y el no repudio de transacciones electrónicas. En los sistemas fiscales digitales más maduros, un tercer elemento caracteriza también a la factura electrónica, la certificación, la cual consiste en la validación de la sintaxis y el certificado digital del emisor que realiza la administración tributaria o un Tercero en Confianza, para garantizar su coherencia con el estándar definido por la autoridad fiscal correspondiente y la validez de la firma electrónica del emisor. Cuando ambas validaciones son exitosas, se adiciona al documento un sello digital que certifica la validez de dicha factura y otorga efectos fiscales a la misma.

Para cada país las empresas basan sus actividades comerciales y fiscales basándose en la normativa que dicta el organismo o ente encargado de hacer las regulaciones pertinentes de las actividades comerciales en dicho país, al encontrarnos en un proceso de globalización comercial, nace la necesidad de ofrecer a los comerciantes o empresarios una solución que les proporcione la

posibilidad de regular los procesos fiscales en sus empresas de una manera rápida, cómoda, segura e innovadora, necesidad que encaja perfectamente con el concepto de factura electrónica.

La empresa Datalog Italia Srl, encargada de ofrecer a sus clientes desde hace 20 años soluciones tecnológicas para la realización de gestiones de índole fiscal en Italia, propone la construcción del sistema integrado DWEB como un portal de Facturación Electrónica en línea, pensando en sus clientes como principales actores y haciendo uso de otros sistemas y subsistemas propios para la certificación de las facturas. Este sistema parte como una versión web de su sistema principal existente llamado KING agregado a todos los requerimientos necesarios para realizar la facturación electrónica.

La raíz de la problemática es que actualmente el sistema KING perteneciente a la empresa Datalog Italia Srl, se encarga de realizar todos estos procesos pero es un sistema de escritorio, motivo por el cual el sistema no puede ser usado para la generación de la factura electrónica, esto abre paso a la creación de DWEB como aplicación web destinada a la generación y construcción de la factura electrónica integrada a KING y otros subsistemas que hacen la conexión necesaria con los entes reguladores que hacen la certificación del documento.

1.1 Antecedentes

El objetivo de esta sección es profundizar brevemente en las investigaciones más relevantes realizadas con anterioridad, así como de sus bases teóricas, para sustentar el desarrollo de este proyecto, en este sentido se analizarán los diferentes estudios previos, encontrando los aportes que dan base al desarrollo de esta investigación. De los fundamentos teóricos revisados se definió como variables de estudio las bases de la Facturación Electrónica y en las implementaciones precedentes basadas en la Facturación Electrónica.

En el área la facturación electrónica, se tienen pocas referencias en este trabajo en Venezuela, puesto que no es obligatoria por ley la realización de la factura electrónica como en otros países, de hecho, la norma vigente en materia de regulación de facturas y otros documentos fiscales es la Providencia Administrativa Nro. 0071 dictada por el SENIAT, publicada en la Gaceta Oficial Nro. 39.795 del 08 de noviembre de 2011. En esta se establecen tres medios de emisión de facturas: Formatos elaborados por imprentas autorizadas por el SENIAT, Formas libres elaboradas por imprentas autorizadas por el SENIAT y Máquinas Fiscales, motivo por el cual el uso de la facturación electrónica es escaso y voluntario, puesto que podría adaptarse entre las Formas libres, pero no están definidas aun en la actualidad.

Son muy pocos los que han enfocado sus estudios en este tema, entre las pocas referencias encontradas se puede destacar un artículo de revista publicado por *Villegas José*, (2014) donde describe

las bases de los Comprobantes fiscales digitales y facturación electrónica, pero realizando sus estudios enfocados en el país de México basándose en su normativa fiscal.

Profundizando y ampliando la investigación en locaciones exteriores a nivel de Latinoamérica y el mundo se encuentran muchos trabajos realizados a partir de las legislaciones Latinoamericanas que surgieron entre los años 2000 a 2005. Actualmente la factura electrónica es empleada de forma mandataria u optativa en distintos países alrededor del mundo, funcionando en países como Chile, Colombia, Ecuador, Guatemala, Perú, República Dominicana, México, Costa Rica, Panamá, Uruguay Argentina, España, Italia y la Unión Europea, funcionando cada uno bajo la normativa tributaria de cada país, pero manteniendo en común los elementos básicos mencionados de la definición de la factura electrónica que garantizan el documento digital de índole fiscal como válido. Entre los países que disponen de normativa de factura electrónica están Argentina, Chile, México, Costa Rica, Colombia y Australia, además de todos los de la Unión Europea, en función de la adopción de la Directiva 2001/115.

En el área de las implementaciones precedentes basadas en la Facturación Electrónica, por lo mencionado anteriormente a nivel de Latinoamérica y el mundo pueden encontrarse diversos trabajos realizados en este ámbito adecuado a las normas de cada país, teniendo en cuenta de que el desarrollo de este proyecto tiene como propósito la integración del mismo a un sistema de facturación para la empresa italiana Datalog Italia Srl, nos enfocaremos en revisar los trabajos previos realizados en Italia, sin restar importancia a que todos los sistemas de facturación electrónica se realizan enfocados con la misma estructura y pueden ser adaptados a cualquier país, motivo por el cual este trabajo es de gran importancia en un futuro para la realización de la facturación electrónica en Venezuela.

Según la Ley de Balance del 2018 (*Legge di Bilancio 2018*) La factura electrónica B2B entre empresas privadas comienza su expansión en Europa. Con esta ley se hace la aprobación del texto que introduce la obligación de facturación electrónica entre empresas privadas. El texto, que modifica el Decreto Legislativo 127 del 5 de agosto 2015, incluye las transacciones entre privados dentro de los procesos de facturación electrónica. Por lo tanto, las empresas están obligadas a transmitir sus facturas a través del SDI (*Sistema di Interscambio*) y en el formato XML definido por *L'Agenzia delle Entrate*. Italia es uno de los países más avanzados a nivel europeo en la adopción de la factura electrónica y también, en la masificación del uso de la red PEPPOL para el intercambio de datos entre el sector Público y sus proveedores.

Esto ha llevado a la creación de diversos softwares privativos que facilitan a las empresas que desempeñan cualquier tipo de actividad económica fiscal en Italia la realización de la factura electrónica y la conexión con los diversos entes de control y gestión como lo es ya mencionada *L'Agenzia delle Entrate*. Estos softwares, aunque son antecedentes claros de este proyecto solo pueden

ser tomados como referencia básica en cuanto a los procesos ya que al ser privativos no contamos con la documentación o información a nivel de implementación que podamos mencionar.

En cambio, dentro de la empresa Datalog Italia Srl, contamos con un proyecto base perteneciente a la empresa, el cual es la aplicación de escritorio llamada KING, un sistema de gestión contable adaptable a los requerimientos de cada empresa. Este sistema se encarga de realizar los cálculos y registro necesarios para la creación de la factura, este sistema cuenta además con todos los elementos con los que se pueden validar el documento de índole fiscal a nivel de operaciones matemáticas, redondeo, cálculos de diversos impuestos, totales, aproximación y tasas que pueden ser requeridas al momento de realizar la factura electrónica, permitiéndonos tener una base sólida de la cual partir para la realización de este proyecto. Es importante mencionar que el sistema KING comparte la base de datos y rutinas con el sistema principal DWEB que el sistema padre de este subsistema de Facturación Electrónica.

1.2 Definición del problema

En Italia, con la aprobación de la Ley del Balance del 2018 (*Legge di Bilancio 2018*) que modifica el Decreto Legislativo 127 del 5 de agosto 2015, incluyendo las transacciones entre privados dentro de los procesos de facturación electrónica, dicta que las empresas están obligadas a transmitir sus facturas a través del SDI (*Sistema di Interscambio*) y en el formato XML definido por *L'Agenzia Delle Entrate*, motivo por el cual se implementa por obligatoriedad el uso de la Facturación Electrónica en Italia.

En la resolución de esta ley se indicó que, a partir del 1 de julio de 2018, comenzaba la obligación para aquellas empresas con actividades relativas a suministros de hidrocarburos para motores o que presten servicios de subcontratistas dentro de concursos públicos y a partir del 1 de enero de 2019, la obligatoriedad se masificó para todas las empresas privadas.

Está medida genera la necesidad de la implementación de soluciones digitales que permitan a las empresas realizar el proceso de facturación electrónica en Italia.

Datalog Italia Srl, hasta la fecha, contaba con el sistema de gestión empresarial y contable llamado KING, el cual es una aplicación de escritorio personalizable a los requerimientos de sus clientes, pero que, al momento no cuenta con todos los implementos necesarios para la creación de la factura electrónica, generando un gran problema para todos sus clientes.

En base a este problema, nace la idea de la creación de una aplicación web, la cual cuenta con las ventajas de no ser un sistema de escritorio dependiente de un solo equipo, que usará rutinas de

KING como sistema base, en complemento con la integración de nuevos subsistemas, que trabajando en conjunto permitirán la creación de la factura electrónica.

Con este desarrollo e integración en el entorno web se solucionará la necesidad que tiene Datalog como empresa de expandir las tecnologías sobre las cuales funcionan sus aplicaciones y el problema actual de sus clientes por no poder realizar la facturación electrónica.

En este proyecto el objetivo es el desarrollo de uno de los subsistemas integrados del sistema DWEB para el cumplimiento de este objetivo, al integrarlo con el subsistema de registro de datos, planificación y configuración de DWEB y los sistemas KING, DHUB.

Los problemas que se pueden apreciar en sistema actual se pueden resumir de la siguiente manera. (Ver figura 1).

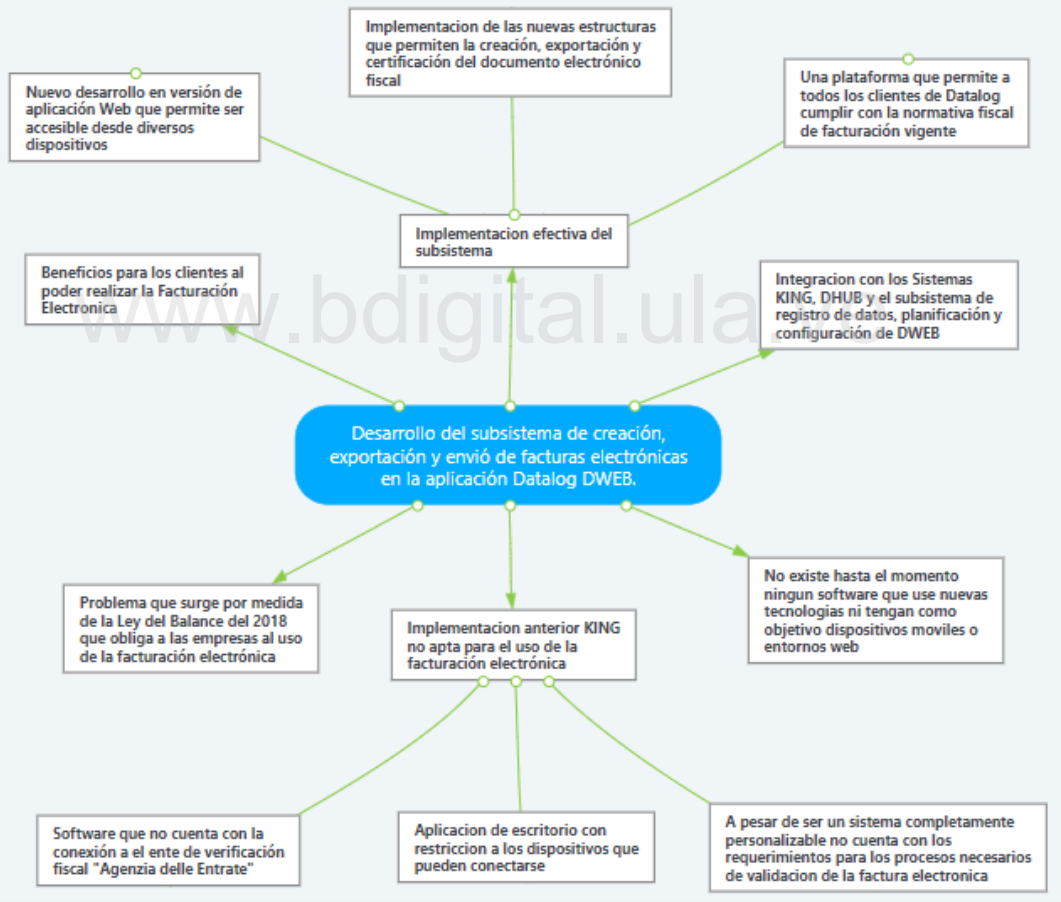


Figura 1: Árbol de Problemas

1.3 Justificación

La empresa Datalog Italia Srl cuenta con un programa que actualmente es capaz de realizar muchas de las validaciones necesarias para hacer una factura correcta fiscalmente, este sistema es una aplicación de escritorio llamado KING, que además es un sistema personalizable a cada uno de los clientes de la empresa.

Actualmente el sistema a pesar de funcionar correctamente, no cumple con los requerimientos necesarios para la realización de la facturación electrónica, la cual desde la aprobación de la Ley del Balance del 2018 paso a ser de carácter obligatorio por ley en el país de Italia.

Por ello, se propone el desarrollo del sistema integrado DWEB que será un conjunto de subsistemas web que permitirán el reutilizamiento de parte de las rutinas de validación ya existentes en el sistema KING, agregando también las rutinas necesarias para el proceso de certificación que requiere la factura electrónica y para la exportación de la misma en los formatos establecidos, permitiendo generar el documento digital fiscal válido.

Sin el desarrollo adecuado del subsistema propuesto, no será posible la integración con los sistemas KING, DHUB y el subsistema de registro de datos, planificación y configuración de DWEB.

Esto generaría un problema para todos los clientes de la empresa Datalog Italia Srl y los dejaría prácticamente fuera del mercado de la distribución de software administrativo y contable por la obligatoriedad existente en el país de la facturación electrónica.

El sistema propuesto al ser web permitirá además maximizar el alcance de los usuarios de la aplicación en dispositivos diversos a los de escritorios, generando así además una solución que se adapta a la era de la globalización y la popularidad de la integración de los dispositivos móviles (en particular las tablets) dentro de los ambientes laborales de cada empresa.

Entre otras razones se pueden mencionar el impacto positivo y los beneficios que proporciona la facturación electrónica en general a nivel mundial entre los cuales destacan los siguientes:

- Dependiendo del tamaño de las empresas y el volumen de su facturación, el ahorro en concepto de emisión y gestión de facturas puede fluctuar entre el 40% y el 80%, lo que causa una reducción de los costes.
- Oportunidad en la información, tanto en la recepción como en el envío.
- Ahorro en el gasto de papelería, la factura electrónica es ecológica, eliminación de espacios para almacenar documentos históricos.
- Facilidad en los procesos de auditoría y para el cálculo de impuestos, agilidad en la localización de información, reducción en tiempos de gestión.
- Mayor seguridad en el resguardo de los documentos, aumenta la seguridad documental, menor probabilidad de falsificación, contabilidad electrónica automatizada.

- Procesos administrativos más rápidos y eficientes, mayor agilidad en la toma de decisiones (mejora de la eficiencia en el proceso).
- Reduce errores en el proceso de generación, captura, entrega y almacenamiento.
- Acceso a financiamiento a través de factoraje de cuentas por cobrar.

1.4 Objetivos

1.4.1 Objetivo General

Desarrollar un subsistema de facturación electrónica que permita generar documentos fiscales válidos, mediante la integración con los otros sistemas existentes (KING y DHUB) y el subsistema de registro de datos, planificación y configuración de DWEB, para completar todos los procesos de certificación que permiten obtener como producto un documento digital de índole fiscal válido (factura electrónica).

1.4.2 Objetivos Específicos

1. Estudio del sistema KING para identificar las funciones y rutinas que pueden reutilizarse en el sistema integrado DWEB.
2. Analizar los requerimientos legales que proporciona *L'Agenzia Delle Entrate* para la correcta construcción de la facturación electrónica.
3. Estudiar los elementos básicos de la facturación electrónica: El mensaje de datos basado en el estándar universal, abierto, no propietario (XML), el uso de firmas electrónicas basadas en infraestructura de clave pública y el proceso de certificación.
4. Desarrollar la estructura del subsistema a implementar.
5. Desarrollar la interfaz de usuario que permita el uso de la funcionalidad del subsistema.
6. Desarrollo de las funciones que regulan el envío de la factura electrónica en sus diversos formatos.
7. Desarrollo de las funciones en el sistema de comunicación API web que permite la comunicación con los sistemas KING y DHUB.
8. Integrar el subsistema de facturación con el subsistema de registro de datos, planificación y configuración de DWEB.
9. Ejecutar los casos de prueba del sistema para validar la creación correcta de los documentos digitales de índole fiscal válidos y el envío de las facturas al ente regulador.
10. Realizar el correspondiente manual de usuario del sistema.

1.5 Metodología

El desarrollo del subsistema planteado, será elaborado utilizando como guía el método SCRUM identificado y definido por *Nonaka* & *Takeuchi*, (1986) y presentado por el cofundador y cocreador *Schwaber Ken*, (1995), método basado en las tecnologías ágiles, el uso de este método se describe como un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto, agregándole visibilidad al proyecto; pues, se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto.

Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o se puede redefinir en el proceso en base a las necesidades del proyecto, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales, también permite identificar y solucionar ineficiencias sistemáticamente o cuando se quiere trabajar utilizando un proceso especializado en el desarrollo de producto. Mediante las iteraciones de las fases del producto. Este método consta de los siguientes componentes:

1. **Modelo del producto:** Describe el tipo de producto estableciendo las características del mismo planificando una lista de tareas por iteración, donde cada iteración tiene consigo una meta u objetivo específico.

2. **Modelo del proceso:** En Scrum un proyecto se ejecuta en ciclos temporales cortos y de duración fija con iteración. Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.

3. **Modelo del grupo de desarrollo:** Este modelo describe como el grupo de desarrollo debe estar organizado y cuáles son los roles de cada uno de sus miembros.

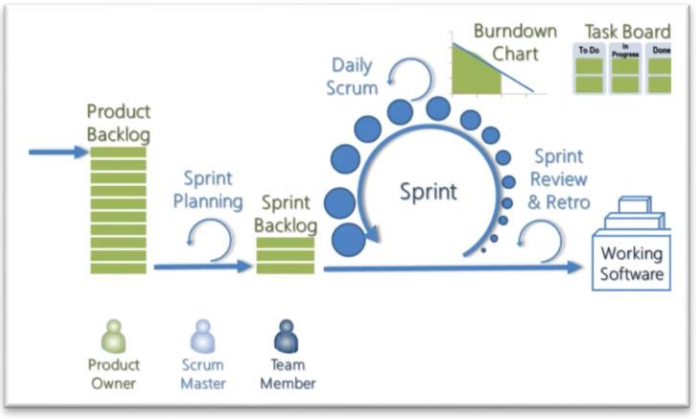


Figura 2: Diagrama de la metodología SCRUM, tomado de Internet, https://joansarda.wordpress.com/tag/scrum

Para el modelado del subsistema de facturación web, se utilizó el entorno de desarrollo integrado (IDE, por sus siglas en inglés) Microsoft Visual Studio para Windows, creado por *Microsoft Corporation, (1997)* y que actualmente se encuentra en su versión *Microsoft Visual Studio 16.1.6, (2019)*. El cual es compatible con múltiples lenguajes de programación, facilitando el desarrollo de la aplicación DWEB, la cual se desarrolló bajo los lenguajes de C#, JQuery, HTML, al igual que entornos de desarrollo web .NET y MVC 4.

1.6 Acuerdo de Confidencialidad

El producto final del desarrollo de este proyecto pertenece a la empresa Datalog Italia Srl y está registrado bajo un acuerdo de confidencialidad, motivo por el cual podrá ser visualizado en la presentación final del proyecto, también será documentado a continuación a nivel de desarrollo, tecnologías utilizadas y funcionalidad teórica, pero el código es 100% propiedad de la empresa Datalog Italia Srl y no podrá ser compartido.

1.7 Alcance

Elaborar un subsistema de facturación electrónica que sea capaz de integrarse con los sistemas ya existentes (KING y DHUB) y con el subsistema que será desarrollado en paralelo (subsistema de registro de datos, planificación y configuración de DWEB), que solucione la problemática existente en la empresa Datalog Italia Srl y permita como producto la creación de la factura electrónica valida por todos los procesos de certificación que tienen los entes reguladores en Italia.

1.8 Planificación

Para la elaboración del presente trabajo se elaboró un cronograma de actividades que deben ser realizadas a lo largo de 16 semanas.

1.8.1 Cronograma de Actividades

- **Actividad 1**: Reunión semanal con el tutor y los asesores para evaluar el sprint.
- **Actividad 2**: Planificación del siguiente sprint.
- **Actividad 3**: Revisión y recolección de fuentes bibliográficas.
- **Actividad 4**: Estudio del tema de la facturación electrónica.
- **Actividad 5**: Desarrollo del subsistema de facturación electrónica.
- **Actividad 6**: Desarrollo de la funcionalidad de interacción con KING, DHUB.
- **Actividad 7**: Integración del subsistema en el sistema DWEB.
- **Actividad 8**: Pruebas y despliegue del subsistema.
- **Actividad 9**: Redacción del informe de Proyecto de Grado.
- **Actividad 10**: Entrega de informe final para realizar correcciones.
- **Actividad 11**: Aplicación de las correcciones.
- **Actividad 12**: Presentación del Proyecto de Grado.

Cronograma de Actividades																
Actividad / Semana	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
2	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
3	X	X	X	X												
4		X	X	X	X	X	X	X								
5			X	X	X	X	X	X	X	X	X	X				
6						X		X		X	X	X				
7					X				X			X	X			
8					X			X			X		X	X		
9												X	X	X		
10													X	X		
11															X	
12																X

Tabla 1 Cronograma de actividades

1.8.2 Cronograma de Evaluación

- **Evaluación 1:** Inscripción del proyecto de grado.
- **Evaluación 2**: Presentación de avances del proyecto a los jurados.
- **Evaluación 3**: Entrega de la primera versión del proyecto a los jurados.
- **Evaluación 4**: Entrega de la versión final del proyecto de grado.
- **Evaluación 5**: Defensa del proyecto de grado.

Cronograma de Evaluaciones																
Evaluación / Semana	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	X	X														
2								X								
3									X	X						
4												X	X			
5													X	X	X	X

Tabla 2 Cronograma de Evaluaciones

1.9 Estructura del Documento

Capítulo 1, Describe los antecedentes que son base para determinar correctamente el problema, así como también, el planteamiento del problema, la justificación, se describen los objetivos, el alcance del proyecto y la metodología que se llevó a cabo para su desarrollo.

Capítulo 2, Marco teórico. Esta sección contiene los conocimientos teóricos básicos que son necesarios para el entendimiento y comprensión de proyecto, entre los cuales, se encuentras las técnicas aplicadas para el desarrollo del subsistema, así como también se describe detalladamente, la metodología de trabajo SCRUM utilizada en dicho desarrollo, igualmente se explica el Framework de desarrollo de Windows MVC 4 en .NET C#/WinForm, el cual fue utilizado bajo la planificación de integración final con los otros subsistemas del entorno de servicios de Datalog Italia Srl mencionados previamente.

Capítulo 3, La facturación electrónica. Comprende el estudio y desarrollo de los conceptos concernientes a la definición y los procesos de la factura electrónica, se documenta esta sección partiendo desde los requerimientos básicos mínimos a nivel internacional y se complementa detallando el caso particular de su implementación en Italia, se explica su normativa fiscal, los entes involucrados en el proceso y los procesos que implican la realización de la misma.

Capítulo 4, Análisis de los requerimientos. Se profundiza en la ingeniería de requerimientos del sistema partiendo con el desarrollo de la metodología SCRUM, se procede con el análisis de los requisitos, clasificándolos en requisitos funcionales y no funcionales, lo cual ayuda a entender, desde un punto de vista razonable, la mejor solución que se propone para el problema abordado, que a su vez satisface con las necesidades del cliente.

Capítulo 5, Diseño e implementación del sistema. Comprende la fase de diseño e implementación del sistema propuesto, se abordan los puntos relacionados al diseño y desarrollo de la interfaz de usuario y de la estructura funcional del sistema, denotando todos los requerimientos necesarios para su correcto funzionamiento, analizados en capitulos anteriores, ademas tomando en cuenta las consideraciones particulares que fueron tomadas por parte de la empresa Datalog Italia Srl.

Capítulo 6, Conclusiones y Recomendaciones. Contiene las conclusiones generales del trabajo realizado y las recomendaciones para trabajos futuros.

CAPITULO 2

Marco Teórico

En este capítulo, se describen los fundamentos teóricos necesarios para el entendimiento y comprensión del proyecto. Se hace una introducción a los sistemas de facturación, se define detalladamente la metodología ágil SCRUM para el desarrollo de aplicaciones que adoptan una estrategia incremental en lugar de la planificación y ejecución completa del producto, así como también, el Framework de desarrollo de Windows MVC 4 en .NET C#/WinForm, utilizado para el desarrollo del sistema web planteado en el capítulo 1.

Además se abordan conceptos básicos de algunos de los programas utilizados en el desarrollo, la programación orientada a objetos (POO) y los archivos en formato JSON, repositorios de código o controladores de versión de proyecto, gestores de bases de datos, Apis Web, Sesiones Web, Tokens de seguridad y cifrado/descifrado de datos, creando una base teórica con la finalidad de tener una introducción a las herramientas de diseño del proyecto y las técnicas de desarrollo utilizadas, para permitir al lector tener una idea de la naturaleza del contenido del resto del documento.

2.1 Sistemas de Facturación

En la actualidad son tres los sistemas de facturación existentes que son usados por las diferentes empresas y comercios que se dedican a la realización de cualquier actividad comercial, debemos considerar que cada uno de estos sistemas de facturación se adapta a las necesidades y requerimientos de dichas entidades, así como también, a los requerimientos y normativas fiscales del país en el cual ejercen sus actividades comerciales.

En aquellas empresas que cuentan con numerosos recursos, es común que se utilicen aquellos sistemas de facturación correspondientes a los programas informáticos. Esto se debe a que los mismos, disponen de diferentes funciones y utilidades que ayudan a mantener un cierto control acerca de los movimientos financieros correspondientes a la empresa. Además, el sistema de base de datos que utiliza resulta muy útil para plantificar diferentes aspectos para los cuales, se requiere de la necesidad de un sistema de facturación de este tipo, como pueden ser las auditorias tributarias, y los balances de fin de año.

Las pequeñas y medianas empresas (PyME), suelen utilizar el sistema de facturación en papel. La misma hace referencia a una factura que se realiza mediante la computadora y que cumple con las mismas características y obligaciones que una factura común, como así también, cumple con las habilitaciones legales correspondientes.

En cuanto concierne a los sistemas de facturación electrónicos, suelen ser muchos y de muchas clases, y se definen según las regulaciones tributarias de cada país, estos sistemas tienen por ventaja la simplificación de la emisión de facturas y el uso de los medios electrónicos para su recepción. De todos modos, debemos decir que, en este caso particular para hacer validos estos sistemas de facturación, la factura debe cumplir con ciertas condiciones que son necesarias para crear un documento con validez tributaria y legal, de lo contrario, la misma no tendrá la habilitación correspondiente para que la misma sea considerada una factura legal. Estas condiciones son necesarias como medidas de prevención para evitar algún fraude o estafa.

En los últimos años, podemos apreciar que existe una tendencia a nivel mundial, que busca reemplazar la tradicional facturación en papel por su homologación electrónica, esta se lleva a cabo mediante un documento digital que garantiza la autenticidad de su contenido y gracias a los avances de la tecnología cloud, la gestión de las facturas electrónicas, es más profesional y sencilla.

Los programas de facturación son cada vez más necesarios para todo tipo de empresarios o comerciantes, ya que les permiten ahorrar tiempo y dinero mediante su uso, permitiendo, una gran cantidad de funciones, implementando el uso de diversas herramientas que sirven para gestionar otros aspectos tales como contabilidad, gestión de inventarios, gestión de cuentas bancarias, información fiscal o gestión de clientes.

En resumen, un sistema de facturación es una herramienta de vital importancia para autónomos y dueños de empresas, ya que no solo les permite la realización de las facturas, sino que también facilita la realización de las gestiones fiscales y otros aspectos de su negocio de forma muy sencilla.

En vista de que el objetivo principal en el desarrollo de este proyecto consiste en la realización de un sistema de facturación web y la información para los requerimientos de este tipo de sistemas de facturación son más extensos, definiremos la facturación electrónica posteriormente en una sección independiente.

2.2 Metodología Scrum

Adicionalmente a lo visto en el Capítulo 1 donde se presenta formalmente el modelo de la Metodología Scrum, podemos añadir que, según *Proyectos Ágiles*, (2008) es un proceso de gestión que reduce la complejidad en el desarrollo de productos para satisfacer las necesidades de los clientes. La

gerencia y los equipos de Scrum trabajan juntos alrededor de requisitos y tecnologías para entregar productos funcionando de manera incremental usando el empirismo.

Scrum es un marco de trabajo simple que promueve, mediante la aplicación de buenas prácticas, la colaboración en los equipos para lograr desarrollar productos complejos. Es reconocida como una técnica para desarrollo ágil de software caracterizado por:

- Adoptar una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto.
- Basar la calidad del resultado más en el conocimiento tácito de las personas en equipos auto organizados, que en la calidad de los procesos empleados.
- Solapar las diferentes fases del desarrollo, en lugar de realizar una tras otra en un ciclo secuencial o en cascada.

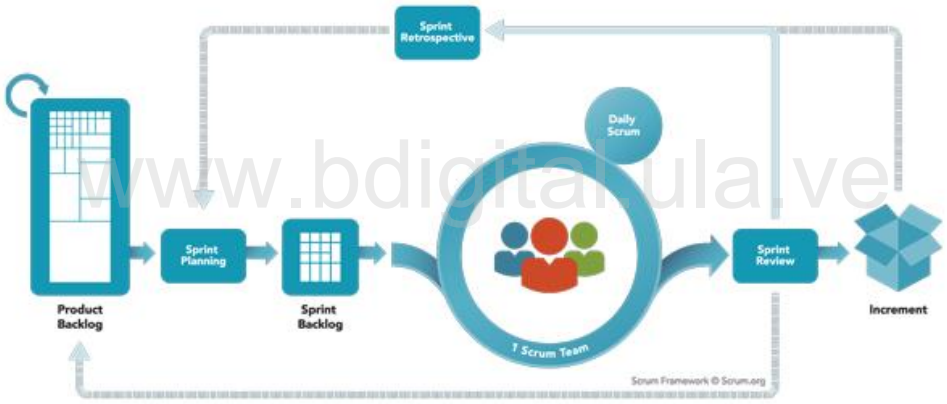


Figura 3: Scrum Model Framework, tomado de Internet,
https://scrumorg-website-prod.s3.amazonaws.com/drupal/inline-images/ScrumFramework.png

2.2.1 Características de Scrum

Ken Schwaber, (2004) Define Scrum como un marco de trabajo que define un conjunto de prácticas y roles, y que puede tomarse como punto de partida para definir el proceso de desarrollo que se ejecutará durante un proyecto. Los roles principales en Scrum son el Scrum Master, que procura facilitar la aplicación de Scrum y gestionar cambios, el Product Owner, que representa a los stakeholders (interesados externos o internos), y el Team (equipo) que ejecuta el desarrollo y demás elementos relacionados con él.

Durante cada sprint, un periodo entre una y cuatro semanas (la magnitud es definida por el equipo y debe ser lo más corta posible), el equipo crea un incremento de software potencialmente entregable (utilizable). El conjunto de características que forma parte de cada sprint viene del Product Backlog, que es un conjunto de requisitos de alto nivel priorizados que definen el trabajo a realizar (PBI, Product Backlog Item). Los elementos del Product Backlog que forman parte del sprint se determinan durante la reunión de Sprint Planning. Durante esta reunión, el Product Owner identifica los elementos del Product Backlog que quiere ver completados y los da a conocer al equipo. Entonces, el equipo converso con el Product Owner buscando la claridad y magnitud adecuadas (Cumpliendo el INVEST) para luego determinar la cantidad de ese trabajo que puede comprometerse a completar durante el siguiente sprint.2 Durante el sprint, nadie puede cambiar el Sprint Backlog, lo que significa que los requisitos están congelados durante el sprint.3

Scrum permite la creación de equipos auto organizados impulsando la localización de todos los miembros del equipo, y la comunicación verbal entre todos los miembros y disciplinas involucrados en el proyecto.

2.2.2 Bases de la Metodología Scrum

La metodología se basa en la colaboración de los equipos de desarrollo formados, trabajando mediante el uso de buenas prácticas que cumplen con los siguientes requerimientos:

- El desarrollo incremental de los requisitos del proyecto en bloques temporales cortos y fijos.
- Se da prioridad a lo que tiene más valor para el cliente.
- El equipo se sincroniza diariamente y se realizan las adaptaciones necesarias.
- Tras cada iteración (un mes o menos entre cada una) se muestra al cliente el resultado real obtenido, para que este tome las decisiones necesarias en relación a lo observado.
- Se le da la autoridad necesaria al equipo para poder cumplir los requisitos.
- Fijar tiempos máximos para lograr objetivos.
- Equipos pequeños (de 3 a 9 personas cada uno).

2.2.3 Roles en Scrum

Los roles en la metodología Scrum se pueden dividir en dos categorías, los roles principales son aquellos mínimos necesarios para completar el producto, desde el punto de vista del desarrollo, según las bases de esta tecnología y los auxiliares son aquellos que no tienen un rol formal y no se involucran frecuentemente en el "proceso Scrum", sin embargo, deben ser tomados en cuenta. Un aspecto importante de una aproximación ágil es la práctica de involucrar en el proceso a los usuarios, expertos del negocio y otros interesados ("stakeholders"). Es importante que esa gente participe y entregue retroalimentación con respecto a la salida del proceso a fin de revisar y planear cada sprint.

En la categoría de los roles principales encontramos:

- **Product Owner**: se asegura de que el equipo Scrum trabaje de forma adecuada desde la perspectiva del negocio. El Product Owner ayuda al usuario a escribir las historias de usuario, las prioriza, y las coloca en el Product Backlog.
- **Scrum Master (o Facilitador)**: el Scrum es facilitado por un ScrumMaster, cuyo trabajo primario es eliminar los obstáculos que impiden que el equipo alcance el objetivo del sprint. El ScrumMaster no es el líder del equipo (porque ellos se auto-organizan), sino que actúa como una protección entre el equipo y cualquier influencia que le distraiga. El ScrumMaster se asegura de que el proceso Scrum se utiliza como es debido. El ScrumMaster es el que hace que las reglas se cumplan.
- **Equipo de desarrollo**: son los desarrolladores que tienen la responsabilidad de entregar el producto. Es recomendable un pequeño equipo de 3 a 9 personas con las habilidades transversales necesarias para realizar el trabajo (análisis, diseño, desarrollo, pruebas, documentación, etc.).

Mientras en la categoría de los roles auxiliares tenemos:

- **Stakeholders (Clientes, Proveedores, Vendedores)**: Son las personas que hacen posible el proyecto y para quienes el proyecto producirá el beneficio acordado que justifica su desarrollo. Sólo participan directamente durante las revisiones del "sprint".
- **Administradores (Managers)**: Son los responsables de establecer el entorno para el desarrollo del proyecto.

2.2.4 Flujo de trabajo en Scrum

El flujo de trabajo se puede definir según los siguientes conceptos en la metodología Scrum según la planificación elegida por el Scrum Master para el desarrollo.

- **Sprint**: El Sprint es el período en el cual se lleva a cabo el trabajo en sí. Es recomendado que la duración de los sprints sea constante y definida por el equipo con base en su propia experiencia. Se puede comenzar con una duración de sprint en particular (2 o 3 semanas) e ir ajustándolo con base en el ritmo del equipo, aunque sin relajarlo demasiado. Al final de cada sprint, el equipo deberá presentar los avances logrados, y el resultado obtenido es un producto que, potencialmente, se puede entregar al cliente.
  Así mismo, se recomienda no agregar objetivos al sprint o sprint backlog a menos que su falta amenace al éxito del proyecto. La constancia permite la concentración y mejora la productividad del equipo de trabajo. El tiempo mínimo de un Sprint es de dos (2) semanas y el máximo es de cuatro (4) semanas.

- **Planificación de sprint**: Al comienzo de un sprint, el equipo de scrum tiene un evento de planificación de sprint, esto se hace generalmente mediante una reunión, donde el principal objetivo es identificar y comunicar cuánto del trabajo es probable que se realice durante el actual Sprint.

- **Revisión de sprint**: Al final de un sprint, el equipo realiza dos eventos: la revisión del sprint y la retrospectiva del sprint. En la reunión de revisión de sprint se presentan los trabajos completados y su duración no debería ser superior a 4 horas para un Sprint de 1 mes.

- **Retrospectiva del sprint**: Después de cada sprint, se lleva a cabo una retrospectiva del sprint, en la cual todos los miembros del equipo dejan sus impresiones sobre el sprint recién superado. El propósito de la retrospectiva es realizar una mejora continua del proceso. Esta reunión tiene un tiempo fijo de cuatro horas.

- **Scrum diario**: También llamado Daily Standup. Cada día durante la iteración, tiene lugar una reunión de estado del proyecto. Su objetivo es que los miembros del equipo se mantengan actualizados unos a otros sobre el trabajo de cada uno desde el último standup, qué problemas han encontrado o prevén encontrar, y qué planean hacer.

    - La reunión tiene una duración fija de entre 5 y 15 minutos.
    - Se recomienda hacerla de pie para recordar que debe ser una reunión breve y centrada en su objetivo, sin divagaciones. Es obligatorio parar todo lo que se está haciendo para concentrarse en la reunión.
    - Si se requiere ampliar un tema, se hará tras el Daily Standup, pero no se interrumpe la dinámica del Standup para elaborar una discusión.
    - Se hace siempre a la misma hora y en el mismo lugar. Si falta alguien, no se pospone la reunión.

2.2.5 Beneficios de Scrum

- **Flexibilidad a cambios**: Gran capacidad de reacción ante los cambiantes requerimientos generados por las necesidades del cliente o la evolución del mercado. El marco de trabajo está diseñado para adecuarse a las nuevas exigencias que implican proyectos complejos.

- **Reducción del Time to Market**: El cliente puede empezar a utilizar las características más importantes del proyecto antes de que esté completamente terminado.

- **Mayor calidad del software**: El trabajo metódico y la necesidad de obtener una versión de trabajo funcional después de cada iteración, ayuda a la obtención de un software de alta calidad.

- **Mayor productividad**: Se logra, entre otras razones, debido a la eliminación de la burocracia y la motivación del equipo proporcionado por el hecho de que pueden estructurarse de manera autónoma.

- **Maximiza el retorno de la inversión (ROI)**: Creación de software solamente con las prestaciones que contribuyen a un mayor valor de negocio gracias a la priorización por retorno de inversión.

- **Predicciones de tiempos**: A través de este marco de trabajo se conoce la velocidad media del equipo por sprint, con lo que es posible estimar de manera fácil cuando se podrá hacer uso de una determinada funcionalidad que todavía está en el Backlog.

- **Reducción de riesgos**: El hecho de desarrollar, en primer lugar, las funcionalidades de mayor valor y de saber la velocidad a la que el equipo avanza en el proyecto, permite despejar riesgos efectivamente de manera anticipada.

2.3 El patrón de desarrollo MVC (Model-View-Controller)

El patrón Modelo-Vista-Controlador, MVC (por sus siglas en ingles Model-View-Controller) se define formalmente como un patrón de arquitectura de software, que separa los datos y la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado, define componentes para la representación de la información, y por otro lado para la interacción del usuario.

Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

Como lo explica su nombre el patrón MVC considera tres piezas fundamentales en el desarrollo de piezas de software, la idea principal parte de separar la interfaz de usuario, los modelos y los controladores donde cada elementos puede evidenciarse de forma simple en una aplicación de la siguiente forma: la vista (que es la pantalla que ve el usuario) se crea llamando al Modelo (según sea necesario para obtener información) y el Controlador posteriormente (responde a las solicitudes del usuario, interactuando con la Vista y los Modelos u otros controladores según sea necesario).

La principal ventaja del patrón MVC es el acoplamiento flojo. Todas las capas están separadas con su propia funcionalidad. Es fácil reemplazar una capa con algún otro tipo de capa. En otras palabras, el patrón MVC consiste en dividir el comportamiento de la interfaz de usuario en partes separadas para aumentar las posibilidades de reutilización y la capacidad de prueba.

Formalmente podemos definir cada elemento de este patrón de la siguiente forma:

- **Modelo**: Deben ser los responsables de representar los datos en el dominio de la aplicación.

- **Vista**: Presenta la visualización del modelo en la interfaz de usuario, son todos los elementos gráficos del programa con los que interactúa el usuario.

- **Controlador**: Es realmente el corazón del MVC, el intermediario que une el Modelo y la Vista, es decir, toma la entrada del usuario, manipula el modelo y hace que la vista se actualice.

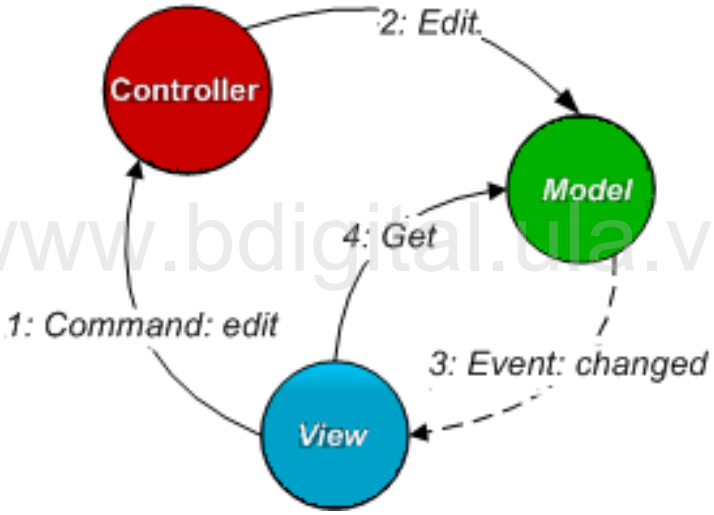


Figura 4: Diagrama del patrón MVC (Model-View-Controller), tomado de Internet, https://www.codeproject.com/KB/dialog/383153/mvc.PNG

2.4 Framework de desarrollo de Windows MVC 4 en .NET C#/WinForm

Para mostrar un ejemplo del desarrollo usando este patrón en el Framework de desarrollo de Windows MVC 4, usaremos un pequeño trozo de la aplicación desarrollada que no infrinja con los acuerdos de confidencialidad firmados con la empresa, es decir, bajo su consentimiento, para dar una

idea más clara de cómo se integra a nivel de codificación estos elementos en cualquier solución o aplicación.

Nota: Las siguientes imágenes fueron modificadas para no mostrar datos privados pertenecientes a los usuarios de los cuales fueron realizadas las capturas de pantalla.

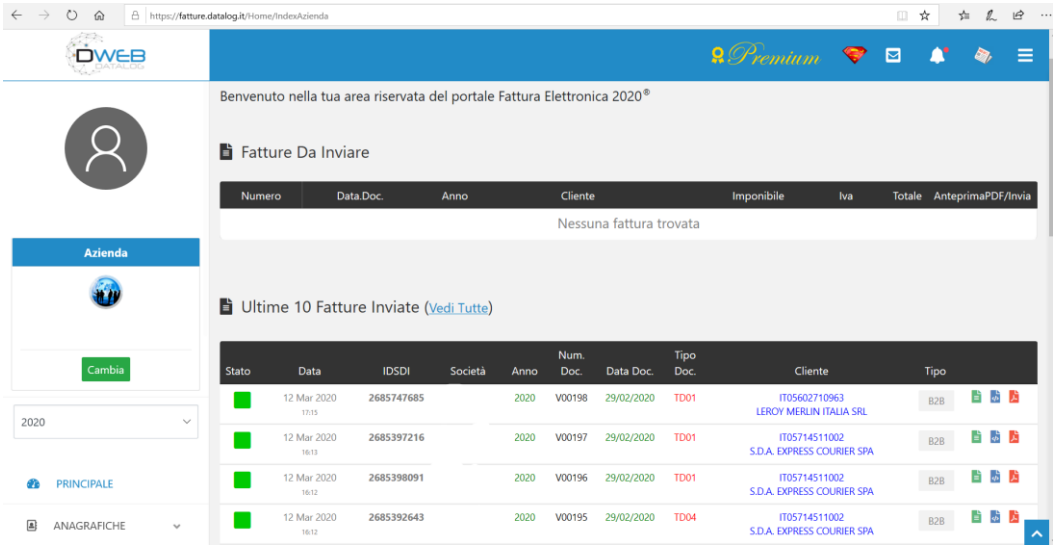


Figura 5: Representación de la Vista

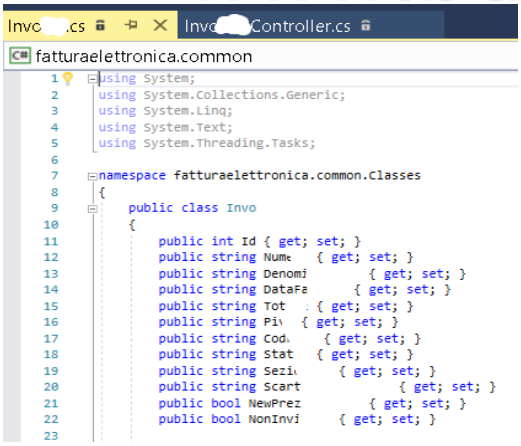


Figura 6: Representación del Modelo

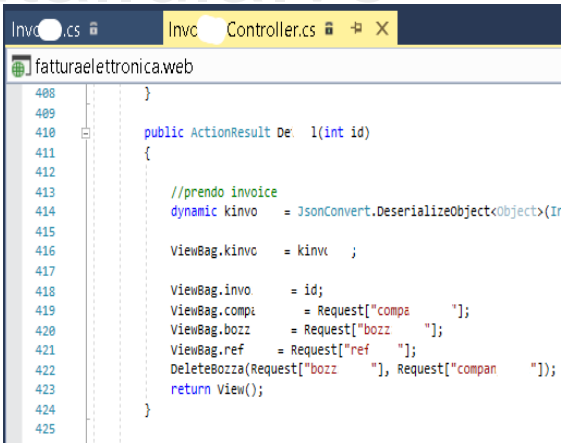


Figura 7: Representación del Controlador

Podemos observar como lo definido anteriormente, se aplica de forma que cada elemento del patrón es independiente y por lo tanto aumenta las posibilidades de reutilización, de facilidad en la identificación de cada elemento para las posibles ediciones o nuevos desarrollos y la velocidad para individuar cada elemento en la fase de prueba.

2.5 Microsoft Visual Studio

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para Windows, Linux y macOS. Es compatible con múltiples lenguajes de programación, tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web, como ASP.NET MVC, Django, entre otros, a lo cual hay que sumarle las nuevas capacidades en línea bajo Windows Azure en forma del editor Monaco.

Visual Studio permite a los desarrolladores crear sitios y aplicaciones web, así como servicios web en cualquier entorno compatible con la plataforma .NET (a partir de la versión .NET 2002). Así, se pueden crear aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles, dispositivos embebidos y videoconsolas, entre otros.

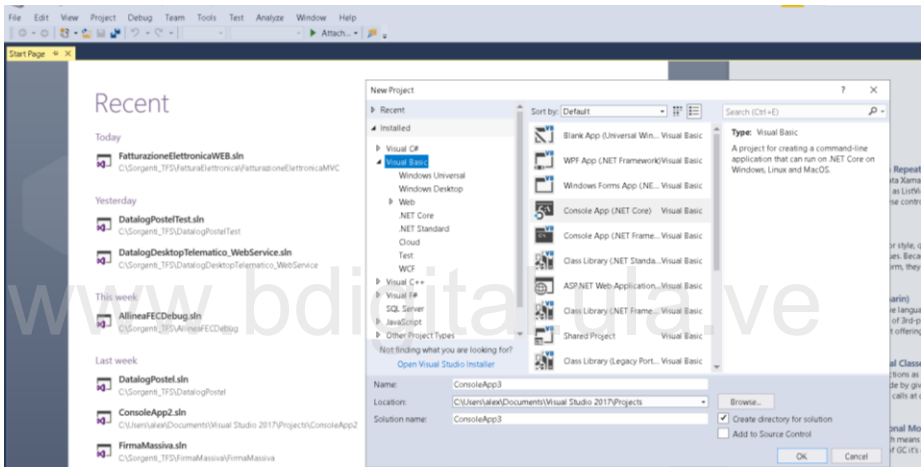


Figura 8: IDE Microsoft Visual Studio

A partir de la versión 2005, Microsoft lanzó gratuitamente las ediciones Express, que son versiones básicas separadas por lenguajes de programación o plataforma orientadas a estudiantes y programadores aficionados. Estas ediciones son similares a las ediciones comerciales, pero carecen de ciertas características avanzadas de integración. Dichas ediciones son:

1. Visual Basic Express Edition.
2. Visual C# Express Edition.
3. Visual C++ Express Edition.
4. Visual Web Developer Express Edition (para programar en ASP.NET).
5. Visual F# (Apareció en Visual Studio 2010, es parecido al J#).
6. Windows Phone 8 SDKk.
7. Windows Azure SDK.

En sus últimas versiones, después de la adquisición de la compañía de software estadounidense Xamarin, el Visual Studio cuenta con la integración del SDK de Xamarin, el cual es una herramienta que permite a los desarrolladores escribir sus Aplicaciones Móviles, partiendo del lenguaje C# y que el mismo código sea traducido para la creación de aplicaciones para Android, IOS y Windows Phone, usando los recursos nativos de cada plataforma.

2.6 Programación orientada a objetos (POO)

La Programación Orientada a Objetos (POO), es un paradigma de programación que viene a innovar la forma de obtener resultados y hacer una representación de los tipos de datos abstractos. Los objetos manipulan los datos de entrada para la obtención de datos de salida específicos, donde cada objeto ofrece una funcionalidad especial.

Está basada en varias técnicas del sexenio: herencia, cohesión, abstracción, polimorfismo, acoplamiento y encapsulamiento. Su uso se popularizó a principios de la década de 1990. En la actualidad, existe una gran variedad de lenguajes de programación que soportan la orientación a objetos. Los objetos prediseñados de estos lenguajes permiten la agrupación en bibliotecas o librerías, sin embargo, muchos de estos lenguajes permiten al usuario la creación de sus propias bibliotecas.

La programación orientada a objetos difiere de la programación estructurada tradicional, en la que los datos y los procedimientos están separados y sin relación, ya que lo único que se busca es el procesamiento de unos datos de entrada para obtener otros de salida. La programación estructurada anima al programador a pensar sobre todo en términos de procedimientos o funciones, y en segundo lugar en las estructuras de datos que esos procedimientos manejan. En la programación estructurada solo se escriben funciones que procesan datos. Los programadores que emplean POO, en cambio, primero definen objetos para luego enviarles mensajes solicitándoles que realicen sus métodos por sí mismos.

Para realizar programación orientada a objetos, existen dos corrientes principales de desarrollo definidas a continuación:

- **Basada en clases**: Es la más ampliamente usada por los lenguajes de programación orientada a objetos. Por ejemplo, es usada por Java, C++ y C#. Se basa en crear una estructura molde llamada clase donde se especifican los campos y métodos que tendrán nuestros objetos. Cada vez que necesitamos un objeto creamos una instancia (o copia del objeto) usando la clase como molde.

- **Basada en prototipos**: Es soportado en JavaScript, Python y Ruby. No hay clases, solo hay objetos. El mecanismo para la reutilización está dado por la clonación de objetos. Se crean directamente los objetos y cuando se quiere generar otro con la misma estructura se usa clonación. Una vez clonado si queremos podemos agregar los campos y métodos necesarios. Un objeto prototípico es un objeto que se utiliza como una plantilla a partir de la cual se obtiene el conjunto inicial de propiedades de un objeto. Cualquier objeto puede ser utilizado como el prototipo de otro objeto, permitiendo al segundo objeto compartir las propiedades del primero.

En resumen, la POO es un paradigma surgido en los años 1970, que utiliza objetos como elementos fundamentales en la construcción de la solución. Un objeto es una abstracción de algún hecho o ente del mundo real, con atributos que representan sus características o propiedades, y métodos que emulan su comportamiento o actividad. Todas las propiedades y métodos comunes a los objetos se encapsulan o agrupan en clases. Una clase es una plantilla, un prototipo para crear objetos; en general, se dice que cada objeto es una instancia o ejemplar de una clase.

2.6.1 Orígenes de la POO

Los conceptos de la POO tienen origen en el lenguaje diseñado para hacer simulaciones Simula 67, creado por Ole-Johan Dahl y Kristen Nygaard, (1967) del Centro de Cómputo Noruego en Oslo. En este centro se trabajaba en simulaciones de naves, que fueron confundidas por la explosión combinatoria de cómo las diversas cualidades de diferentes naves podían afectar unas a las otras. La idea surgió al agrupar los diversos tipos de naves en diversas clases de objetos, siendo responsable cada clase de objetos de definir sus "propios" datos y comportamientos. Fueron refinados más tarde en Smalltalk, desarrollado en Simula en Xerox PARC (cuya primera versión fue escrita sobre Basic) pero diseñado para ser un sistema completamente dinámico en el cual los objetos se podrían crear y modificar "sobre la marcha" (en tiempo de ejecución) en lugar de tener un sistema basado en programas estáticos.

La POO se fue convirtiendo en el estilo de programación dominante a mediados de los años 1980, en gran parte debido a la influencia de C++, una extensión del lenguaje de programación C. Su dominación fue consolidada gracias al auge de las interfaces gráficas de usuario, para las cuales la POO está particularmente bien adaptada. En este caso, se habla también de programación dirigida por eventos.

Las características de orientación a objetos fueron agregadas a muchos lenguajes existentes durante ese tiempo, incluyendo Ada, BASIC, Lisp más Pascal, entre otros. La adición de estas características a los lenguajes que no fueron diseñados inicialmente para ellas condujo a menudo a problemas de compatibilidad y en la capacidad de mantenimiento del código. Los lenguajes orientados a objetos "puros", por su parte, carecían de las características de las cuales muchos programadores habían venido a depender. Para saltar este obstáculo, se hicieron muchas tentativas para crear nuevos lenguajes basados en métodos orientados a objetos, pero permitiendo algunas características imperativas de maneras "seguras". El lenguaje de programación Eiffel de Bertrand Meyer fue un temprano y moderadamente acertado lenguaje con esos objetivos, pero ahora ha sido esencialmente reemplazado por Java, en gran parte debido a la aparición de Internet y a la implementación de la máquina virtual Java en la mayoría de navegadores web. PHP en su versión 5 se ha modificado; soporta una orientación completa a objetos, cumpliendo todas las características propias de la orientación a objetos.

2.6.2 Estructura básica de los objetos

Los objetos son entidades que tienen un determinado estado (atributos), comportamiento (método) e identidad, los cuales permiten cumplir con las características de la POO.

La identidad es una propiedad de un objeto que lo diferencia del resto, dicho con otras palabras, es su identificador (concepto análogo al de identificador de una variable o una constante). Mientras que por su parte los métodos (comportamiento) y atributos (estado) están estrechamente relacionados por la propiedad de conjunto. Esta propiedad destaca que una clase requiere de métodos para poder tratar los atributos con los que cuenta.

El programador debe pensar indistintamente en los conceptos de los métodos y los atributos en un objeto, sin separar ni darle mayor importancia a alguno de ellos. Hacerlo podría producir el hábito erróneo de crear clases contenedoras de información por un lado y clases con métodos que manejen a las primeras por el otro. De esta manera se estaría realizando una "programación estructurada camuflada" en un lenguaje de POO, lo cual no cumple con las características que buscan resolver los problemas por los que se originó este paradigma de la programación.

Es por esto que, al momento de utilizar la POO, es importante tener en cuenta la estructura básica con la que se deben definir los objetos lo cual permitirá llegar a una programación adecuada según las bases de este paradigma de la programación.

2.6.3 Características de la POO

Desde el punto de definición del paradigma de la POO existen muchas características según como ha sido presentada por cada lenguaje de programación que ha realizado su implementación y los objetivos en que ha enfocado su desarrollo desde sus orígenes, sim embargo con el pasar del tiempo se ha llegado a un consenso acerca de qué características contempla la "orientación a objetos". Las características presentadas a continuación son las más importantes:

- **Abstracción**: Denota las características esenciales de un objeto, donde se capturan sus comportamientos. Cada objeto en el sistema sirve como modelo de un "agente" abstracto que puede realizar trabajo, informar y cambiar su estado, y "comunicarse" con otros objetos en el sistema sin revelar "cómo" se implementan estas características. Los procesos, las funciones o los métodos pueden también ser abstraídos, y, cuando lo están, una variedad de técnicas es requeridas para ampliar una abstracción. El proceso de abstracción permite seleccionar las características relevantes dentro de un conjunto e identificar comportamientos comunes para definir nuevos tipos de entidades en el mundo real. La abstracción es clave en el proceso de análisis y diseño orientado a objetos, ya que mediante ella podemos llegar a armar un conjunto de clases que permitan modelar la realidad o el problema que se quiere atacar.

- **Encapsulamiento**: Significa reunir todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción. Esto permite aumentar la cohesión (diseño estructurado) de los componentes del sistema. Algunos autores confunden este concepto con el principio de ocultación, principalmente porque se suelen emplear conjuntamente.

- **Polimorfismo**: Comportamientos diferentes, asociados a objetos distintos que pueden compartir el mismo nombre, al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando. O, dicho de otro modo, las referencias y las colecciones de objetos pueden contener objetos de diferentes tipos, y la invocación de un comportamiento en una referencia producirá el comportamiento correcto para el tipo real del objeto referenciado. Cuando esto ocurre en "tiempo de ejecución", esta última característica se llama asignación tardía o asignación dinámica. Algunos lenguajes proporcionan medios más estáticos (en "tiempo de compilación") de polimorfismo, tales como las plantillas y la sobrecarga de operadores de C++.

- **Herencia**: Las clases no se encuentran aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento, permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que volver a implementarlo. Esto suele hacerse habitualmente agrupando los objetos en clases, y estas en árboles o enrejados que reflejan un comportamiento común. Cuando un objeto hereda de más de una clase, se dice que hay herencia múltiple; siendo de alta complejidad técnica por lo cual suele recurrirse a la herencia virtual para evitar la duplicación de datos.

- **Modularidad**: Se denomina "modularidad" a la propiedad que permite subdividir una aplicación en partes más pequeñas (llamadas módulos), cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las restantes partes. Estos módulos se pueden compilar por separado, pero tienen conexiones con otros módulos. Al igual que la encapsulación, los lenguajes soportan la modularidad de diversas formas.

- **Principio de ocultación**: Cada objeto está aislado del exterior, es un módulo natural, y cada tipo de objeto expone una "interfaz" a otros objetos que especifica cómo pueden interactuar con los objetos de la clase. El aislamiento protege a las propiedades de un objeto contra su modificación por quien no tenga derecho a acceder a ellas; solamente los propios métodos internos del objeto pueden acceder a su estado. Esto asegura que otros objetos no puedan cambiar el estado interno de un objeto de manera inesperada, eliminando efectos secundarios e interacciones inesperadas. Algunos lenguajes relajan esto, permitiendo un acceso directo a los datos internos del objeto de una manera controlada y limitando el grado de abstracción. La aplicación entera se reduce a un agregado o rompecabezas de objetos.

- **Recolección de basura**: La recolección de basura (garbage collection) es la técnica por la cual el entorno de objetos se encarga de destruir automáticamente, y por tanto desvincular la memoria asociada, los objetos que hayan quedado sin ninguna referencia a ellos. Esto significa que el programador no debe preocuparse por la asignación o liberación de memoria, ya que el entorno la asignará al crear un nuevo objeto y la liberará cuando nadie lo esté usando. En la mayoría de los lenguajes híbridos que se extendieron para soportar el Paradigma de

Programación Orientada a Objetos como C++ u Object Pascal, esta característica no existe y la memoria debe desasignarse expresamente.

2.6.4 Conceptos fundamentales en la POO

La POO es una forma de programar que trata de encontrar una solución a estos problemas. Introduce nuevos conceptos, que superan y amplían conceptos antiguos ya conocidos. Entre ellos destacan los siguientes:

- **Clase**: Una clase es una especie de "plantilla" en la que se definen los atributos y métodos predeterminados de un tipo de objeto. Esta plantilla se crea para poder crear objetos fácilmente. Al método de crear nuevos objetos mediante la lectura y recuperación de los atributos y métodos de una clase se le conoce como instanciación.

- **Herencia**: Por ejemplo, herencia de la clase C a la clase D, es la facilidad mediante la cual la clase D hereda en ella cada uno de los atributos y operaciones de C, como si esos atributos y operaciones hubiesen sido definidos por la misma D. Por lo tanto, puede usar los mismos métodos y variables registrados como "públicos" (public) en C. Los componentes registrados como "privados" (private) también se heredan, pero se mantienen escondidos al programador y sólo pueden ser accedidos a través de otros métodos públicos. Para poder acceder a un atributo u operación de una clase en cualquiera de sus subclases, pero mantenerla oculta para otras clases es necesario registrar los componentes como "protegidos" (protected), de esta manera serán visibles en C y en D, pero no en otras clases.

- **Objeto**: Instancia de una clase. Entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad (métodos), los mismos que consecuentemente reaccionan a eventos. Se corresponden con los objetos reales del mundo que nos rodea, o con objetos internos del sistema (del programa).

- **Método**: Algoritmo asociado a un objeto (o a una clase de objetos), cuya ejecución se desencadena tras la recepción de un "mensaje". Desde el punto de vista del comportamiento, es lo que el objeto puede hacer. Un método puede producir un cambio en las propiedades del objeto, o la generación de un "evento" con un nuevo mensaje para otro objeto del sistema.

- **Evento**: Es un suceso en el sistema (tal como una interacción del usuario con la máquina, o un mensaje enviado por un objeto). El sistema maneja el evento enviando el mensaje adecuado al objeto pertinente. También se puede definir como evento la reacción que puede desencadenar un objeto; es decir, la acción que genera.

- **Propiedad o atributo**: Contenedor de un tipo de datos asociados a un objeto (o a una clase de objetos), que hace los datos visibles desde fuera del objeto y esto se define como sus características predeterminadas, y cuyo valor puede ser alterado por la ejecución de algún método.

- **Mensaje**: Una comunicación dirigida a un objeto, que le ordena que ejecute uno de sus métodos con ciertos parámetros asociados al evento que lo generó.

- **Estado interno**: Es una variable que se declara privada, que puede ser únicamente accedida y alterada por un método del objeto, y que se utiliza para indicar distintas situaciones posibles para el objeto (o clase de objetos). No es visible al programador que maneja una instancia de la clase.

- **Miembros de un objeto**: Atributos, identidad, relaciones y métodos.

- **Identificación de un objeto**: Un objeto se representa por medio de una tabla o entidad que esté compuesta por sus atributos y funciones correspondientes.

En comparación con un lenguaje imperativo, una "variable" no es más que un contenedor interno del atributo del objeto o de un estado interno, así como la "función" es un procedimiento interno del método del objeto.

2.7 Application Programming Interface (API)

En el entorno del mundo Web, La Application Programming Interface (por sus siglas en Ingles API), también conocida como la interfaz de programación de aplicaciones en español, es un conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a

objetos) que ofrece acceso a funciones de una cierta biblioteca o un determinado software, para ser utilizado por otro software como una capa de abstracción.

Son publicadas por los constructores de software para permitir acceso a características de bajo nivel o propietarias, detallando solamente la forma en que cada rutina debe ser llevada a cabo y la funcionalidad que brinda, sin otorgar información acerca de cómo se lleva a cabo la tarea. Son utilizadas por los programadores para construir sus aplicaciones sin necesidad de volver a programar funciones ya hechas por otros, reutilizando código que se sabe que está probado y que funciona correctamente.

En la web, las API's son publicadas por sitios para brindar la posibilidad de realizar alguna acción o acceder a alguna característica o contenido que el sitio provee. Algunas de las más conocidas son las API's de:

- Servicios de Google (Search, Maps, Translator, Contacts, entre muchos otros).
- Amazon.
- Block Chains y Sistemas de Trading.
- MEGA.
- Redes sociales como: Twitter, Facebook, Instagram, entre otros.

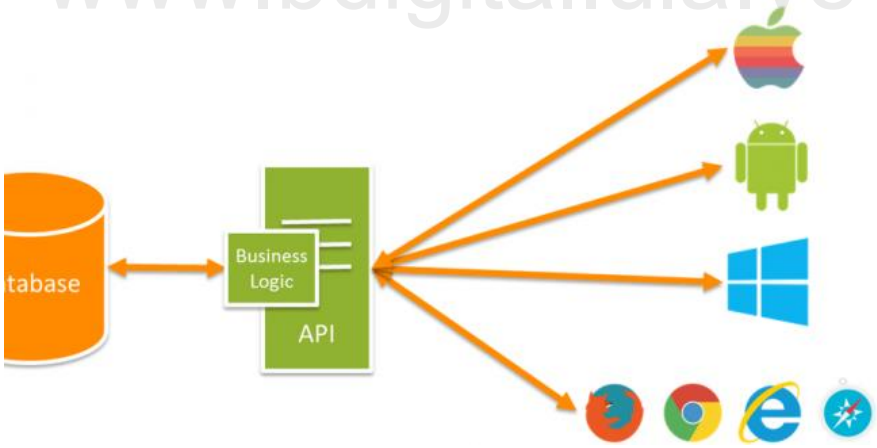


Figura 9: Diagrama de Comunicación API, tomado de Internet,
https://blog.comunicafacile.eu/wp-content/uploads/2016/05/webapi-660x330.png

Las API se desarrollan para permitir también reutilización del código y funcionalidad desarrolladas previamente, para ser compartidas entre diferentes tipos de aplicaciones, ya sean web, de escritorio o móviles, lo que permite la versatilidad en desarrollos de librerías o Dll que pueden ser utilizados por diversos programas que comparten entre si su información, sus datos y su funcionalidad.

De esta forma un desarrollador o programador que conozca en entorno de diferentes aplicaciones y gracias al uso de los servicios web y la POO, puede implementar librerías propias que contengan funcionalidad destinada a diversas plataformas y aplicaciones, que además pueden gracias a esto contar con datos compartidos en tiempo real, ampliando el potencial y las limitaciones de sus aplicaciones.

Es importante destacar que esta definición se relaciona al entorno de las API's WEB (REST y SOAP) y que no se debe confundir con otras definiciones del mismo término en el mundo informática, como es la definición de API a nivel de sistemas operativos, que son las capas que permiten la comunicación entre las aplicaciones y el sistema operativo del dispositivo, que, a pesar de tener un significado similar, no hace referencia al mismo tipo de aplicación con la cual fue definida en este documento.

2.8 Sistema de Control de Versiones

Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Una versión, revisión o edición de un producto, es el estado en el que se encuentra el mismo en un momento dado de su desarrollo o modificación.

Aunque un sistema de control de versiones puede realizarse de forma manual, es muy aconsejable disponer de herramientas que faciliten esta gestión dando lugar a los llamados sistemas de control de versiones o VCS (del inglés Version Control System). Estos sistemas facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas (por ejemplo, para algún cliente específico). Ejemplos de este tipo de herramientas son entre otros: CVS, Subversion, SourceSafe, ClearCase, Darcs, Bazaar, Plastic SCM, Git, SCCS, Mercurial, Perforce, entre muchos otros.

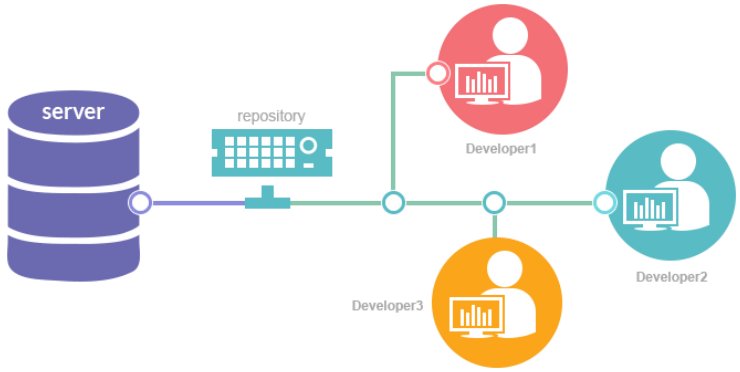


Figura 10: Diagrama de un sistema de control de versiones, tomado de Internet, http://www.ulbegroup.com/wp-content/uploads/2015/11/post1.png

Los sistemas de control de versiones son utilizados generalmente en equipos de desarrollo donde varios miembros del equipo tienen la responsabilidad de desarrollar partes comunes de un determinado proyecto, lo cual permite fácilmente obtener una lista de los cambios proporcionados en cada uno, confrontar sus cambios con las versiones precedentes y unir los diferentes cambios que puedan ser realizados en un archivo de código.

Por lo general cuando existen conflictos el sistema de control de versiones intenta realizar la unión de los cambios proporcionados por diferentes programadores, pero en ocasiones no es capaz de realizar este proceso automáticamente, para resolver este problema estos sistemas cuentan con herramientas que permiten identificar fácilmente donde se encuentran estos conflictos y gestionar la solución de estos conflictos de forma simple.

A pesar de ser usado generalmente por equipos donde participan una gran cantidad de personas, un sistema de control de versiones, también puede ser usado por una persona para mantener versiones funcionales de su proyecto, de manera que pueda revertir los cambios si algún nuevo desarrollo genera errores o no funciona, permitiéndole al programador volver fácilmente a la última versión funcional de su producto.

2.8.1 Terminología Básica

La terminología empleada puede variar de sistema a sistema, pero para ayudar a la comprensión del lector en el desarrollo de los siguientes puntos a continuación se describen algunos términos de uso común:

- **Repositorio**: El repositorio es el lugar en el que se almacenan los datos actualizados e históricos de cambios, a menudo en un servidor. A veces se le denomina depósito o depot. Puede ser un sistema de archivos en un disco duro, un banco de datos, etc.

- **Módulo**: Conjunto de directorios y/o archivos dentro del repositorio que pertenecen a un proyecto común.

- **Revisión ("versión")**: Una revisión es una versión determinada de la información que se gestiona. Hay sistemas que identifican las revisiones con un contador (Ej. subversion). Hay otros sistemas que identifican las revisiones mediante un código de detección de modificaciones (Ej. Git usa SHA1). A la última versión se le suele identificar de forma

especial con el nombre de HEAD. Para marcar una revisión concreta se usan los rótulos o tags.

- **Rotular ("tag")**: Darle a alguna versión de cada uno de los ficheros del módulo en desarrollo en un momento preciso un nombre común ("etiquetar") para asegurarse de reencontrar ese estado de desarrollo posteriormente bajo ese nombre. En la práctica se rotula a todos los archivos en un momento determinado. Para eso el módulo se "congela" durante el rotulado para imponer una versión coherente. Pero bajo ciertas circunstancias puede ser necesario utilizar versiones de algunos ficheros que no coinciden temporalmente con las de los otros ficheros del módulo. Los tags permiten identificar de forma fácil revisiones importantes en el proyecto. Por ejemplo, se suelen usar tags para identificar el contenido de las versiones publicadas del proyecto. En algunos sistemas se considera un tag como una rama en la que los ficheros no evolucionan, están congelados.

- **Línea base ("baseline")**: Una revisión aprobada de un documento o fichero fuente, a partir del cual se pueden realizar cambios subsiguientes.

- **Abrir rama ("branch") o ramificar**: Un módulo puede ser branched o bifurcado en un instante de tiempo de forma que, desde ese momento en adelante se tienen dos copias (ramas) que evolucionan de forma independiente siguiendo su propia línea de desarrollo. El módulo tiene entonces 2 (o más) "ramas". La ventaja es que se puede hacer un "merge" de las modificaciones de ambas ramas, posibilitando la creación de "ramas de prueba" que contengan código para evaluación, si se decide que las modificaciones realizadas en la "rama de prueba" sean preservadas, se hace un "merge" con la rama principal. Son motivos habituales para la creación de ramas la creación de nuevas funcionalidades o la corrección de errores.

- **Desplegar ("Check-out", "checkout", "co")**: Un despliegue crea una copia de trabajo local desde el repositorio. Se puede especificar una revisión concreta, y predeterminadamente se suele obtener la última.

- **Publicar o Enviar ("commit", "check-in", "submit")**: Un commit sucede cuando una copia de los cambios hechos a una copia local es escrita o integrada sobre el repositorio.

- **Conflicto**: Un conflicto ocurre cuando el sistema no puede manejar adecuadamente cambios realizados por dos o más usuarios en un mismo archivo. Por ejemplo, si se da esta secuencia de circunstancias:

    - Los usuarios X e Y despliegan versiones del archivo A en que las líneas n1 hasta n2 son comunes.
    - El usuario X envía cambios entre las líneas n1 y n2 al archivo A.
    - El usuario Y no actualiza el archivo A tras el envío del usuario X.
    - El usuario Y realiza cambios entre las líneas n1 y n2.
    - El usuario Y intenta posteriormente enviar esos cambios al archivo A.
    - El sistema es incapaz de fusionar los cambios. El usuario Y debe resolver el conflicto combinando los cambios, o eligiendo uno de ellos para descartar el otro.

- **Resolver**: El acto de la intervención del usuario para atender un conflicto entre diferentes cambios al mismo archivo.

- **Cambio ("change", "diff", "delta")**: Un cambio representa una modificación específica a un archivo bajo control de versiones. La granularidad de la modificación considerada un cambio varía entre diferentes sistemas de control de versiones.

- **Lista de cambios ("changelist", "change set", "patch")**: En muchos sistemas de control de versiones con commits multi-cambio atómicos, una lista de cambios identifica el conjunto de cambios hechos en un único commit. Esto también puede representar una vista secuencial del código fuente, permitiendo que el mismo, sea examinado a partir de cualquier identificador de lista de cambios particular.

- **Exportación ("export")**: Una exportación es similar a un check-out, salvo porque crea un árbol de directorios limpio sin los metadatos de control de versiones presentes en la copia de trabajo. Se utiliza a menudo de forma previa a la publicación de los contenidos.

- **Importación ("import")**: Una importación es la acción de copia un árbol de directorios local (que no es en ese momento una copia de trabajo) en el repositorio por primera vez.

- **Integración o fusión ("merge")**: Una integración o fusión une dos conjuntos de cambios sobre un fichero o un conjunto de ficheros en una revisión unificada de dicho fichero o ficheros. Esto puede suceder por diversos motivos:

    - Puede suceder cuando un usuario, trabajando en esos ficheros, actualiza su copia local con los cambios realizados, y añadidos al repositorio, por otros usuarios. Análogamente, este mismo proceso puede ocurrir en el repositorio cuando un usuario intenta check-in sus cambios.
    - Puede suceder después de que el código haya sido branched, y un problema anterior al branching sea arreglado en una rama, y se necesite incorporar dicho arreglo en la otra.
    - Puede suceder después de que los ficheros hayan sido branched, desarrollados de forma independiente por un tiempo, y que entonces se haya requerido que fueran fundidos de nuevo en un único trunk unificado.

- **Integración inversa**: El proceso de fundir ramas de diferentes equipos en el trunk principal del sistema de versiones.

- **Actualización ("sync" o "update")**: Una actualización integra los cambios que han sido hechos en el repositorio (por ejemplo, por otras personas) en la copia de trabajo local.

- **Copia de trabajo ("workspace")**: La copia de trabajo es la copia local de los ficheros de un repositorio, en un momento del tiempo o revisión específicos. Todo el trabajo realizado sobre los ficheros en un repositorio se realiza inicialmente sobre una copia de trabajo, de ahí su nombre. Conceptualmente, es un cajón de arena o sandbox.

- **Congelar**: Significa permitir los últimos cambios (commits) para solucionar las fallas a resolver en una entrega (release) y suspender cualquier otro cambio antes de una entrega, con el fin de obtener una versión consistente. Si no se congela el repositorio, un desarrollador podría comenzar a resolver una falla cuya resolución no está prevista y cuya solución dé lugar a efectos colaterales imprevistos.

2.8.2 Arquitecturas de almacenamiento

Podemos clasificar los sistemas de control de versiones atendiendo a la arquitectura utilizada para el almacenamiento del código como:

- **Distribuidos**: cada usuario tiene su propio repositorio. Los distintos repositorios pueden intercambiar y mezclar revisiones entre ellos. Es frecuente el uso de un repositorio, que está normalmente disponible, que sirve de punto de sincronización de los distintos repositorios locales. Ejemplos: Git y Mercurial.

- **Centralizados**: existe un repositorio centralizado de todo el código, del cual es responsable un único usuario (o conjunto de ellos). Se facilitan las tareas administrativas a cambio de reducir flexibilidad, pues todas las decisiones fuertes (como crear una nueva rama) necesitan la aprobación del responsable. Algunos ejemplos son CVS, Subversion o Team Foundation Server.

Entre estas dos diversas arquitecturas de almacenamiento del código podemos encontrar diferentes ventajas, estas son importantes de considerar al momento de la elección para crear el sistema de control de versiones en nuestro proyecto, a continuación, presentaremos las principales ventajas de cada arquitectura:

Ventajas de sistemas distribuidos

- Necesita menos veces estar conectado a la red para hacer operaciones. Esto produce una mayor autonomía y una mayor rapidez.
- Aunque se caiga el repositorio remoto la gente puede seguir trabajando.
- Al hacer de los distintos repositorios una réplica local de la información de los repositorios remotos a los que se conectan, la información está muy replicada y por tanto el sistema tiene menos problemas en recuperarse si por ejemplo se quema la máquina que tiene el repositorio remoto. Por tanto, hay menos necesidad de backups. Sin embargo, los backups siguen siendo necesarios para resolver situaciones en las que cierta información todavía no haya sido replicada.
- Permite mantener repositorios centrales más limpios en el sentido de que un usuario puede decidir que ciertos cambios realizados por él en el repositorio local, no son relevantes para el resto de usuarios y por tanto no permite que esa información sea

accesible de forma pública, ya sea porque contiene versiones inestables, en proceso de codificación o con etiquetas personalizadas y de uso exclusivo del usuario.

- El servidor remoto requiere menos recursos que los que necesitaría un servidor centralizado ya que gran parte del trabajo lo realizan los repositorios locales.
- Al ser los sistemas distribuidos más recientes que los sistemas centralizados, y al tener más flexibilidad por tener un repositorio local y otro u otros remotos, estos sistemas han sido diseñados para hacer fácil el uso de ramas (creación, evolución y fusión) y poder aprovechar al máximo su potencial. Por ejemplo, se pueden crear ramas en el repositorio remoto para corregir errores o crear funcionalidades nuevas. Pero también se pueden crear ramas en los repositorios locales para que los usuarios puedan hacer pruebas y dependiendo de los resultados fusionarlos con el desarrollo principal o no. Las ramas dan una gran flexibilidad en la forma de trabajo.

Ventajas de sistemas centralizados

- En los sistemas distribuidos hay menos control a la hora de trabajar en equipo ya que no se tiene una versión centralizada de todo lo que se está haciendo en el proyecto.
- En los sistemas centralizados las versiones vienen identificadas por un número de versión. Sin embargo, en los sistemas de control de versiones distribuidos no hay números de versión, ya que cada repositorio tendría sus propios números de revisión dependiendo de los cambios. En lugar de eso cada versión tiene un identificador al que se le puede asociar una etiqueta (tag).

2.8.3 Formas de Colaborar

Para colaborar en un proyecto usando un sistema de control de versiones lo primero que hay que hacer es crearse una copia local obteniendo información del repositorio. A continuación, el usuario puede modificar la copia. Existen dos esquemas básicos de funcionamiento para que los usuarios puedan ir aportando sus modificaciones:

- **De forma exclusiva**: en este esquema para poder realizar un cambio es necesario comunicar al repositorio el elemento que se desea modificar y el sistema se encargará de impedir que otro usuario pueda modificar dicho elemento. Una vez hecha la modificación, esta se comparte con el resto de colaboradores. Si se ha terminado de modificar un elemento entonces se libera ese elemento para que otros lo puedan modificar. Este modo de

funcionamiento es el que usa por ejemplo SourceSafe. Otros sistemas de control de versiones (ejemplo: subversion), aunque no obligan a usar este sistema, disponen de mecanismos que permiten implementarlo.

- **De forma colaborativa**: en este esquema cada usuario modifica la copia local y cuando el usuario decide compartir los cambios el sistema automáticamente intenta combinar las diversas modificaciones. El principal problema es la posible aparición de conflictos que deban ser solucionados manualmente o las posibles inconsistencias que surjan al modificar el mismo fichero por varias personas no coordinadas. Además, esta semántica no es apropiada para ficheros binarios. Los sistemas de control de versiones subversion o Git permiten implementar este modo de funcionamiento.

El control de versiones Team Foundation Server permite escoger cualquiera de las dos formas de colaboración.

2.8.4 Uso de Ramas

Las ramas, en un sistema de control de versiones, constituyen una potente herramienta que flexibiliza la forma en la que los colaboradores cooperan en el proyecto (en inglés Branching Workflows). Las ramas son solo una herramienta que es posible utilizar de distintas formas para conseguir distintos objetivos. A continuación, podemos apreciar algunas posibilidades:

- **Ramas de largo recorrido**: En algunos proyectos se tienen varias ramas siempre abiertas, que indican diversos grados de estabilidad del contenido. Por ejemplo, en la rama 'master' es frecuente mantener únicamente lo que es totalmente estable. Luego se tienen otras ramas que revelan distintos grados de estabilidad. Por ejemplo, podríamos tener una rama 'beta' (versión beta) y otra 'alfa' (versión alfa), en las que se va trabajando. Cuando se alcanza cierto grado de estabilidad superior a la rama en la que se está entonces se realiza una fusión con rama de estabilidad superior.

- **Ramas puntuales**: Las ramas puntuales, también llamadas ramas de soporte, son ramas que se crean de forma puntual para realizar una funcionalidad muy concreta. Por ejemplo, añadir una nueva característica (se les llama ramas de nueva funcionalidad o directamente por su

nombre en inglés topic branch o feature branch) o corregir un fallo concreto (se les llama ramas para corregir error o directamente por su nombre en inglés hotfix branch).

Este tipo de ramas permiten trabajar centrándonos exclusivamente en el desarrollo de una característica concreta y cuando esta esté concluida se fusiona con una de las ramas de largo recorrido (normalmente con la de más bajo nivel de estabilidad, para que sea probada en ese entorno). La fusión solo se realiza cuando se está 'seguro' de que esa característica está correctamente implementada, en lugar de fusionar en el orden que se van desarrollando las cosas. Esto permite por un lado tener un historial de las distintas versiones que se han tenido hasta conseguir la funcionalidad. Por otro lado, permiten que el historial de las ramas de largo recorrido no sea 'ensuciados' con distintas modificaciones relativas a una funcionalidad concreta.

El uso de este tipo de ramas permite más flexibilidad a la hora de probar posibles soluciones. Solo se fusiona con ramas de largo recorrido una vez que estamos seguros de elegir la solución mejor.

Un tipo de ramas de este tipo que tienen un funcionamiento especial son las llamadas ramas de versión o ramas de release (en inglés release branch). Este tipo de ramas se crean para dar soporte a la preparación de una nueva versión de producción. Permiten tener bajo control el contenido de la versión y poder realizar cierto mantenimiento sobre ella (añadirle pequeñas mejoras y corrección de errores).

Es frecuente y una buena práctica utilizar en el nombre de la rama un prefijo que indique el tipo de rama de la que se trata. Por ejemplo, podría usar ‘feature-’ para ramas de nueva funcionalidad, ‘hotfix-’ para ramas que arreglan errores, y ‘release-’ para ramas de versión.

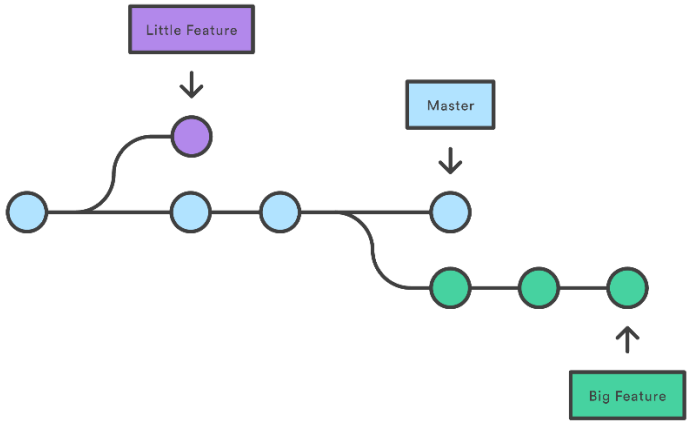


Figura 11: Ejemplo del uso de la Ramas (Branching), tomado de Internet, http://www.ulbegroup.com/wp-content/uploads/2015/11/post1.png

2.8.5 Flujos de Trabajo

El flujo de trabajo de un sistema de control de versiones indica cómo se relacionan los distintos usuarios para colaborar entre sí en la consecución de los objetivos del proyecto.

En los sistemas de control de versiones centralizados el diseño del sistema restringe la forma en que los distintos usuarios colaboran entre sí. Sin embargo, en los sistemas de control de versiones distribuidos hay mucha más flexibilidad en la forma de colaborar ya que cada desarrollador puede tanto contribuir como recibir contribuciones. Hay distintos flujos de trabajo, donde cada uno se adapta mejor a cierto tipo de proyectos. Veamos algunos ejemplos habituales que también se pueden combinar para así adaptarse mejor al proyecto concreto (por ejemplo, podría usarse en general un flujo centralizado, y usar un gestor de integraciones para ciertas partes críticas y de especial importancia).

- **Flujo de trabajo centralizado**: En el flujo de trabajo centralizado (en inglés Centralized Workflow) cada desarrollador es un nodo de trabajo. Por otro lado, hay un repositorio remoto central que funciona a modo de punto de sincronización. Todos los nodos de trabajo operan en pie de igualdad sobre el repositorio remoto central.

  Si se trata de un sistema de control de versiones distribuido cada nodo de trabajo consiste en un repositorio local privado.

  Una desventaja de este modo de trabajo es que, si dos usuarios clonan desde un punto central, y ambos hacen cambios; tan solo el primero que envíe sus cambios lo podrá hacer limpiamente. El segundo desarrollador deberá fusionar previamente su trabajo con el del primero, antes de enviarlo, para evitar sobrescribir los cambios del primero. Es decir, es necesario hacer un paso previo ya que no se pueden subir cambios no directos (non-fast-forward changes).

- **Flujo de trabajo con un Gestor de Integraciones**: En el flujo de trabajo con un Gestor de Integraciones (en inglés Integration-Manager Workflow) en el que cada desarrollador tiene acceso de escritura a un repositorio propio público y acceso de lectura a los repositorios públicos de todos los demás usuarios. Por otro lado, hay un repositorio canónico, representante 'oficial' del proyecto.

  Para contribuir en estos proyectos cada desarrollador crea su propio clon público del repositorio canónico y envía sus cambios (realizados en un repositorio privado) a él. Para

'subir' sus cambios al repositorio canónico cada desarrollador tiene que realizar una petición a la persona gestora del mismo.

La principal ventaja de esta forma de trabajar es que puedes continuar trabajando, y la persona gestora del repositorio canónico podrá recuperar tus cambios en cualquier momento. Las personas colaboradoras no tienen por qué esperar a que sus cambios sean incorporados al proyecto, cada cual puede trabajar a su propio ritmo.

- **Flujo de trabajo con Dictador y Tenientes**: El flujo de trabajo con Dictador y Tenientes (en inglés Dictator and Lieutenants Workflow) es una ampliación del flujo de trabajo con gestor de integraciones. Es utilizado en proyectos muy grandes, con cientos de colaboradores, como el kernel de Linux.

  Hay una serie de gestores de integración que se encargan de partes concretas del repositorio, a los que se denominan tenientes. Todos los tenientes rinden cuentas a un gestor de integración; conocido como el dictador. El dictador integra todos los aportes de los tenientes publicando el trabajo en un repositorio de referencia del que recuperan todos los colaboradores.

  Este sistema de trabajo permite al líder del grupo (el dictador) delegar gran parte del trabajo en los tenientes, relegando su trabajo en recolectar el fruto de múltiples puntos de trabajo.

2.9 Autenticación por Token

Una práctica recurrente a la hora de mantener la seguridad de un API REST, es el uso de la autenticación por token. Es habitual en arquitecturas REST, aunque también puede ser usado para cualquier otra, por ejemplo, GraphQL u otros mecanismos para implementar servicios web.

La autenticación por token se debe realizar del lado del servidor y por tanto su implementación depende en gran medida de las tecnologías que estemos usando en el Backend. Sin embargo, siempre se trabaja con el mismo flujo de aplicación y los conceptos básicos que vamos a describir a continuación son perfectamente válidos para cualquier lenguaje, base de datos o tipo de servidor que podamos estar usando.

Si hay algo característico de las API, REST o cualquier otra arquitectura, es que no manejan estado y por lo tanto no tienen sesiones. Por lo tanto, es destacable que, como consecuencia de no tener estado y no tener sesiones en el servidor para cada usuario, el backend es incapaz de recordar al usuario entre las diversas llamadas que se realizan a la API.

Para evitar que en cada llamada el usuario deba entregar su clave y password se usa un token, que no es más que una cadena de text, generalmente cifrada con una clave. Cuando se realiza el login, el servidor devuelve al cliente un token y el cliente en futuros accesos entrega ese token para certificar que está autenticado y autorizado a ver la información que desea.

Por lo tanto, el flujo de comunicación, autenticación y entrega del token al usuario autorizado se puede desarrollar en tres simples pasos, los cuales se representan a través del siguiente diagrama:

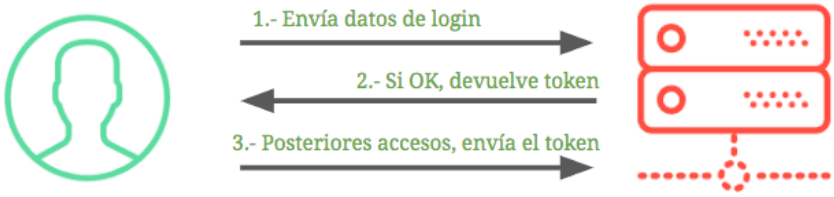


Figura 12: Flujo de autenticación y autorización por Token, tomado de Internet, https://desarrolloweb.com/archivoimg/general/4483.png

1. **El cliente envía los datos de login**: A través de un formulario en la página se recaba el usuario y la clave de acceso. Estos datos se enviarán al servidor. En este punto pueden pasar dos cosas:
    - Si el login es correcto, se da al cliente por autenticado. Entonces se genera un token en el lado del servidor, que se enviará al cliente de vuelta.
    - Si el login no fue correcto, entonces se le manda un código de error al cliente.

2. **Almacenar el token**: En el caso que se realizase un intento correcto de autenticación, y por tanto el servidor nos devolviese el token, en el lado del cliente se deberá almacenar el token, para poder usarlo más tarde. Lo ideal será almacenar el token en el navegador, en un lugar donde se pueda acceder a él en futuras consultas. Es decir, sería interesante proveer algún sistema de persistencia, para que, por ejemplo, si el usuario refresca la página o accede a la misma web pasados unos minutos, se disponga del token generado anteriormente y no se tenga que obligar al usuario a pasar de nuevo por el proceso de login. Lo más normal para almacenar el token sería el LocalStorage del navegador.

3. **Envío del token en posteriores accesos**: Cada vez que el cliente, una vez autenticado, desee acceder de nuevo a un recurso de ese API, entonces tendrá que enviar el token al servidor para informarle que es él mismo y que ya se autenticó correctamente en un paso anterior. Al recibir el token, el servidor lo tendrá que verificar y, si lo encuentra válido, sabrá

que corresponde a un usuario de su aplicación, por lo que podrá conceder el acceso al recurso consultado.

En este flujo, aunque no se menciona, se deriva la caducidad del token. El token generalmente tiene un tiempo de validez, que se establece en el servidor, al momento de generar el token. Durante el tiempo de validez del token podrá ser utilizado para autorizar el acceso a ciertos recursos por parte del cliente autenticado. Sin embargo, si el token ha caducado, entonces el servidor lo rechazará y el cliente no podrá acceder a aquella información u operación solicitada. Por lo tanto, por lo general se procede a redireccionar hacia la página de inicio de sesión o login.

Para establecer un flujo mediante el cual se mitiguen posibles problemas por caducidad del token, se puede proporcionar también un token de refresco. Ese token servirá para aumentar el tiempo de validez del token o generar uno nuevo más adelante, sin necesidad de obligar al usuario a volver a enviar sus datos de inicio de sesión.

El token de refresco se utiliza opcionalmente, por lo que no todas las implementaciones deben controlarlo necesariamente.

2.10 Cifrado y Descifrado de datos

En el mundo de la informática, el cifrado es la conversión de datos de un formato legible a un formato codificado, que solo se pueden leer o procesar después de haberlos descifrado. El cifrado es el elemento fundamental de la seguridad de datos y es la forma más simple e importante de impedir que alguien robe o lea la información de un sistema informático con fines malintencionados. Utilizado tanto por usuarios individuales como por grandes corporaciones, el cifrado se usa ampliamente en Internet para garantizar la inviolabilidad de la información personal enviada entre navegadores y servidores. Dicha información podría incluir todo, desde datos de pagos hasta información personal. Las empresas de todos los tamaños suelen utilizar el cifrado para proteger los datos confidenciales en sus servidores y bases de datos.

Figura 13: Diagrama del proceso de cifrado, tomado de Internet, https://www.osi.es/sites/default/files/images/concienciacion/c9-img-blog-criptografia.png

Más allá de la ventaja obvia de proteger la información privada contra robos o amenazas, el cifrado también ofrece un medio para demostrar tanto la autenticidad como el origen de la información. Se puede utilizar para verificar el origen de un mensaje y confirmar que no ha sufrido modificaciones durante la transmisión.

Lo esencial del cifrado es el concepto de algoritmos de cifrado y "claves". Cuando se envía información, esta se cifra mediante un algoritmo y solo se puede descodificar con la clave apropiada. Dicha clave puede almacenarse en el sistema de recepción, o bien, transmitirse junto con los datos cifrados.

Se utilizan una serie de métodos para codificar y descodificar la información, y dichos métodos evolucionan a medida que el software informático y los métodos de intercepción y robo de información cambian. Estos métodos incluyen lo siguiente:

- **Cifrado de clave simétrica**: también conocido como algoritmo de clave secreta, es un singular método de descodificación de mensajes que debe ser provisto al receptor antes de que el mensaje se pueda descodificar. La clave que se usa en la codificación es la misma que se utiliza en la descodificación, lo que resulta más conveniente para los usuarios individuales y los sistemas cerrados. De lo contrario, se le tiene que enviar la clave al receptor, lo que aumenta el riesgo de alteraciones en caso de que terceros, como un hacker, la intercepten. La ventaja es que este método es mucho más ágil que el método asimétrico.

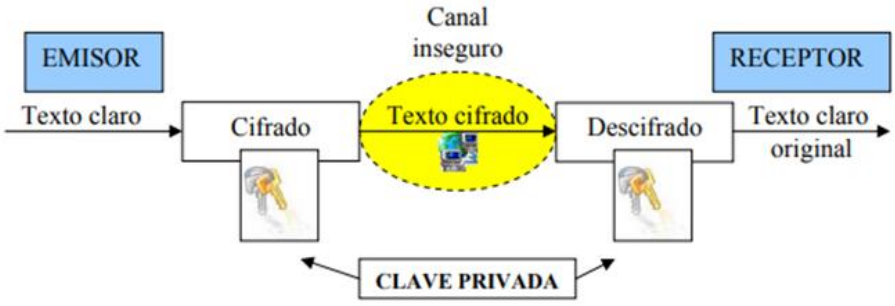


Figura 14: Cifrado de clave simétrica, tomado de Internet,
https://www.osi.es/sites/default/files/images/concienciacion/c9-img-blog-simetrico.png

- **Criptografía asimétrica**: este método utiliza dos claves diferentes (pública y privada), que están vinculadas entre sí matemáticamente. Las claves son solo números extensos vinculados entre sí, pero no son idénticos, de ahí el término asimétrico. La clave pública se puede compartir con cualquier persona, mientras que la clave privada debe mantenerse en secreto. Ambas se pueden usar para cifrar un mensaje, y la clave opuesta a la que se emplee para cifrarlo se utiliza luego para descodificarlo.

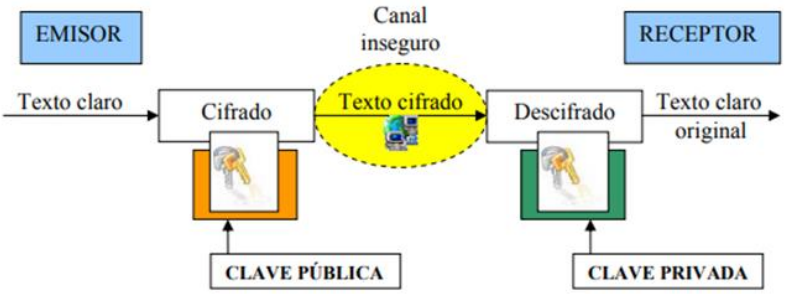


Figura 15: Criptografía asimétrica, tomado de Internet, https://www.osi.es/sites/default/files/images/concienciacion/c9-img-blog-asimetrico.png

2.11 Archivos en formato JSON

El formato de archivo de JavaScript Object Notation (JSON) es un formato estándar abierto basado en texto que se utiliza para serializar y transmitir datos estructurados entre un servidor y una aplicación web. El formato JSON es fácil de leer y escribir para los humanos. También es fácil para las máquinas analizar y generar. Aunque está basado en un subconjunto del lenguaje de programación JavaScript, es completamente independiente del lenguaje. El formato JSON es más pequeño, más rápido y más fácil de analizar que XML. Debido a estas propiedades, el formato JSON es el lenguaje de intercambio de datos ideal. Los tipos de datos en el formato JSON incluyen:

- **Número**: Punto flotante de doble precisión en JavaScript.
- **String**: Unicode con comillas dobles con escape de barra invertida.
- **Boolean**: Verdadero (true) o Falso (false).
- **Formación**: Una secuencia ordenada de valores separados por comas encerrados entre corchetes.
- **Objeto**: Una colección desordenada de clave: pares de valores, con los dos puntos ":" separando la clave y el valor. Es una lista separada por comas encerrado entre llaves.
- **Nulo**: Valor que representa el vacío o nulo (Null).

CAPITULO 3

Facturación electrónica.

El contenido de este capítulo comprende el desarrollo de los conceptos que definen a la factura electrónica y los procesos que se deben ejecutar sobre ella para ser reconocida como un documento fiscalmente valido.

El desarrollo comienza partiendo desde su definición base, documentando los requerimientos establecidos a nivel internacional y termina detallando el caso particular de su implementación en Italia, explicando los requerimientos fiscales que conlleva el proceso de realizar la facturación electrónica, los entes involucrados en los procesos que implican su realización y posterior verificación y toda la documentación disponible de las herramientas que proporciona *L'Agenzia Delle Entrate* como organismo regulador de los procesos fiscales (y por tanto de la facturación electrónica) en el país.

3.1 ¿Que es facturación electrónica?

La factura electrónica es un tipo de factura que se diferencia de la factura en papel por la forma de gestión informática y el envío mediante un sistema de comunicaciones que conjuntamente permiten garantizar la autenticidad y la integridad del documento electrónico. Una factura electrónica se construye en 2 fases:

1. Se crea la factura con las mismas normas que rigen una factura normal en papel y posteriormente se almacena en un fichero de datos.

2. Se procede a el proceso de firma del documento generado con un certificado digital o electrónico propiedad del emisor que cifra el contenido de factura y añade el sello digital a la misma.

Al terminar obtenemos una factura que nos garantiza que la persona física o jurídica que firmó la factura es quien dice ser (autenticidad) puesto que viene firmada con el certificado digital o electrónico que es exclusivo del emisor y que el contenido de la factura no ha sido alterado (integridad).

El emisor envía la factura al receptor mediante medios electrónicos, como pueden ser CD, memorias Flash e incluso Internet. Si bien se dedican muchos esfuerzos para unificar los formatos de factura electrónica, actualmente está sometida a distintas normativas y tiene diferentes requisitos legales exigidos por las autoridades tributarias de cada país, de forma que no siempre es posible el uso de la factura electrónica, especialmente en las relaciones con empresas extranjeras que tienen normativas distintas a la del país destinado a realizar el registro de la factura.

Los requisitos legales respecto al contenido mercantil de las facturas electrónicas son exactamente las mismas que regulan las tradicionales facturas en papel. Los requisitos legales en relación con la forma imponen determinado tratamiento en aras de garantizar la integridad y la autenticidad y ciertos formatos que faciliten la interoperabilidad.

Existen algunas normativas internacionales aplicables de forma general a la factura electrónica, aunque las Naciones Unidas, a través de UN/ROSA han publicado recomendaciones tales como UNeDocs que definen plantillas para las facturas impresas y formatos EDI y XML para las modalidades electrónicas. En Europa, la facturación electrónica se regula en la Directiva 115/2001, que debía ser adoptada en cada país antes del 31 de diciembre de 2003.

Hoy día la organización GS1 (antes EAN/UCC) a nivel mundial ha organizado comités internacionales de usuarios de 108 países miembro, para conformar las guías de facturación electrónica estándar a nivel mundial.

La factura electrónica permite que instituciones, empresas y profesionales dejen atrás las facturas en papel y las reemplacen por la versión electrónica del documento tributario. Tiene exactamente la misma validez y funcionalidad tributaria que la factura tradicional en papel. Todo el ciclo de la facturación puede ser administrado en forma electrónica.

3.2 Elementos básicos y avanzados de la facturación electrónica a nivel internacional

La factura electrónica cuenta con al menos dos elementos básicos que son el Mensaje de Datos basado en el estándar universal y el uso de firmas electrónicas basadas en Infraestructura de Clave Pública, además existe un tercer elemento avanzado que está presente en los sistemas fiscales digitales más maduros que se conoce como la Certificación del documento. Gracias al uso de estos elementos el proceso de la generación de la factura electrónica cuenta con los valores intrínsecos de Autenticidad e Integridad, Confidencialidad y No Repudio, explicados a continuación:

- Autenticidad e Integridad: Al ser enviado el documento con el certificado digital o electrónico, que comúnmente suele ser conocido como la firma del documento y es proporcionada por el emisor, se puede garantizar que el documento ha sido enviado por la persona física o jurídica que firma el documento y que existe la integridad de sus datos, puesto que si desde el inicio existe incongruencias con la firma del documento o este documento viene posteriormente alterado perdiendo la firma realizada, se da por entendido que el documento no es auténtico o que el emisor no es quien dice ser, convirtiéndose automáticamente en un documento descartado.

  La autenticidad asegura que el destinatario de un mensaje y su remitente son los que realmente tienen acceso a los datos y tienen una identidad electrónica verificada, mientras la integridad garantiza la no alteración accidental o intencional del mensaje.

- Confidencialidad: En los pasajes del envió de la factura como documento digital para el proceso certificación del mismo, además de venir firmado, pasa por diversos procesos de otorgación de identificadores por el ente regulador (claves únicas), transformación del documento (en algunos casos cifrado o descifrados a partir de llaves parciales, cuyos únicos conocedores son el emisor y el ente certificador y en otros casos cambios de extensión en los formatos en el documento) para evitar su interceptación. La confidencialidad garantiza que solo el destinatario legítimo tenga acceso a los datos.

- No Repudio: Con respecto a la seguridad digital, el significado criptológico y aplicación de los cambios de no repudio significa, un servicio que proporciona pruebas de la integridad y origen de los datos y además posee una autentificación que con un alto aseguramiento pueda ser reafirmado como genuino. Entonces al certificar que el documento está firmado por el emisor y existen integridad de los datos se puede garantizar que el documento es genuino. El no repudio garantiza que el autor de un mensaje no puede ser desacreditado bajo ninguna circunstancia.

Es importante resaltar que quienes realizan la facturación electrónica no están obligados a usar o comprar software privativo para la generación o el envío de documento, por lo general cada ente fiscal regulador de la factura se encarga de proporcionar un modelo de estos formatos, mediante los cuales, las personas naturales o jurídicas, pueden realizar la construcción de la factura en el formato deseado sin necesidad de apoyarse en software de terceros, además proporcionan los programas necesarios para el proceso de envió y certificación de los documentos, mediante el uso de sus

credenciales particulares. De esta forma cualquier figura fiscal es capaz de crear el documento, realizar el registrarlo y certificarlo, siguiendo la normativa y los requisitos fiscales preestablecidos por el ente regulador.

3.2.1 Mensaje de Datos basado en el estándar universal

El mensaje de datos basado en el estándar universal hace referencia a un fichero de datos que contiene la factura presentada en un formato que es similar al habitual a la presentación de la misma factura hecha en papel, debe cumplir con todas las normativas impuestas por el ente fiscal del país en el cual se desea registrar dicho documento para su aceptación.

Cabe destacar que no existen requisitos formales respecto al formato o la codificación en que se envía la factura, pero las modalidades más habituales son las siguientes:

- PDF: Cuando el destinatario es un particular, un profesional o una PYME cuyo único interés sea guardar electrónicamente la factura, pero no evitar volver a teclear los datos ya que con este formato no se facilita el ingreso de los datos de la factura en el ordenador de destino.

- XML: Cuando el envío es de ordenador a ordenador, puede también utilizarse este tipo de sintaxis. Existen diversas variantes cuya convergencia se espera en el marco de las Naciones Unidas. Las más importantes son UBL respaldado por OASIS y GS1 respaldado por la organización del mismo nombre. En España la variante "facturae" (procedente de CCI-AEAT), respaldada por el Centro de Cooperación Interbancaria, la Agencia Tributaria y el Ministerio de Industria, Turismo y Comercio es la más difundida, y cuenta con sistemas de traducción a y desde UBL.

3.2.2 Firmas electrónicas basadas en la infraestructura de clave pública.

La infraestructura de clave pública o Public Key Infrastructure (PKI) proporciona certificados digitales que permiten hacer las operaciones de cifrado. Estas se utilizan para la verificación y autenticación de las diferentes partes involucradas en un intercambio electrónico. La PKI consiste en

un conjunto de servicios que se basan en el uso del cifrado asimétrico y permiten la administración del ciclo de vida de certificados digitales o electrónicos.

Entonces al realizar la firma del documento en base a una infraestructura de clave pública podemos decir que se realiza un cifrado con clave pública, el cual, partiendo de la definición anterior, es un método de cifrado al que pueden acceder todos los miembros de una organización, permitiendo por un lado transmitir los mensajes de forma confidencial a su único propietario y por otro lado autenticar los mensajes que han sido emitidos por el propietario.

Por lo tanto, la PKI ofrece a sus usuarios un alto nivel de protección de la privacidad, pero también el control de acceso a la información, la integridad, la autenticación y el no repudio durante las transacciones electrónicas.

Para cada formato de los ficheros de datos mencionados existe una forma peculiar de codificar o cifrar la firma electrónica:

- PDF: El formato de firma de Adobe (derivado de PKCS#7) queda embebido dentro del formato PDF y permite asociar una imagen, por lo que es uno de los más adecuados para su visualización. La especificación del formato es la 1.6 y para la visualización se emplea Acrobat Reader v7 o Foxit PDF Reader. La apariencia de la firma es muy visual, ya que es posible asociar a la misma un gráfico como una firma digitalizada o un sello de empresa.

- XML: El formato de firma electrónica se denomina XAdES y se rige por la especificación TS 101 903. De las diferentes modalidades previstas por la norma, la más recomendable es la ES-XL que incluye información sobre el tiempo en el que se llevó a cabo la firma electrónica e información sobre la validez del certificado electrónico cualificado que la acompaña.

3.2.3 Certificación del documento

Consiste en la validación de la sintaxis y el certificado digital del emisor que realiza la administración tributaria o un Tercero en Confianza, para garantizar su coherencia con el estándar definido por la autoridad fiscal correspondiente y la validez de la firma electrónica del emisor.

Cuando ambas validaciones son exitosas, se adiciona al documento un sello digital que certifica la validez de dicha factura y otorga efectos fiscales a la misma a partir de ese momento.

La existencia de terceros da mayor respaldo a la información y agrega un testigo al proceso de la FE. Esta es otra faceta nueva y diferencial, en este caso impulsado por la FE, de la era digital por la cual el sector privado colabora con la AT para cumplir con la función de control (antagónica con otros contribuyentes y represiva de la evasión) que siempre le ha caracterizado.

De acuerdo con la normatividad de la Factura Electrónica vigente en cada legislación, la factura electrónica puede entregarse a la administración tributaria en tiempo real, es decir, en el momento de la celebración de la operación por la que surge la necesidad de emitir un comprobante fiscal, o bien en un momento posterior de acuerdo a la periodicidad de declaraciones o de auditoría que señale el marco jurídico aplicable.

Algunos países han optado ya por obligar al uso de documento electrónico a todos los contribuyentes responsables de impuestos indirectos al consumo, otros han establecido estrategias progresivas para extender el alcance de la misma, por aspectos como actividad o sector económico, volumen de facturación, situación previa de los contribuyentes en relación con la emisión de facturas de papel, total de ventas o ubicación geográfica.

La factura electrónica admite por su naturaleza el uso de medios digitales para su generación, procesamiento, envío, recepción y conservación.

3.3 La facturación electrónica en Italia

Como se puede apreciar del contenido anterior, hemos realizado la definición de la facturación electrónica a un nivel general, sus elementos, sus diversos procesos y sus requerimientos para que sea considerada como un documento digital fiscalmente válido, hasta ahora hemos definido todas los componentes sin entrar en la definición particular, pero a fines del proyecto, necesitamos definir el caso particular para Italia lugar al cual se encuentra destinado la implementación del desarrollo realizado en este proyecto, a continuación veremos los aspectos particulares según la normativa que proporciona *L'Agenzia delle Entrate* que es el ente fiscal oficial encargado de los tramites fiscales en Italia.

3.3.1 L'Agenzia Delle Entrate

L'Agenzia delle Entrate, operativa desde el 1 de Enero del 2001, nace de la reorganización de la *Amministrazione Finanziaria* (Administración de Finanza) a partir del *Decreto Legislativo número 300*, (1999).

Tiene un estatuto propio y reglamentos afines que regulan la administración y la contabilidad. Los órganos de la *Agenzia delle Entrate* están constituidos por el Director, el comité de gestiones y el Colegio de Revisores de las cuentas (Contadores). Desde el 1 de Diciembre del 2012 *L'Agenzia Delle Entrate* ha incorporado a *L'Agenzia del Territorio*.

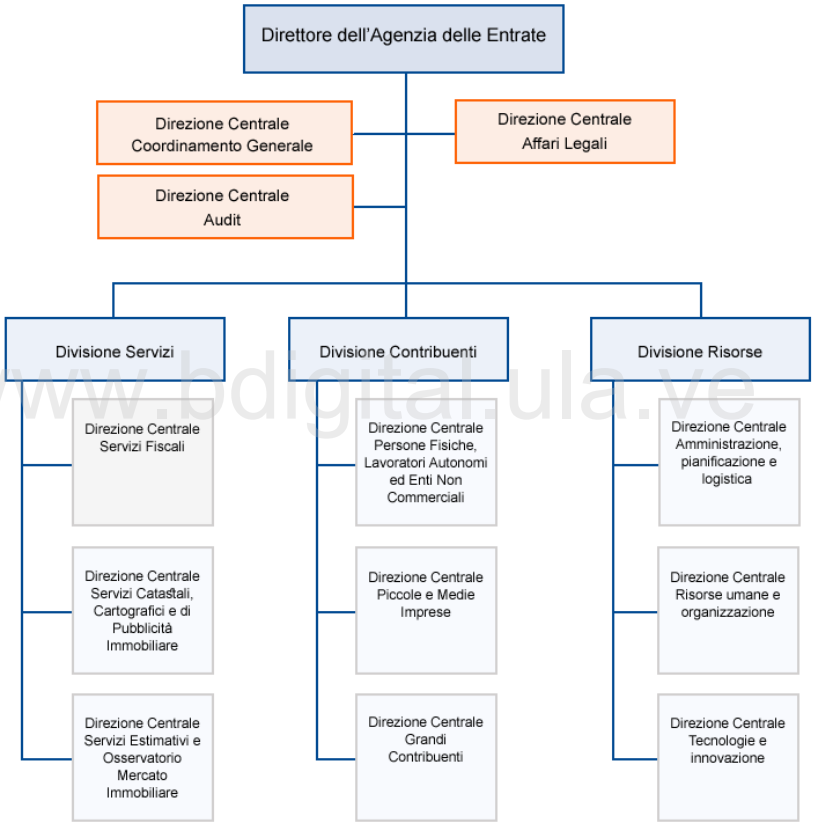


Figura 16: Organigrama Central y Articulaciones de las oficinas Agenzia delle Entrate, tomado de Internet, https://www.agenziaentrate.gov.it/portale/web/guest/agenzia/chi-siamo/organigramma-centrale

En su página web se pueden encontrar los diferentes servicios que este ente proporciona a nivel de gestiones fiscales, pero para nuestros propósitos, *L'Agenzia delle Entrate* es el ente que realiza la administración tributaria y por tanto quien se encarga de realizar todos los procesos fiscales requeridos en torno a la facturación electrónica en Italia.

3.3.2 Legge di Bilancio 2018 (Ley del Banlance 2018)

La *Legge di Bilancio*, es el instrumento previsto en el artículo 81 de la Constitución italiana a través del cual el Gobierno, con un documento contable preventivo, comunica al Parlamento el gasto público y los ingresos esperados para el año siguiente sobre la base de las leyes vigentes (a diferencia de la declaración final, que es un documento contable que enumera los ingresos y gastos que ocurrieron en el ejercicio al que se refiere el presupuesto).

Basado en el mencionado art. 81, la ley de aprobación del presupuesto no puede, a diferencia de la ley de estabilidad, introducir nuevos impuestos y nuevos gastos. Cualquier otra regla que introduzca nuevos gastos debe indicar su respectiva cobertura financiera. Según este artículo, el Presidente de la República puede negarse a firmar leyes sin cobertura financiera.

En particular la *Legge di Bilancio, (2018)* en su Artículo 1 – 411 decreta explícitamente que a partir del 1 de Enero del 2019:

> "Con el fin de fomentar la eficiencia y la transparencia del sistema de suministro de la administración pública, la emisión, transmisión y almacenamiento y la presentación de los documentos necesarios para el pedido y la ejecución de las compras de bienes y servicios debe realizarse electrónicamente. Para ello, sin perjuicio las disposiciones de los párrafos 412, 413 y 414, con decretos del Ministro de Economía y Finanzas, previa consulta a la Agencia para Italia digital (AGID), de acuerdo con la Conferencia se han adoptado reglamentos unificados y específicos para regular los métodos técnicos y las fechas de entrada en vigor de los métodos de presentación electrónica obligatoria de la documentación antes mencionada."

Entrando en vigencia la obligatoriedad de la realización para todos los tramites comerciales en Italia de la facturación electrónica, reemplazado en esta modalidad a las facturas físicas, especificando, en el mismo artículo en los puntos 909,920,995,1011,1012,1026,1028,1033 y 1123 todas las normas que rigen a las mismas.

L'Agenzia Delle Entrate también dicta que "La factura electrónica original será el fichero XML, como siempre se confirmado. Existe el derecho por parte del remitente de la factura electrónica de enviar también un PDF adjunto que, sin embargo, no tiene ningún valor, especialmente en caso de diferencias de contenido (no de forma) entre XML y PDF" resaltando que se optó por usar el XML como el fichero de datos aceptado por el ente fiscal.

Entre los plazos establecidos en esta ley, se establecen dos obligaciones más con diferentes plazos de ejecución:

- A partir del 1 de julio de 2018, la obligación de utilizar la facturación electrónica afecta a dos nuevos colectivos:

    1. La obligación se extiende no sólo a los servicios prestados por los contratistas a la Administración Pública, sino también a los prestados por los subcontratistas a la empresa principal en el marco de un contrato de licitación estipulado con la Administración Pública. Se excluyen los niveles inferiores; esto significa que, si los subcontratistas también utilizan, a su vez, bienes o servicios suministrados por otra entidad, esta es libre de emitir facturas de conformidad con los usos comunes.
    2. Igualmente, es obligatorio para las transacciones de venta de combustibles usados como carburantes para motores.

- Desde el 1 de enero del 2019, la obligación del uso de la factura electrónica tendrá carácter general, se extenderá a todas las transacciones entre todos los particulares registrados a efectos de IVA, siempre que las entregas de bienes y las prestaciones de servicios se efectúen entre personas y/o empresas residentes o establecidas en el territorio del Estado Italiano.

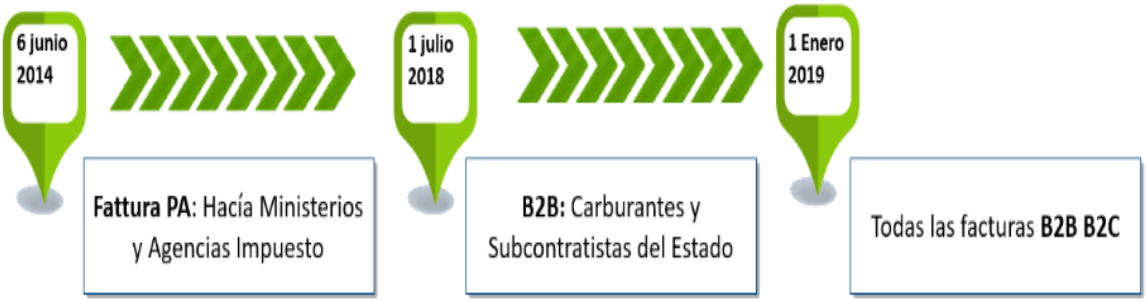


Figura 17: Fechas de lanzamiento de la factura electrónica, tomado de Internet, https://www.techedgegroup.com/es/blog/factura-electronica-sera-obligatoria-italia-a-partir-2019

Los pasos que se deben realizar para generar una factura electrónica fiscalmente válida y certificada por *L'Agenzia Delle Entrate* son los siguientes:

1. El emisor crea la factura a través de su sistema de contabilización.
2. La factura tendrá que ser transformada en formato XML y firmada con el certificado del representante legal de la empresa antes de ser enviada.
3. El sistema SDI recibe la factura y la envía, según su dirección PEC (correo electrónico certificado) o código fiscal, al receptor.
4. El receptor recibe la factura electrónica.
5. Una vez aceptada o rechazada la factura, se enviará una respuesta al SdI.
6. El SdI recibe la respuesta y la envía al emisor.
7. El emisor recibe el mensaje de respuesta del emisor y cambia el estado de su factura.

Entre las sanciones dictadas por *L'Agenzia Delle Entrate* cabe enumerar los siguientes puntos dispuestos en dicha ley:

1. A partir del 1 de enero, toda factura emitida en papel se considerará inexistente o nula.
2. En caso de incumplimiento de la obligación de facturación electrónica por parte de los operadores privados, hay que tener en cuenta que la posible emisión de la factura en papel se considerará inexistente y el documento como no emitido.
3. En caso de incumplimiento, está prevista una sanción administrativa de entre el noventa y el ciento ochenta por ciento del impuesto correspondiente a la base imponible no debidamente documentada o registrada durante el ejercicio.

Con respecto a la fecha de emisión y recepción del documento fiscal, *L'Agenzia delle Entrate* dicta que la fecha de emisión es la fecha del documento que debe figurar en la factura. También se confirmó que si el SDI (*Sistema di Interscambio*) descarta la factura electrónica, debería considerarse que no ha sido emitida, con la consecuencia de que el proveedor tendría que emitir una nota interna de cambio si ya hubiera registrado la factura.

Para el tratamiento de las facturas electrónicas, definiendo el direccionamiento de una factura electrónica como el acto que permite al SDI entregar el documento al destinatario, por tanto, debe preverse la inserción del código del destinatario, el código de identificación del SDI.

Como alternativa, es posible utilizar el Correo Electrónico Certificado (PEC) del destinatario, además también será posible "reportar" el método de entrega estándar preferido, que prevalecerá sobre el contenido mostrado en la factura, poniendo a disposición, un servicio en la página web de *L'Agenzia Delle Entrate* que permite realizar esta elección, permitiendo combinar la indicación del código de destino (SDI ID) o un PEC con el número de IVA.

En resumen, las facturas electrónicas emitidas por las empresas serán recibidas por los sistemas centrales de *L'Agenzia Delle Entrate* para ser después enviadas a los correspondientes destinatarios. Es el ente tributario quien (mediante esta ley decretada) establece las normas prácticas para la emisión y la recepción de la factura electrónica entre empresas, proceso que puede ser interpretado más fácilmente a partir de los siguientes esquemas.

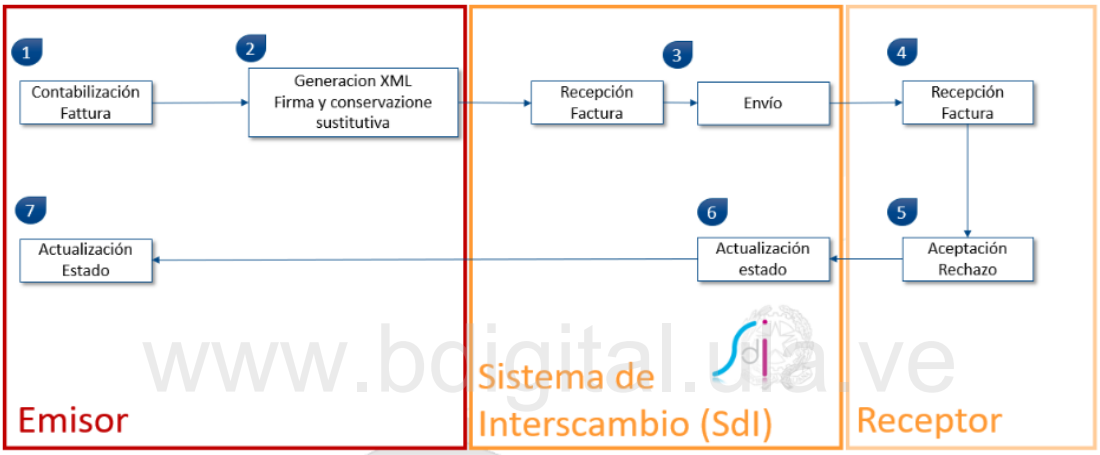


Figura 18: El Flujo de la Factura, tomado de Internet,
https://www.techedgegroup.com/es/blog/factura-electronica-sera-obligatoria-italia-a-partir-2019

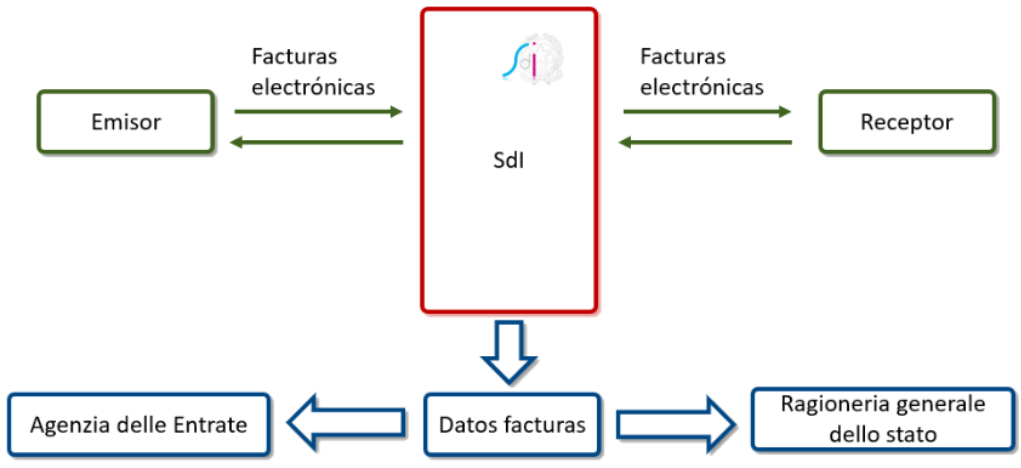


Figura 19: Distribución de la factura electrónica - Sistema di Interscambio (SdI), tomado de Internet,
https://www.techedgegroup.com/es/blog/factura-electronica-sera-obligatoria-italia-a-partir-2019

3.3.3 Información adjunta relevante a la facturación electrónica en Italia

L'Agenzia Delle Entrate, en su portal web, cuenta con una amplia variedad de información y normativas referentes la facturación electrónica, entre los que se menciona el proceso de la firma, el cual para las normas establecidas por el ente no es obligatoria de realizar, pero es opcional, algunos sistemas como el presentado en este proyecto cuenta con su proceso de firma ya que así proporciona un aval más de seguridad en el proceso de certificación de la factura digital.

Dado que describir todos estos elementos como pueden ser guías prácticas, requerimientos técnicos del documento, programas propios que utiliza el ente, por ejemplo el SDI (*Sistema di Interscambio*) y otras herramientas de índole fiscal, sería un proceso demasiado largo y que no entra en la parte principal del desarrollo de este proyecto, el objetivo de esta sección es dejar adjunto todos los link que llevan al sitio de *L'Agenzia Delle Entrate* que contienen información relevante a nivel fiscal y de sus procesos internos relacionados con la facturación electrónica.

- Guía a la facturación electrónica:
  https://www.agenziaentrate.gov.it/portale/web/guest/aree-tematiche/fatturazione-elettronica/guida-fatturazione-elettronica

- Manual de especificaciones Técnicas:
  https://www.agenziaentrate.gov.it/portale/documents/20143/288192/Allegato+A+-+Specifiche+tecniche+vers+1.5.pdf/7b366fd0-d76d-047d-4921-218e150b7234

- Facturación electrónica y Datos para facturas de extranjeros (Fatturazione elettronica e dati fatture transfrontaliere – Provvedimento del 30 aprile 2018):
  https://www.agenziaentrate.gov.it/portale/web/guest/schede/comunicazioni/fatture-e-corrispettivi/fatture-e-corrispettivi-st/st-invio-di-fatturazione-elettronica

- Servicio gratuito para predisponer, enviar, conservar y consultar las facturas electrónicas:
  https://www.agenziaentrate.gov.it/portale/web/guest/aree-tematiche/fatturazione-elettronica/fatturazione-elettronica-site-area/servizi-consultazione-e-conservaz-fatture-elettroniche

- Área de preguntas frecuentes:
  https://www.agenziaentrate.gov.it/portale/web/guest/schede/comunicazioni/fatture-e-corrispettivi/faq-fe

CAPITULO 4

Análisis de los requerimientos.

El análisis del sistema se apoya en la ingeniería de requisitos, la cual ayuda a entender, desde un punto de vista razonable, la mejor solución que se propone para el problema abordado y es el mecanismo más efectivo para descubrir y desarrollar el producto que satisface las necesidades o requerimientos del cliente.

Los requisitos de un sistema se pueden definir como la especificación técnica de lo que el sistema debe hacer y sus propiedades esenciales, en otras palabras, son la lista de servicios y delimitaciones operativas que proporcionara el sistema para satisfacer con las necesidades de sus clientes.

Para realizar el estudio de estos requisitos del sistema, normalmente se procede a consultar con los clientes y usuarios finales del sistema, pero en el desarrollo de este proyecto en particular, que es un software privado destinado a ofrecer el proceso de la facturación electrónica en Italia, los requerimientos serán consultados de una forma diversa a la común por dos motivos importantes a destacar.

El primer motivo es que, para la realización de una factura electrónica, que se pueda considerar como un documento fiscalmente valido, el proceso de creación, formatación y envío se apega a ciertas normas y leyes fiscales como pudimos apreciar en el capítulo anterior, lo cual permite a través de la investigación realizada definir, diseñar y delimitar ciertos requisitos mínimos a nivel de la creación de una factura.

El segundo motivo es que al ser un software privado, es su propietario quien define la flexibilidad a nivel de servicios adjuntos que desea poner a disposición de sus clientes, siendo Datalog Italia Srl una empresa con más de 19 años en desarrollo de software administrativo y los propietarios del desarrollo planteado, cuentan con personal especializado en el área de procedimientos fiscales quienes actúan como el cliente y están encargados de proporcionar la lista de funcionalidades necesarias a desarrollar dentro del sistema, así como también de los servicios disponibles desarrollados anteriormente y que pueden ser reutilizados dentro del área de la facturación (servicios que pertenecen a otros sistemas como lo son KING, Datalog HUB y el subsistema de registro de datos, planificación y configuración de DWEB. Este personal además está capacitado para proporcionar el asesoramiento pertinente en materia fiscal que sea necesario para el desarrollo del proyecto.

Usualmente, los requisitos se derivan en dos tipos: funcionales y no funcionales. Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que el mismo debe reaccionar a entradas particulares y de cuál debe ser el comportamiento de este en situaciones particulares. Por otra parte, los requisitos no funcionales son restricciones de los servicios ofrecidos por el sistema, incluyendo restricciones de tiempo, sobre el proceso de desarrollo y estándares.

Para el desarrollo del subsistema propuesto se realizó una lista de los requerimientos mínimos proporcionados por la empresa Datalog Italia Srl, la lista se presenta más adelante, dividida en dos secciones según el tipo (funcional o no funcional).

4.1 Estructura del sistema DWEB

Antes de partir con la definición de los requisitos cabe destacar que el desarrollo del subsistema propuesto forma parte de un entorno más grande que en conjunto es llamado Datalog Web (DWEB), para poner en contexto al lector, DWEB es el sistema encargado de toda la gestión a nivel de registro de datos en el sistema y de envío de las facturas electrónicas al SDI de la *Agenzia Delle Entrate*.

Se puede decir que el sistema DWEB es la integración de un sistema que nace a partir de dos subsistemas propuestos, cada uno de estos subsistemas se encargan de realizar funciones específicas para la gestión de datos y creación de las facturas, el desarrollo de nuestro subsistema está destinado a formar parte de una integración con un subsistema adicional, para la creación del sistema DWEB.

Nuestro subsistema depende en un alto nivel del subsistema de registro de datos, planificación y configuración, puesto que es el encargado del registro y la carga de datos en comunicación con el sistema KING, mientras por su parte, nuestro subsistema es el usado para realizar los procesos de firma y envío al SDI apoyándose en el sistema Datalog HUB.

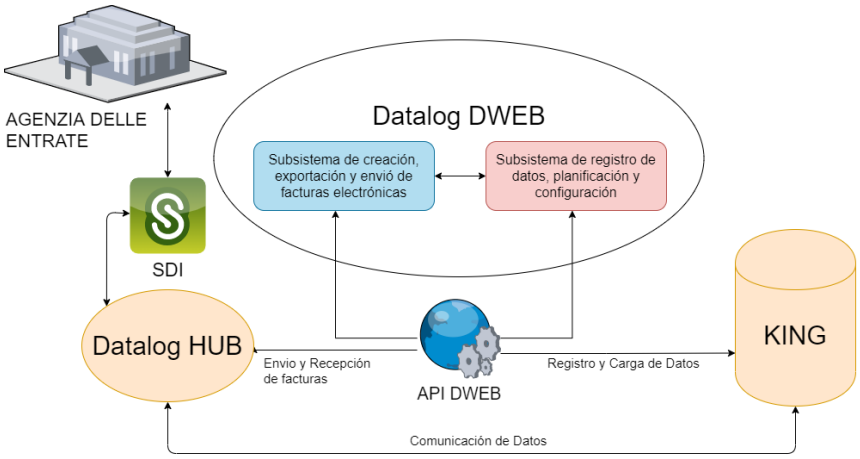


Figura 20: Estructura del sistema DWEB

4.2 Requisitos de Software Funcionales

Requisito	Nombre	Tipo
RS-01	Carga de datos iniciales de la factura	Funcional
RS-02	Guardar borrador de la factura en automático	Funcional
RS-03	Cargar factura a partir de un borrador	Funcional
RS-04	Agregar y cambiar datos a la cabecera de la factura	Funcional
RS-05	Recargar campos seleccionables	Funcional
RS-06	Agregar datos del cliente a la factura	Funcional
RS-07	Agregar datos de pagamento a la factura	Funcional
RS-08	Agregar gastos adicionales a la factura	Funcional
RS-09	Realizar la autofactura	Funcional
RS-10	Agregar línea de producto o nota a la factura	Funcional
RS-11	Editar una línea de producto o nota en la factura	Funcional
RS-12	Eliminar un producto o nota de la factura	Funcional
RS-13	Cambiar de posición una línea de la factura	Funcional
RS-14	Actualización de cálculos de la factura	Funcional
RS-15	Personalizar las cuotas de pago de la factura	Funcional
RS-16	Creación de nuevos clientes, bancos o productos	Funcional
RS-17	Culminar el proceso de crear la factura	Funcional
RS-18	Modificar una factura ya creada	Funcional
RS-19	Crear un témplate de una factura	Funcional
RS-20	Listar las 10 últimas facturas enviadas al DHUB	Funcional
RS-21	Lista las facturas por enviar al DHUB	Funcional
RS-22	Documentos adjuntos al envío de la factura	Funcional
RS-23	Enviar una factura al DHUB	Funcional
RS-24	Visualizar PDF, XML y Assosoftware de la factura	Funcional
RS-25	Eliminar una factura	Funcional
RS-26	Listar y filtrar resumen de facturas de una empresa	Funcional
RS-27	Exportar facturas singulares o Zip	Funcional
RS-28	Exportar Excel con la lista resumen de facturas	Funcional

Tabla 3 Lista de Requerimientos Funcionales

Nombre: Carga de datos iniciales de la factura	**Tipo**: Funcional	**Requisito**: RS-01
Descripción: El sistema debe proporcionar una carga de datos inicialmente a la creación de la factura como son el último número de documento, el tipo de documento, los seccionales, combos seleccionables, campos específicos a llenar según la tipología y los datos de la empresa que se utilizan en la cabecera de la factura.		
Justificación: El usuario final mediante el registro de la empresa y posterior configuración con el subsistema de registro de datos, planificación y configuración predispone los datos de la cabecera de la factura y los valores por defecto para realizar la factura de forma veloz (adaptándose a la tipología). Además, deben realizarse diversas verificaciones con los rangos de fecha disponible para la factura (teniendo en cuenta número y fecha de la última factura según el seccional elegido) y otros valores como son: IVA por defecto, *rivalsa*, causas de pagos, tipos de pagos por defecto, *cassa tratenuta*, condición de pagamento, entre otros. Se verifican también algunas configuraciones sobre el envío de la factura.		

Tabla 3.1 Requisito RS-01. Carga de datos iniciales de la factura

Para obtener una idea clara de todo el proceso de carga de datos que debe ofrecer el sistema al inicio de la creación de la factura, se muestra en la siguiente figura el área de configuración de la empresa que registra cada usuario en el sistema, los campos que se aprecian en esta vista son todos aquellos que vienen precargados cada vez que se inicia una nueva factura. Es importante destacar que pertenecientes al otro subsistema que será integrado al entorno DWEB (subsistema de registro de datos, planificación y configuración).

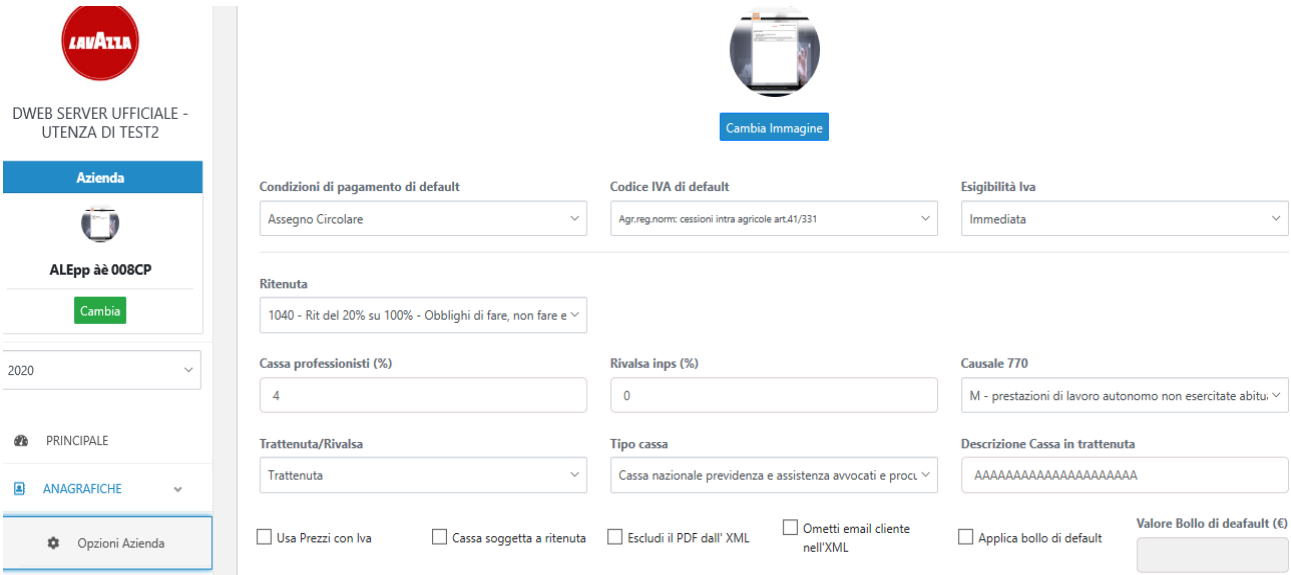


Figura 21: Área de configuración de la empresa

Nombre: Guardar borrador de la factura en automático	**Tipo**: Funcional	**Requisito**: RS-02
Descripción: El sistema debe guardar un borrador (*bozza*) en automático cada vez que se realizan cambios en la factura.		
Justificación: El usuario debe poder registrar el ultimo estado disponible de la fase de creación de la factura en forma automática, de forma que si ocurre una interrupción inesperada como pueden ser: vencimiento de la sesión, falta de conexión a internet, cierre accidental del navegador reinicio del dispositivo, perdida de energía en el dispositivo o cualquier otro motivo. Al tener disponible un borrador de la factura, esto le permite retomar su trabajo desde el último punto guardado (de gran utilidad para empresas que realizan facturas con muchas líneas de productos o notas).		

Tabla 3.2 Requisito RS-02. Guardar borrador de la factura en automático

Nombre: Cargar factura a partir de un borrador	**Tipo**: Funcional	**Requisito**: RS-03
Descripción: El sistema debe poder cargar el estado de una factura a partir del borrador almacenado.		
Justificación: El usuario debe poder recuperar el ultimo estado disponible guardado de la fase de creación de la factura, al buscar el borrador debe poder cargar todos los datos de la factura de dicho borrador, para posteriormente, poder continuar con la realización del documento.		

Tabla 3.3 Requisito RS-03. Cargar factura a partir de un borrador

Nombre: Agregar y Cambiar datos a la cabecera de la factura	**Tipo**: Funcional	**Requisito**: RS-04
Descripción: El sistema debe permitir cambiar ciertos datos de la cabecera de la factura.		
Justificación: El usuario debe poder cambiar los datos precargados en la creación de la factura, como la fecha del documento, el seccional, y los campos seleccionables que vienen cargado por defecto según la configuración de la empresa.		

Tabla 3.4 Requisito RS-04. Agregar y Cambiar datos a la cabecera de la factura

Nombre: Recargar campos seleccionables	**Tipo**: Funcional	**Requisito**: RS-05
Descripción: El sistema debe en automático actualizar los campos seleccionables.		
Justificación: El usuario puede cambiar algunos campos, como la tipología o el seccional de la factura, que son críticos e influyen sobre otros campos dependientes de estos, el sistema antes de guardar el borrador, debe también actualizar las opciones disponibles de estos campos dependientes.		

Tabla 3.5 Requisito RS-05. Recargar campos seleccionables

Nombre: Agregar datos del cliente a la factura	**Tipo**: Funcional	**Requisito**: RS-06
Descripción: El sistema debe poder cargar los datos de un cliente a la factura.		
Justificación: El usuario puede seleccionar un cliente del registro de clientes que tiene en su empresa para ser agregado a la factura, estos datos son los que se agregan a la cabecera de la factura para ser el receptor de la factura, además de cargar datos básicos fiscales, se cargan datos como Iva por defecto o la modalidad de pagamento predefina para el cliente en particular, todos estos datos son agregados mediante el otro subsistema que integra al DWEB.		

Tabla 3.6 Requisito RS-06. Agregar datos del cliente a la factura

Nombre: Agregar datos de pagamento a la factura	**Tipo**: Funcional	**Requisito**: RS-07
Descripción: El sistema debe poder cargar y registrar los datos del pagamento.		
Justificación: El usuario puede seleccionar una banca del registro (proporcionados por el subsistema de registro) para cargarlo a la factura o puede usar la función de verificación de Iban para autocompletar los datos bancarios, adicionalmente se puede cambiar la modalidad de pagamento, lo cual proporciona el plan de cuotas de pago de la factura en automático (proporcionado por KING mediante el subsistema de registro).		

Tabla 3.7 Requisito RS-07. Agregar datos de pagamento a la factura

Nombre: Agregar gastos adicionales a la factura	**Tipo**: Funcional	**Requisito**: RS-08
Descripción: El sistema debe poder agregar gastos adicionales a la factura.		
Justificación: El usuario puede agregar gastos adicionales definidos a la factura, estos gastos son independientes a los demás elementos de la factura, son gastos adjuntos como: gastos de transporte, gastos bancarios y gastos de sellos (estos son los únicos que tienen una verificación sobre el imponible de la factura). Estos gastos no interfieren con el cálculo de la factura pero van especificados en el Total y en una sección específica en el PDF de la factura creada.		

Tabla 3.8 Requisito RS-08. Agregar gastos adicionales a la factura

Nombre: Realizar la autofactura	**Tipo**: Funcional	**Requisito**: RS-09
Descripción: El sistema debe permitirle a un cliente la realización de la autofactura.		
Justificación: El usuario final, bajos ciertas circunstancias específicas, debe poder realizarse una autofactura, esto ocurre según ciertas normas fiscales al momento de la declaración y le permite a una empresa crear una autofactura, cuyo proceso es el de invertir los datos de emisor y el remitente de la factura, usando los datos del cliente como emisor y los de la empresa como receptor.		

Tabla 3.9 Requisito RS-09. Realizar la autofactura

Nombre: Agregar línea de producto o nota a la factura	**Tipo**: Funcional	**Requisito**: RS-10
Descripción: El sistema debe permitir agregar nuevos productos o notas al detalle de la factura.		
Justificación: El usuario, una vez ingresado el cliente (ya que por motivos fiscales puede alterar campos de la línea de detalle del producto según el tipo de cliente), debe poder agregar nuevos artículos al detalle de la factura o una línea de nota, para este proceso además se deben realizar diversas verificaciones en los campos insertados.		

Tabla 3.10 Requisito RS-10. Agregar línea de producto o nota a la factura

Nombre: Editar una línea de producto o nota en la factura	**Tipo**: Funcional	**Requisito**: RS-11
Descripción: El sistema debe permitir la edición de un producto o nota ya insertado en la factura.		
Justificación: El usuario final, debe poder editar los campos de un producto o la nota que ya ha sido insertada a la factura, para corregir posibles errores o agregar más detalle.		

Tabla 3.11 Requisito RS-11. Editar una línea de producto o nota en la factura

Nombre: Eliminar un producto o nota de la factura	**Tipo**: Funcional	**Requisito**: RS-12
Descripción: El sistema debe permitir la eliminación de un producto o nota de la factura.		
Justificación: El usuario debe poder realizar la eliminación de una línea de detalle de la factura, es decir debe poder eliminar una nota o producto de la misma.		

Tabla 3.12 Requisito RS-12. Eliminar un producto o nota de la factura

Nombre: Cambiar de posición una línea de la factura	**Tipo**: Funcional	**Requisito**: RS-13
Descripción: El sistema debe permitirle mover un producto o nota de la factura.		
Justificación: El usuario final, debe poder ordenar la lista de artículos o notas, después de haberlas agregado en la factura, esto le evita tener que rehacer la factura si debe agregar alguna nota después de un producto o si decide reorganizar los productos de la factura.		

Tabla 3.13 Requisito RS-13. Cambiar de posición una línea de la factura

Nombre: Actualización de cálculos de la factura	**Tipo**: Funcional	**Requisito**: RS-14
Descripción: El sistema debe en automático actualizar los cálculos de la factura.		
Justificación: En el proceso de creación de la factura pueden ser realizados los cálculos, además mientras realiza cambios, pueden cambiar ciertos elementos puntuales sobre la factura como el área de totalización o el cálculo de las cuotas de pago, para realizar todos estos cálculos se apoya en las funciones de guardado y cargado del borrador de la factura.		

Tabla 3.14 Requisito RS-14. Actualización de cálculos de la factura

Nombre: Personalizar las cuotas de pago de la factura	**Tipo**: Funcional	**Requisito**: RS-15
Descripción: El sistema debe permitirle a un cliente cambiar las cuotas de pago de la factura.		
Justificación: El sistema, mientras el usuario crea la factura, realiza diversos cálculos automáticos sobre la factura, entre estos cálculos (según la modalidad de pagamento) se genera una programación de cuotas de pago de la factura, estas cuotas deben poder ser personalizables en la factura y debe una vez agregada o eliminada una cuota, recalcular los montos de las cuotas de pago.		

Tabla 3.15 Requisito RS-15. Personalizar las cuotas de pago de la factura

Nombre: Creación de nuevos clientes, bancos o productos	**Tipo**: Funcional	**Requisito**: RS-16
Descripción: El sistema debe permitir agregar nuevos clientes, bancos o productos.		
Justificación: El usuario final, regularmente añade los clientes, el banco y los productos, a partir de la data existente en el subsistema de registro, el sistema debe permitir (reutilizando las funciones de este sistema) agregar nuevos clientes, bancos o productos a la factura, si estos no se encuentra aún en el registro (siendo el producto el único que no se guarda en el subsistema).		

Tabla 3.16 Requisito RS-16. Creación de nuevos clientes, bancos o productos

Nombre: Culminar el proceso de crear la factura	**Tipo**: Funcional	**Requisito**: RS-17
Descripción: El sistema debe poder culminar la creación de la factura.		
Justificación: El sistema debe poder permitir culminar el proceso de creación de una factura, para esto se realizan diversas validaciones propias en el formulario y se procede a eliminar el borrador de la factura, de esta forma se genera una factura consultable en sus exportaciones y lista para el proceso de envío al sistema DHUB.		

Tabla 3.17 Requisito RS-17. Culminar el proceso de crear la factura

Nombre: Modificar una factura ya creada	**Tipo**: Funcional	**Requisito**: RS-18
Descripción: El sistema debe permitirle a un cliente modificar una factura.		
Justificación: El usuario final, puede por dos motivos modificar una factura, el primero es si ha cometido algún error y desea modificarla (siempre que esta no haya sido enviada al DHUB), el segundo es si la factura ha sido previamente enviada al DHUB y posteriormente descartada por la *Agenzia Delle Entrate* (quiere decir que existe un error en la factura, por lo tanto, debe de ser modificada y reenviada), para esto se procede a la modificación creando un borrador de la factura existente (ver RS-03), realizando los cambios sobre este borrador y posteriormente se reescribe la factura existente (ver RS-17).		

Tabla 3.18 Requisito RS-18. Modificar una factura ya creada

Nombre: Crear un témplate de una factura	**Tipo**: Funcional	**Requisito**: RS-19
Descripción: El sistema debe permitirle a un cliente la creación de templates.		
Justificación: El sistema debe permitir la creación de los templates de factura, esto no es más que realizar un guardado diverso del borrador de una factura, esto permite que el mismo no se elimine automáticamente posteriormente a la creación y es aplicable en los casos donde el cliente debe realizar facturación de servicios o despachos de productos recurrentes, una vez necesite enviar la próxima factura solo debe de cargar el témplate (borrador) que contiene todos los datos de la factura.		

Tabla 3.19 Requisito RS-19. Crear un témplate de una factura

Nombre: Listar las 10 últimas facturas enviadas al DHUB	**Tipo**: Funcional	**Requisito**: RS-20
Descripción: El sistema debe mostrar la lista de las ultimas facturas enviadas al DHUB.		
Justificación: El usuario final, debe poder inmediatamente al realizar el login y seleccionar su empresa (página *home azienda*), ver el estado de las ultimas 10 facturas enviadas al DHUB, de esta forma puede ver inmediatamente el estado actual de las facturas enviadas por el DHUB al SDI, esta visualización tiene botones de exportación rápida.		

Tabla 3.20 Requisito RS-20. Listar las 10 últimas facturas enviadas al DHUB

Nombre: Lista las facturas por enviar al DHUB	**Tipo**: Funcional	**Requisito**: RS-21
Descripción: El sistema debe mostrar la lista de las facturas creadas que no han sido enviadas.		
Justificación: El sistema, debe mostrarle al usuario (en la página *home azienda*), las facturas que han sido creadas en DWEB y que a su vez están pendientes de enviar al HUB, en esta lista se añade el botón modificación y de envío rápido.		

Tabla 3.21 Requisito RS-21. Lista las facturas por enviar al DHUB

Nombre: Documentos adjuntos al envío de la factura	**Tipo**: Funcional	**Requisito**: RS-22
Descripción: El sistema debe permitirle adjuntar documentos a la factura.		
Justificación: El usuario, una vez culminado el proceso de creación de la factura, será redirigido a la vista de pre-visualización del pdf, en este momento, podrá añadir documentos adjuntos a la factura para su posterior envío al sistema DHUB, esta función es opcional, pero necesaria para la realización de facturas que requieren de documentos adjuntos de soporte, los cuales deberán ser enviados con la factura para evitar su rechazo por la *Agenzia Delle Entrate*.		

Tabla 3.22 Requisito RS-22. Documentos adjuntos al envío de la factura

Nombre: Enviar una factura al DHUB	**Tipo**: Funcional	**Requisito**: RS-23
Descripción: El sistema debe poder realizar el envío de facturas al sistema DHUB.		
Justificación: El usuario final, una vez considera la factura terminada, debe poder enviar la factura al sistema DHUB que se encarga de continuar con el proceso fiscal de firma y envío de facturas al SDI.		

Tabla 3.23 Requisito RS-23. Enviar una factura al DHUB

Nombre: Visualizar PDF, XML y Assosoftware de la factura	**Tipo**: Funcional	**Requisito**: RS-24
Descripción: El sistema debe permitir la pre-visualización de la factura en diversos formatos.		
Justificación: El sistema, una vez creada la factura debe poder permitirle al usuario la visualización en los diversos formatos predispuestos por KING y las DLL de exportación ya disponibles, estos formatos son creados con diversos fines para el emisor, el receptor y el sistema destino de la factura (SDI).		

Tabla 3.24 Requisito RS-24. Visualizar PDF, XML y Assosoftware de la factura

Nombre: Eliminar una factura	**Tipo**: Funcional	**Requisito**: RS-25
Descripción: El sistema debe permitir la eliminación de factura.		
Justificación: El sistema, verificando la condición especifica del envío de la factura, debe poder permitir eliminar una o más facturas ya creadas, la condición es netamente fiscal, puesto que si una factura ha sido enviada al DHUB y por tanto al SDI la factura queda registrada (aun si es rechazada), por lo tanto las facturas podrán ser eliminadas solo hasta la última creada que ha sido enviada (sin incluirla).		

Tabla 3.25 Requisito RS-25. Eliminar una factura

Nombre: Listar y filtrar resumen de facturas de una empresa	**Tipo**: Funcional	**Requisito**: RS-26
Descripción: El sistema debe presentar un resumen de la facturas de una empresa.		
Justificación: El usuario final, debe tener un área que le permita listar y filtrar las facturas creadas y enviadas por su empresa, esta áreas debe contar además con botones de acción rápida y características que permitan operar rápidamente sobre el elenco de las facturas de su empresa.		

Tabla 3.26 Requisito RS-26. Listar y filtrar resumen de facturas de una empresa

Nombre: Exportar facturas singulares o Zip	**Tipo**: Funcional	**Requisito**: RS-27
Descripción: El sistema debe permitir la exportación de una o más facturas.		
Justificación: El usuario final, debe poder realizar la exportación en diversos formatos para su descarga (ver RS-24), esta funcionalidad será disponible en diversas partes del sistema pero principalmente en el área de resumen de facturas, donde adicionalmente podrá descargar un Zip con más de una factura.		

Tabla 3.27 Requisito RS-27. Exportar facturas singulares o Zip

Nombre: Exportar Excel con la lista de facturas	**Tipo**: Funcional	**Requisito**: RS-28
Descripción: El sistema debe realizar una exportación Excel con una lista de facturas.		
Justificación: El usuario final, podrá descargar un listado Excel con el detalle relevante a diversas facturas, así como aplicar los filtros presentes en la vista mediante los cuales podrá generar rápidamente (usando las combinaciones de filtros correcta) reportes de balances para los periodos seleccionados, siendo de gran utilidad para los *Commercialistas* (profesionales que se ocupan de los reportes económicos o comerciales del punto de vista organizativo, financiero, tributario o jurídico), que suelen gestionar varias empresas en el sistema.		

Tabla 3.28 Requisito RS-28. Exportar Excel con la lista de facturas

4.3 Requisitos de Software No Funcionales

Requisito	Nombre	Tipo
RS-29	Interfaz gráfica de usuario responsive	No Funcional
RS-30	Resalto de campos (obligatorios o con errores)	No Funcional
RS-31	Validación automática de algunos campos	No Funcional
RS-32	Visibilidad de secciones en la creación factura	No Funcional
RS-33	Diversidad en visualización de tablas y elementos	No Funcional
RS-34	Modal para agregar línea de detalle en vista móvil	No Funcional

Tabla 4 Lista de Requerimientos No Funcionales

Nombre: Interfaz gráfica de usuario responsive	**Tipo**: No Funcional	**Requisito**: RS-29
Descripción: El sistema debe presentar una interfaz de usuario responsive.		
Justificación: Los avances tecnológicos ponen a disposición de los usuarios una multitud de dispositivos móviles, cabe destacar que su uso tiene un crecimiento exponencial, esto incluye los entornos empresariales, donde es común encontrar cada vez más estos dispositivos. Nuestro sistema al ser un programa de facturación digital debe presentar una interfaz que permita abarcar la mayoría de los dispositivos para tener un mayor alcance del producto a los usuarios a los que se destina. Además esto permite la realización de la factura electrónica en cualquier momento y lugar con cualquier dispositivo que pueda conectarse a internet, otorgando miles de posibilidades y soluciones a los usuarios del sistema.		

Tabla 4.1 Requisito RS-29. Interfaz gráfica de usuario responsive

Nombre: Resalto de campos (obligatorios o con errores)	**Tipo**: No Funcional	**Requisito**: RS-30
Descripción: El sistema debe alertar al usuario si se encuentra un error o no puede realizar una acción.		
Justificación: El sistema, a través de los procesos de creación de la factura electrónica, debe ser capaz de advertirle al usuario las diversas respuestas a las acción que intenta realizar que están incompletas o presentan un error, como por ejemplo insertar un producto a la factura sin antes haber insertado el cliente o intentar agregar un producto a la factura sin insertar su precio o cantidad, de otra forma el usuario final no podría ser consciente de estos error y pensar que sea un mal funcionamiento del sistema.		

Tabla 4.2 Requisito RS-30. Resalto de campos (obligatorios o con errores)

Nombre: Validación automática de algunos campos	**Tipo**: No Funcional	**Requisito**: RS-31
Descripción: El sistema debe automáticamente realizar ciertas validaciones sobre algunos campos.		
Justificación: Si bien este requisito podría considerarse funcional, no lo es puesto que el objetivo es, que el sistema, realice diversas validaciones sobre campos específicos con la función de realizar alertas al usuario por si tiene un error de escritura (pero sin interferir en el proceso), un ejemplo de estos campos son los de email, *partita iva* o *codice fiscale* (número de registro fiscal italiano que representa a la persona natural o jurídica) que poseen formatos y longitudes específicos. Esta validación se realiza sobre el campo en el momento del perdida del focus, se procede a señalar el campo (ver RS-30).		

Tabla 4.3 Requisito RS-31. Validación automática de algunos campos

Nombre: Visibilidad de secciones en la creación factura	**Tipo**: No Funcional	**Requisito**: RS-32
Descripción: El sistema debe presentar o esconder algunas secciones de la factura.		
Justificación: Es importante destacar que el diseño propuesto para la realización de la factura consta de diversas secciones que se ven en una sola página, dependiendo de las diversas acciones o elecciones que toma el usuario final (principalmente la selección de la tipología), el sistema debe presentar o esconder algunas secciones que son propias a cada tipología de factura.		

Tabla 4.4 Requisito RS-32. Visibilidad de secciones en la creación factura

Nombre: Diversidad en visualización de tablas y elementos	**Tipo**: No Funcional	**Requisito**: RS-33
Descripción: El sistema debe presentar una diversidad en algunos elementos según el dispositivo.		
Justificación: Aunque el requerimiento podría pertenecer al diseño responsive (ver RS-29), se distingue en la particularidad de que, en los dispositivos móviles algunas de las acciones rápidas (visualización o exportación de los formatos Zip, XML o Assosoftware y el Drag and Drop de líneas) no son posibles de realizar o son limitadas por el dispositivo, motivo por el cual este es un requerimiento especifico.		

Tabla 4.5 Requisito RS-33. Diversidad en visualización de tablas y elementos

Nombre: Modal para agregar línea de detalle en vista móvil	**Tipo**: No Funcional	**Requisito**: RS-34
Descripción: El sistema debe presentar un modal para agregar la línea de factura en vista móvil.		
Justificación: De la misma forma que el requerimiento anterior (ver RS-33), este requerimiento podría pertenecer a la presentación responsive del sistema (ver RS-29), pero tiene una particularidad mucho más distintiva, puesto que en la presentación en vista Desktop del sistema, las notas o productos de la factura se agregan visualmente operando sobre una nueva línea en la tabla que representa la sección del detalle de la factura, en dispositivos móviles, realizar este proceso era tedioso para el usuario, motivo por el cual se decidió rediseñar de forma la inserción del detalle, presentando un modal (en vez de la línea nueva), esto si bien no cambia en nada la funcionalidad para agregar la nueva línea, es un gran cambio de forma, que presenta una solución adecuada al problema en todos los dispositivos que no son ordenadores de escritorio o portátiles, este modal será presentado también para la edición de una línea en vista móvil.		

Tabla 4.6 Requisito RS-34. Modal para agregar línea de detalle en vista móvil

Es importante destacar que estos dos últimos requerimientos no funcionales (RS-33 y RS-34), puesto que son de los últimos requerimientos realizados cuando ya el sistema se había publicado online en el servidor de producción en fase de test, son una modalidad configurable por el usuario en la página de las opciones de la cuenta, el motivo es que al ser tan complicado individuar todo tipo de dispositivos, en primera fase se usó las dimensiones de boostrap, posteriormente se recibieron estas observaciones por los usuarios para agregar o editar productos y los problemas de visualización de las tablas.

Aunque fueron corregidos, existen usuarios que poseen mini laptops o tablets (que veían una presentación desktop en la primera publicación) y que se adaptaron a la visualización desktop por compleja que fuera, al ser un cambio visual tan grande, estos usuarios deseaban continuar utilizando la visualización clásica (desktop).

Técnicamente el usuario puede decidir la visualización que desea en el sistema configurando su cuenta, con el campo que forma parte de las verificaciones que vienen hechas al momento de presentan el entorno visual a cliente, leyendo este campo del KING.

4.4 Casos de Uso

Para comprender mejor los requisitos del software a desarrollar, se identifican los casos de uso, ya que en estos se permite describir escenarios típicos donde el usuario y el sistema interactúan entre sí, con la finalidad realizar una acción o proceso, reaccionando a un evento que inicia el actor principal tal como lo explica *Alistair Cockburn, (2000).*

En el contexto de ingeniería del software, los diagramas de casos de uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas, en otras palabras, es un diagrama que muestra la relación entre los actores y los casos de uso en un sistema, a continuación, se definen los casos de uso, presentándolos en un diagrama para posteriormente especificar cada uno de ellos.

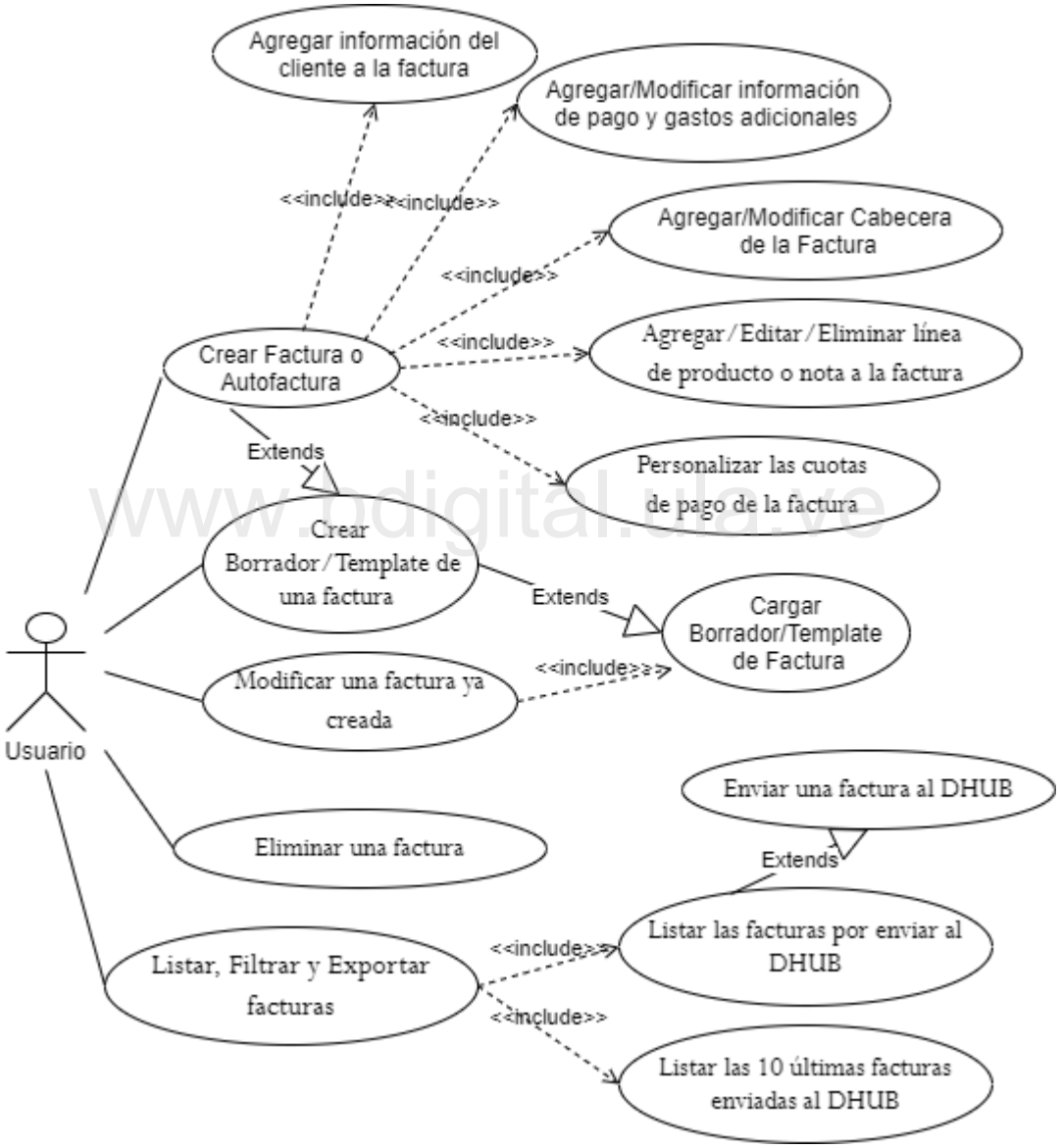


Figura 22: Casos de Uso

Caso de Uso	Nombre	Prioridad
CU-01	Crear Factura o Autofactura	5
CU-02	Agregar información del cliente a la factura	4
CU-03	Agregar/Modificar información de pago y gastos adicionales	4
CU-04	Agregar/Modificar Cabecera de la Factura	4
CU-05	Agregar/Editar/Eliminar línea de producto o nota a la factura	4
CU-06	Personalizar las cuotas de pago de la factura	3
CU-07	Crear Borrador/Template de una factura	4
CU-08	Cargar Borrador/Template de Factura	5
CU-09	Modificar una factura ya creada	5
CU-10	Eliminar una factura	4
CU-11	Listar, Filtrar y Exportar facturas	4
CU-12	Listar las 10 últimas facturas enviadas al DHUB	2
CU-13	Listar las facturas por enviar al DHUB	4
CU-14	Enviar una factura al DHUB	5

Prioridad: (1)No crítica; (2) Baja; (3) Intermedia; (4) Alta; (5) Crítica.

Tabla 5 Casos de Uso

Nombre: Crear Factura o Autofactura	**Caso de uso**: CU-01
Requerimientos Relacionados: RS-01, RS-02, RS-05, RS-09, RS-14, RS-17.	**Prioridad:** 5

Descripción: El usuario accede a la página de creación de la factura/autofactura.

Precondiciones: El usuario debe realizar anteriormente la creación y posterior configuración de al menos una "*Azienda*" (Empresa) en el subsistema de registro de datos, planificación y configuración. Esta debe ser seleccionada antes de acceder al menú de creación de facturas (opcionalmente se propone la página de selección de azienda).

Flujo Normal:

(1) El usuario desde el menú selecciona la opción Fatture -> Crea.

(2) Accede a la página de creación de la factura, precargando los datos disponibles en el sistema.

(3) Inserta todos los campos mínimos para poder crear la factura.

(4) Finaliza el proceso de creación confirmando el botón de "Crea Documento".

Flujos Alternativos:

(1) Si se elige autofactura se procede a realizan la inversión de la cabecera de la factura.

(2) Si no se culmina el proceso el sistema en automático crea el borrador de la factura.

(3) Posibilidad de crear un template de la factura.

Postcondiciones: El sistema realiza las validaciones necesarias de lógica y según los campos mínimos requeridos (cliente, detalle, cabecera, tipo, entre otros). Si la factura es lógicamente correcta crea el documento.

Tabla 5.1 Caso de Uso CU-01. Crear Factura o Autofactura

Nombre: Agregar información del cliente a la factura	**Caso de uso**: CU-02
Requerimientos Relacionados: RS-04, RS-06, RS-16.	**Prioridad: 4**
Descripción: El usuario inserta o modifica los datos del cliente de la factura.	
Precondiciones: El usuario debe encontrarse en la página de creación de la factura.	
Flujo Normal: (1) El usuario presiona el botón de selección de clientes. (2) Carga los datos de un cliente presente en el subsistema de registro de datos, planificación y configuración. (3) El sistema autocompleta y realiza los cálculos pertinentes según el cliente.	
Flujos Alternativos: (1) El usuario pulsa la opción de agregar un cliente lo cual permite de añadirlo mediante la función de creación de clientes del subsistema de registro de datos, planificación y configuración, autocompletando los datos posteriormente. (2) El usuario puede digitar los datos en cada campo directamente sin necesidad de guardar el cliente en sus registros y proceder con la creación de la factura.	
Postcondiciones: El sistema desbloquea todos los campos subsiguientes en la creación de la factura, puesto que si el cliente no está insertado no se permite agregar ninguno de los campos que contienen cálculos en base al régimen fiscal del cliente.	

Tabla 5.2 Caso de Uso CU-02. Agregar información del cliente a la factura

Nombre: Agregar/Modificar información de pago y gastos adicionales	**Caso de uso**: CU-03
Requerimientos Relacionados: RS-04, RS-07, RS-08, RS-14, RS-15, RS-16.	**Prioridad: 4**
Descripción: El usuario inserta o modifica los datos del pago o gastos adicionales de la factura como gastos de transporte o marca di bollo (sellos pagos para validar documentos públicos).	
Precondiciones: El usuario debe encontrarse en la página de creación de la factura y debe haber agregado un cliente para que venga calculado en la factura.	
Flujo Normal: (1) El usuario inserta los campos referentes al pago de la factura. (2) Inserta los gatos adicionales a la factura o el tipo de pago a realizar. (3) El sistema actualiza los campos pertinentes según la selección del usuario.	
Flujo Alternativo: (1) El usuario carga a partir de los datos registrados en el subsistema de registro de datos, planificación y configuración (previa selección de la "*Azienda*" que posee estos datos).	
Postcondiciones: El sistema realiza los cálculos en automático para la totalización de la factura.	

Tabla 5.3 Caso de Uso CU-03. Agregar/Modificar información de pago y gastos adicionales

Nombre: Agregar/Modificar Cabecera de la Factura	**Caso de uso**: CU-04
Requerimientos Relacionados: RS-04, RS-05, RS-14.	**Prioridad: 4**
Descripción: El usuario inserta los datos referente a la cabecera de la factura (Que no comprenden el cliente ni el remitente) como son la tipología de la factura, seccionales y fecha del documento.	
Precondiciones: El usuario debe encontrarse en la página de creación de la factura.	
Flujo Normal: (1) El usuario inserta los campos referentes a la cabecera de la factura. (2) El sistema muestra o esconde los campos pertinentes según la selección del usuario.	
Postcondiciones: El sistema realiza las validaciones necesarias para presentar la visualización adecuada con los campos requeridos según el tipo de factura y si es necesario elimina los datos de las líneas de la factura existente previo al cambio de la tipología.	

Tabla 5.4 Caso de Uso CU-04. Agregar/Modificar Cabecera de la Factura

Nombre: Agregar/Editar/Eliminar línea de producto o nota a la factura	**Caso de uso**: CU-05
Requerimientos Relacionados: RS-10, RS-11, RS-12, RS-13, RS-14, RS-16.	**Prioridad: 4**
Descripción: El usuario puede insertar, modificar, mover o eliminar las líneas del detalle de la factura.	
Precondiciones: El usuario debe encontrarse en la página de creación de la factura y haber insertado un cliente.	
Flujo Normal: (1) El usuario carga a partir de los datos registrados en el subsistema de registro de datos, planificación y configuración el detalle de un producto/articulo (previa selección de la "*Azienda*" que posee estos datos). (2) El sistema valida los campos obligatorios mínimos para insertar una línea de detalle.	
Flujos Alternativos: (1) El usuario inserta los campos referentes a la línea de detalle del producto/articulo o la nota. (2) El sistema valida los campos obligatorios mínimos para insertar una línea de detalle.	
Postcondiciones: El sistema realiza los cálculos pertinentes y actualiza los cálculos de totalización de la factura, adicionalmente realiza cálculos sobre campo especiales como el iva del productos según como lo indican las normativas fiscales (en base al cliente, la empresa y el producto).	

Tabla 5.5 Caso de Uso CU-05. Agregar/Editar/Eliminar línea de producto o nota a la factura

Nombre: Personalizar las cuotas de pago de la factura	**Caso de uso**: CU-06
Requerimientos Relacionados: RS-14, RS-15, RS-17.	**Prioridad: 3**
Descripción: El usuario puede insertar, modificar, mover o eliminar las líneas del detalle de la factura.	
Precondiciones: El sistema debe haber realizado los cálculos del *scadenzario* (calendario de pagos), para esto debe haber insertado al menos un producto (por tanto también un cliente) y el tipo de pago con la modalidad.	
Flujo Normal: (1) El usuario puede agregar, remover o modificar las cuotas de pago de la factura propuestas por el sistema según las indicaciones fiscales, lo cual le permite tener una flexibilización con sus clientes en el calendario de pagos pudiendo realizar cambios en una fecha de un pago o en el monto.	
Postcondiciones: Una vez modificado, agregado o removido un pago en el "*scadenzario*" el sistema procede a la realización de los cálculos pertinentes y posterior validación con respecto a la totalización de la factura, si no corresponden, da una alerta al usuario que no permite proceder con la creación del documento, inhabilitándolo en primera instancia y dándole una segunda alerta que permite crear el documento de todas formas (El scadenzario es una funcionalidad interna del KING y el DWEB para permitirle control de los pagos de sus clientes).	

Tabla 5.6 Caso de Uso CU-06. Personalizar las cuotas de pago de la factura

Nombre: Crear Borrador/Template de una factura	**Caso de uso**: CU-07
Requerimientos Relacionados: RS-02, RS-19.	**Prioridad: 4**
Descripción: El usuario puede convertir su factura en un borrador o témplate.	
Precondiciones: El usuario debe haber hecho cualquier modificación en la factura.	
Flujo Normal: (1) El sistema en automático guarda una "*bozza*" (borrador) de la factura cada vez que se realiza un cambio sobre la factura, este documento podrá posteriormente ser cargado para retomar el estado de la factura y continuar con el trabajo si es interrumpido por algún motivo.	
Flojo Alternativo: (1) El usuario puede decidir en cualquier momento guardar la "*bozza*" o template (único flujo posible).	
Postcondiciones: Una vez realizada alguna modificación o por instrucción directa del usuario, la factura se convierte en formato Json y es guardada para dar al cliente la posibilidad de tomar su ultimo estado.	
Nota: Cabe destacar que la diferencia del concepto entre una ***bozza*** y un **template** son que el template solo puede ser creado por usuarios con los privilegios necesarios (proyectada a ser función premium) y que además la diferencia de forma fundamental es que la *bozza* es eliminada en automático una vez que se crea el documento, mientras que el template queda registrado en el sistema permitiendo realizar una nueva factura precargando datos del último estado del template (ideal para facturas de servicios o pagos recurrentes) .	

Tabla 5.7 Caso de Uso CU-07. Crear Borrador/Template de una factura

Nombre: Cargar Borrador/Template de Factura	**Caso de uso**: CU-08
Requerimientos Relacionados: RS-01, RS-03, RS-05, RS-14.	**Prioridad: 4**
Descripción: El usuario debe tener la posibilidad de cargar el estado de una factura a partir de un borrador o template.	
Precondiciones: El sistema debe haber generado en automático un borrador de la factura o el usuario debe haber anteriormente creado un template de una factura.	
Flujo Normal: (1) El usuario va al menú de borradores o template y procede con la selección de uno. (2) El sistema procede a cargar los datos y recupera el estado de la factura.	
Postcondiciones: Si la carga es un borrador, la respuesta es la página de la creación de la factura con los últimos datos guardados en el borrador de la factura, mientras que si la carga es de un template, el sistema realiza exactamente el mismo procedimiento, con la diferencia de recalcular los rangos de datos disponibles para el documento como la fecha y el último número de factura correcto.	

Tabla 5.8 Caso de Uso CU-08. Cargar Borrador/Template de Factura

Nombre: Modificar una factura ya creada	**Caso de uso**: CU-09
Requerimientos Relacionados: RS-14, RS-18, RS-21, RS-22.	**Prioridad: 5**
Descripción: El usuario debe tener la posibilidad de modificar una factura ya creada bajo ciertas circunstancias específicas.	
Precondiciones: El sistema debe habilitar la modificación de facturas ya creadas, si y solo si, se creó previamente la factura y esta no ha sido aún enviada al SDI (puesto que una vez enviada puede ser aceptada y la factura no puede cambiar) o si la factura ha sido enviada y posteriormente descartada por el SDI (Normalmente los descartes se producen por errores en los datos de la factura y deben ser modificadas).	
Flujo Normal: (1) El usuario va a la página principal donde se listan las facturas por enviar. (2) Selecciona la factura que desea modificar.	
Postcondiciones: Esta selección genera un borrador a partir de los datos de la factura y reutiliza los módulos de carga de borrador y creación de factura para permitir generar un nuevo documento, todos los datos previos de la factura generada anteriormente vienen actualizados con los cambios realizados, posteriormente la factura cambia al estado de poder ser reenviada al SDI.	

Tabla 5.9 Caso de Uso CU-09. Modificar una factura ya creada

Nombre: Eliminar una factura	**Caso de uso**: CU-10
Requerimientos Relacionados: RS-21, RS-25, RS-26.	**Prioridad: 4**

Descripción: El usuario debe tener la posibilidad de eliminar una factura ya creada bajo ciertas circunstancias específicas.

Precondiciones: El sistema debe habilitar la eliminación de facturas ya creadas, bajo las siguientes condiciones:

- Debe haberse completado la creación de la factura previamente.
- La factura que se desea eliminar no ha sido aún enviada al SDI.
- No se ha enviado al SDI ningún documento con numeración o fecha superior a la factura que se desea eliminar, puesto que también estas deben ser eliminadas (si existen).

Flujo Normal:

(1) El usuario va a la página de resumen de facturas para la empresa.

(2) Selecciona la factura que desea eliminar.

(3) Confirma la eliminación de la factura (y las posibles posteriores).

Postcondiciones: Una vez confirmada la eliminación el sistema procede a eliminar la misma, en el particular caso de que existan facturas posteriores a la que se desea eliminar, y solo si ninguna de estas ha sido enviada, después de la alerta personalizada para este caso y la posterior confirmación, estas facturas también son eliminadas.

Tabla 5.10 Caso de Uso CU-10. Eliminar una factura

Nombre: Listar, Filtrar y Exportar facturas	**Caso de uso**: CU-11
Requerimientos Relacionados: RS-24, RS-26, RS-27, RS-28.	**Prioridad: 4**

Descripción: El usuario debe tener la posibilidad de listar, filtrar y exportar las facturas que ha enviado.

Precondiciones: El usuario debe encontrarse en la ventana principal (después de la selección de la *Azienda*) o en la vista del resumen de las facturas, podrá listar/filtrar solo las facturas que ya han sido creadas y realizar las exportaciones en los formatos que le permita la factura según su estado actual y la respuesta proporcionada por el SDI.

Flujo Normal:

(1) El usuario va a la página principal o la página del resumen de facturas.

(2) Ingresa los filtrar pertinentes para la búsqueda de la factura.

(3) Una vez encontrada elige el tipo de importación que desea realizar.

Postcondiciones: El sistema presenta los resultados de las búsquedas según los filtros y una vez seleccionada la exportación, procede a descargar el documento (o los documentos filtrados) en el formato indicado.

Tabla 5.11 Caso de Uso CU-11. Listar, Filtrar y Exportar facturas

Nombre: Listar las 10 últimas facturas enviadas al DHUB	**Caso de uso**: CU-12
Requerimientos Relacionados: RS-20.	**Prioridad: 2**
Descripción: El usuario debe tener la posibilidad de visualizar la ultimas 10 facturas enviadas al DHUB.	
Precondiciones: El usuario debe encontrarse en la ventana principal (después de seleccionar la *Azienda*) y debe haber enviado facturas al HUB.	
Flujo Normal: (1) El usuario selecciona una Azienda y es redireccionado a la página Dashboard de la Azienda.	
Postcondiciones: El sistema, en esta página, presenta la lista con las ultimas 10 facturas enviadas al DHUB.	

Tabla 5.12 Caso de Uso CU-12. Listar las 10 últimas facturas enviadas al DHUB

Nombre: Listar las facturas por enviar al DHUB	**Caso de uso**: CU-13
Requerimientos Relacionados: RS-26.	**Prioridad: 4**
Descripción: El usuario debe tener la posibilidad de visualizar las facturas que han sido creadas y aún no han sido enviadas al DHUB o enviadas y no aceptadas por el SDI.	
Precondiciones: El usuario debe encontrarse en la ventana principal (después de seleccionar la *Azienda*) o en la vista del resumen de las facturas de la empresa.	
Flujo Normal: (1) El usuario se posiciona en la página principal o la página del resumen de facturas. (2) Se presenta la lista de las facturas por enviar al DHUB o el icono para enviar la factura respectivamente.	
Postcondiciones: El sistema presenta los resultados evidenciando las facturas por enviar con la imagen de un aeroplano (resaltando con un color las facturas descartadas que deben ser enviadas nuevamente).	

Tabla 5.13 Caso de Uso CU-13. Listar las facturas por enviar al DHUB

Nombre: Enviar una factura al DHUB	**Caso de uso**: CU-14
Requerimientos Relacionados: RS-20**,** RS-23, RS-26.	**Prioridad: 5**
Descripción: El usuario debe tener la posibilidad de enviar una factura al DHUB.	
Precondiciones: El usuario debe encontrarse en cualquiera de las páginas que listan las facturas habilitando el icono con la acción de envió en aquellas que no han sido aún enviadas o fueron descartadas (deben reenviarse).	
Flujo Normal: (1) El usuario filtra las facturas para visualizar cuales tienen la posibilidad de ser enviadas. (2) Pulsa el botón de envió de la factura.	
Postcondiciones: El sistema, realiza el envió de la factura al DHUB y el icono con la opción de envió desaparece de la vista del usuario (posteriormente si viene descartada la factura el icono reaparece con otro color). El DHUB se encarga en autonomía del proceso de envió de la factura al SDI y de la gestión de las respuestas.	

Tabla 5.14 Caso de Uso CU-14. Enviar una factura al DHUB

CAPITULO 5

Diseño e implementación del sistema.

En este capítulo se abordarán los puntos referentes a las fases de diseño e implementación del subsistema propuesto, en base a la planificación realizada y la metodología de desarrollo propuesta, se muestras los diferentes aspectos que fueron tomados en consideración para lograr el producto final.

Evaluaremos las etapas principales involucradas en el desarrollo del proyecto basadas en la definición de las especificaciones, los requerimientos de usuario y los casos de uso, los cuales fueron premisa en la concepción del producto final presentado.

Nota: Cabe destacar que, por motivo de los acuerdos de confidencialidad firmados con la empresa, la información presentada en este capítulo está limitada en cuanto a los derechos de autoría y propiedad intelectual del software desarrollado.

5.1 Fase de planificación

Es la fase principal en la etapa de la estructuración para el desarrollo del subsistema propuesto, donde se realiza el análisis necesario para entender y definir todos los posibles elementos que interactúan con el subsistema, como pueden ser los tipos de usuarios del sistema, los requerimientos fiscales, las tipologías de facturas, los campos requeridos y las validaciones necesarias según la normativa fiscal que envuelve cada proceso de la facturación electrónica, entre otros.

Para poder realizar una planificación adecuada esta fase se rige bajo las normas dictadas en la metodología de desarrollo Scrum, logrando, de esta forma el camino más efectivo para realizar una comunicación continua entre las partes involucradas en el desarrollo de este subsistema y de la integración final en el entorno DWEB, además de continuamente revisar la correcta implementación que al final pueda ser compatible en la integración con los programas existentes DHUB y KING.

Esta fase es fundamental para poder realizar el producto que sea capaz de realizar una factura fiscalmente valida según las normas dictadas por *L'Agenzia delle Entrate*.

A continuación, procedemos con la definición de cada uno de los diversos elementos que comprenden la fase de planificación.

5.1.1 Planificación de Sprints

La planificación de los sprints es el elemento principal de la metodología de desarrollo elegida para la realización de nuestro proyecto, a través de esta planificación se puede subdividir el desarrollo de un sistema complejo y su funcionalidad en pequeñas partes, que progresivamente forman un todo (el producto final), estas actividades se diseñan partiendo de la información recolectada en el proceso de los análisis de los requerimientos del sistema.

Esto nos permite realizar una reunión semanal entre el tutor, los asesores empresariales y establecer la planificación semanal, en la cual se programan las actividades a desarrollar para cumplir con las metas de cada sprint.

Dentro de esta planificación también se evalúa continuamente el estado del desarrollo del producto y se realiza el proceso de retroalimentación con el reparto del personal que realiza test, para los cuales se destinó un ambiente de test dedicado en primera instancia a las pruebas a realizar entre cada sprint, pero también pensado posteriormente para los desarrollos futuros una vez que el producto sea entregado y publicado en el ambiente de producción.

Adicionalmente Datalog Italia Srl, cuenta con un elenco de clientes privilegiados por sus relaciones comerciales, los cuales obtuvieron acceso a esta plataforma de test y que realizaban una participación activa en el desarrollo del proyecto como clientes finales del producto pensado, aportado también sus ideas y observaciones con el pasar de cada sprint, a través de la comunicación directa con el reparto de test de Datalog.

Como herramienta de planificación interna se eligió la plataforma Backlog, en la cual se refleja la asignación de actividades a realizar cada semana mediante un tablero de trabajo donde los participantes, son los encargados de desarrollo de cada una de las plataformas a integrar en el sistema principal, los responsables de la organización del proyecto (tutores empresariales), el responsable de la retroalimentación obtenida del reparto de test y a su vez de los clientes con acceso a la plataforma de test y los encargados de establecer las metodologías para las respectivas comunicaciones con las aplicaciones DHUB y KING.

En las siguientes imágenes se puede apreciar el uso de la plataforma, la cual está basada en los conceptos de la metodología de desarrollo ágil Scrum, cada uno de los actores mencionados en el texto anterior representas los roles descritos por la metodología en el marco teórico.

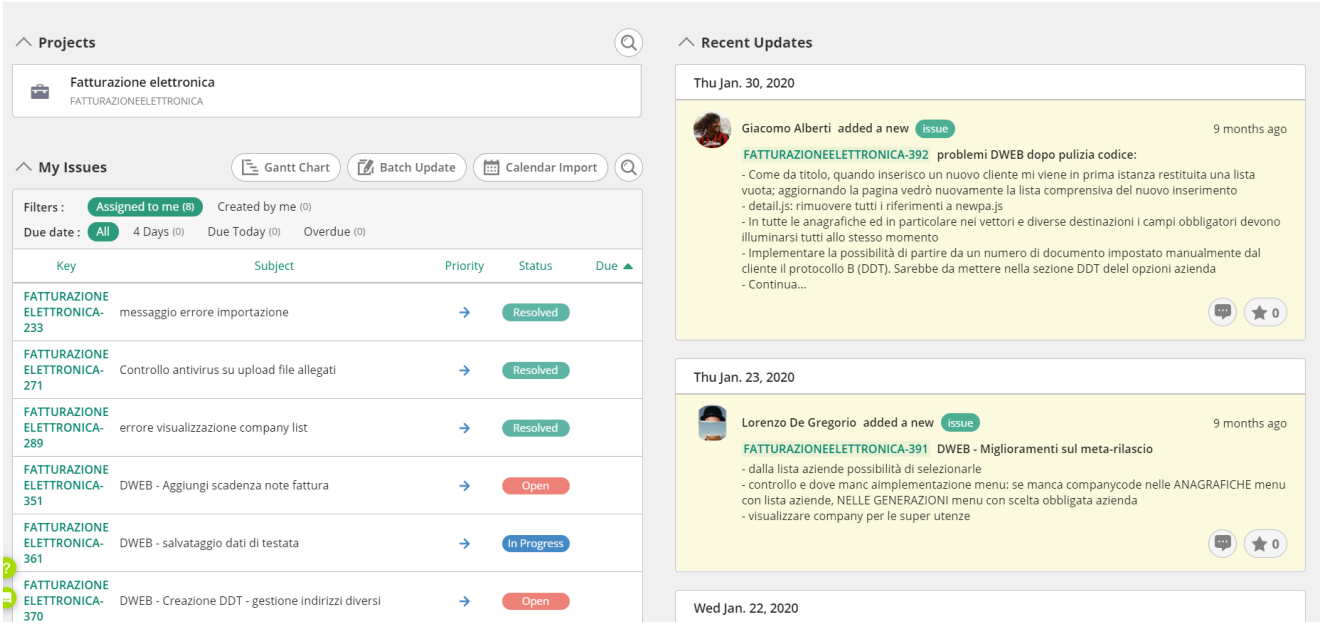


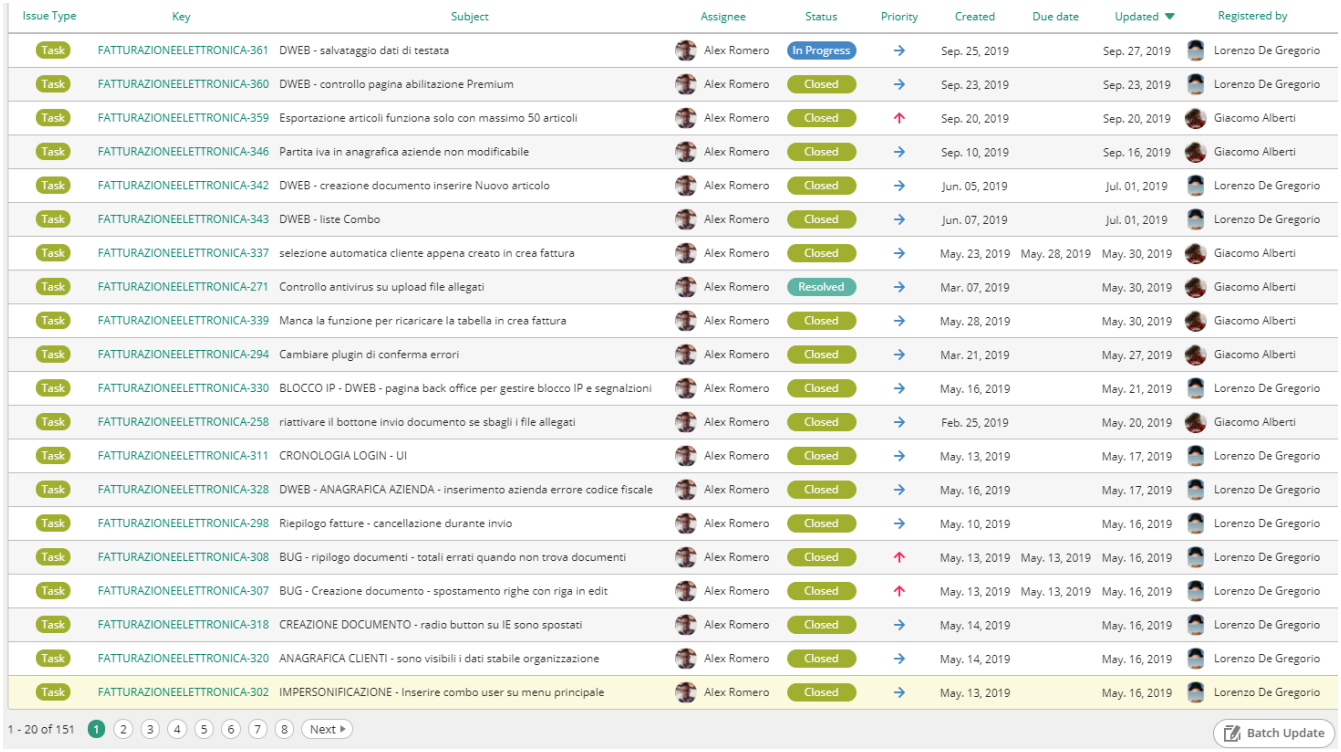
Figura 23: Dashboard Backlog

Issue Type	Key	Subject	Assignee	Status	Priority	Created	Due date	Updated	Registered by
Task	FATTURAZIONEELETTRONICA-317	API google maps	Giacomo Alberti	Closed	→	May. 13, 2019		Sep. 30, 2019	Lorenzo De Gregorio
Task	FATTURAZIONEELETTRONICA-360	DWEB - controllo pagina abilitazione Premium	Alex Romero	Closed	→	Sep. 23, 2019		Sep. 23, 2019	Lorenzo De Gregorio
Task	FATTURAZIONEELETTRONICA-359	Esportazione articoli funziona solo con massimo 50 articoli	Alex Romero	Closed	↑	Sep. 20, 2019		Sep. 20, 2019	Giacomo Alberti
Task	FATTURAZIONEELETTRONICA-346	Partita iva in anagrafica aziende non modificabile	Alex Romero	Closed	→	Sep. 10, 2019		Sep. 16, 2019	Giacomo Alberti
Task	FATTURAZIONEELETTRONICA-342	DWEB - creazione documento inserire Nuovo articolo	Alex Romero	Closed	→	Jun. 05, 2019		Jul. 01, 2019	Lorenzo De Gregorio
Task	FATTURAZIONEELETTRONICA-343	DWEB - liste Combo	Alex Romero	Closed	→	Jun. 07, 2019		Jul. 01, 2019	Lorenzo De Gregorio
Task	FATTURAZIONEELETTRONICA-333	Errore registrazione nuovo utente in fatturetest	Lorenzo De Gregorio	Closed	→	May. 17, 2019		Jun. 26, 2019	Giacomo Alberti
Task	FATTURAZIONEELETTRONICA-316	BACk OFFICE - info spazio utilizzato e disponibile	Lorenzo De Gregorio	Closed	→	May. 13, 2019		Jun. 26, 2019	Lorenzo De Gregorio
Task	FATTURAZIONEELETTRONICA-341	DWEB - API - Zip&Download	Lorenzo De Gregorio	Closed	→	May. 30, 2019		Jun. 26, 2019	Lorenzo De Gregorio
Task	FATTURAZIONEELETTRONICA-315	BACK OFFICE - cancellazione di tutto	Lorenzo De Gregorio	Closed	→	May. 13, 2019		Jun. 05, 2019	Lorenzo De Gregorio
Task	FATTURAZIONEELETTRONICA-338	DWEB - Gestione latenza invio documenti	Lorenzo De Gregorio	Closed	→	May. 27, 2019		Jun. 05, 2019	Giacomo Alberti
Task	FATTURAZIONEELETTRONICA-340	DWEB - time line documenti	Lorenzo De Gregorio	Closed	→	May. 30, 2019		Jun. 05, 2019	Lorenzo De Gregorio
Task	FATTURAZIONEELETTRONICA-336	DWEB - creazione documento	Maria Monsegui	Closed	→	May. 23, 2019		May. 30, 2019	Lorenzo De Gregorio
Task	FATTURAZIONEELETTRONICA-337	selezione automatica cliente appena creato in crea fattura	Alex Romero	Closed	→	May. 23, 2019	May. 28, 2019	May. 30, 2019	Giacomo Alberti
Bug	FATTURAZIONEELETTRONICA-334	Visualizzazione prezzo in edit in PARCELLA	Maria Monsegui	Closed	→	May. 17, 2019		May. 30, 2019	Giacomo Alberti
Bug	FATTURAZIONEELETTRONICA-151	Errore visualizzazione - doppia scrollbar in anteprima documento	Maria Monsegui	Closed	→	Dec. 10, 2018		May. 30, 2019	Giacomo Alberti
Task	FATTURAZIONEELETTRONICA-339	Manca la funzione per ricaricare la tabella in crea fattura	Alex Romero	Closed	→	May. 28, 2019		May. 30, 2019	Giacomo Alberti
Task	FATTURAZIONEELETTRONICA-335	DWB - Gestione icona user e azienda	Lorenzo De Gregorio	Closed	→	May. 22, 2019		May. 27, 2019	Lorenzo De Gregorio
Task	FATTURAZIONEELETTRONICA-294	Cambiare plugin di conferma errori	Alex Romero	Closed	→	Mar. 21, 2019		May. 27, 2019	Giacomo Alberti
Task	FATTURAZIONEELETTRONICA-310	CRONOLOGIA LOGIN - Chiamata per segnalazione eventi strani	Lorenzo De Gregorio	Closed	→	May. 13, 2019		May. 23, 2019	Lorenzo De Gregorio

1 - 20 of 302 1 2 3 4 5 6 7 8 9 ... Next

Batch Update

Figura 24: Vista General de actividades Realizadas



Issue Type	Key	Subject	Assignee	Status	Priority	Created	Due date	Updated	Registered by
Task	FATTURAZIONEELETTRONICA-361	DWEB - salvataggio dati di testata	Alex Romero	In Progress	→	Sep. 25, 2019		Sep. 27, 2019	Lorenzo De Gregorio
Task	FATTURAZIONEELETTRONICA-360	DWEB - controllo pagina abilitazione Premium	Alex Romero	Closed	→	Sep. 23, 2019		Sep. 23, 2019	Lorenzo De Gregorio
Task	FATTURAZIONEELETTRONICA-359	Esportazione articoli funziona solo con massimo 50 articoli	Alex Romero	Closed	↑	Sep. 20, 2019		Sep. 20, 2019	Giacomo Alberti
Task	FATTURAZIONEELETTRONICA-346	Partita iva in anagrafica aziende non modificabile	Alex Romero	Closed	→	Sep. 10, 2019		Sep. 16, 2019	Giacomo Alberti
Task	FATTURAZIONEELETTRONICA-342	DWEB - creazione documento inserire Nuovo articolo	Alex Romero	Closed	→	Jun. 05, 2019		Jul. 01, 2019	Lorenzo De Gregorio
Task	FATTURAZIONEELETTRONICA-343	DWEB - liste Combo	Alex Romero	Closed	→	Jun. 07, 2019		Jul. 01, 2019	Lorenzo De Gregorio
Task	FATTURAZIONEELETTRONICA-337	selezione automatica cliente appena creato in crea fattura	Alex Romero	Closed	→	May. 23, 2019	May. 28, 2019	May. 30, 2019	Giacomo Alberti
Task	FATTURAZIONEELETTRONICA-271	Controllo antivirus su upload file allegati	Alex Romero	Resolved	→	Mar. 07, 2019		May. 30, 2019	Giacomo Alberti
Task	FATTURAZIONEELETTRONICA-339	Manca la funzione per ricaricare la tabella in crea fattura	Alex Romero	Closed	→	May. 28, 2019		May. 30, 2019	Giacomo Alberti
Task	FATTURAZIONEELETTRONICA-294	Cambiare plugin di conferma errori	Alex Romero	Closed	→	Mar. 21, 2019		May. 27, 2019	Giacomo Alberti
Task	FATTURAZIONEELETTRONICA-330	BLOCCO IP - DWEB - pagina back office per gestire blocco IP e segnalzioni	Alex Romero	Closed	→	May. 16, 2019		May. 21, 2019	Lorenzo De Gregorio
Task	FATTURAZIONEELETTRONICA-258	riattivare il bottone invio documento se sbagli i file allegati	Alex Romero	Closed	→	Feb. 25, 2019		May. 20, 2019	Giacomo Alberti
Task	FATTURAZIONEELETTRONICA-311	CRONOLOGIA LOGIN - UI	Alex Romero	Closed	→	May. 13, 2019		May. 17, 2019	Lorenzo De Gregorio
Task	FATTURAZIONEELETTRONICA-328	DWEB - ANAGRAFICA AZIENDA - inserimento azienda errore codice fiscale	Alex Romero	Closed	→	May. 16, 2019		May. 17, 2019	Lorenzo De Gregorio
Task	FATTURAZIONEELETTRONICA-298	Riepilogo fatture - cancellazione durante invio	Alex Romero	Closed	→	May. 10, 2019		May. 16, 2019	Lorenzo De Gregorio
Task	FATTURAZIONEELETTRONICA-308	BUG - ripilogo documenti - totali errati quando non trova documenti	Alex Romero	Closed	↑	May. 13, 2019	May. 13, 2019	May. 16, 2019	Lorenzo De Gregorio
Task	FATTURAZIONEELETTRONICA-307	BUG - Creazione documento - spostamento righe con riga in edit	Alex Romero	Closed	↑	May. 13, 2019	May. 13, 2019	May. 16, 2019	Lorenzo De Gregorio
Task	FATTURAZIONEELETTRONICA-318	CREAZIONE DOCUMENTO - radio button su IE sono spostati	Alex Romero	Closed	→	May. 14, 2019		May. 16, 2019	Lorenzo De Gregorio
Task	FATTURAZIONEELETTRONICA-320	ANAGRAFICA CLIENTI - sono visibili i dati stabile organizzazione	Alex Romero	Closed	→	May. 14, 2019		May. 16, 2019	Lorenzo De Gregorio
Task	FATTURAZIONEELETTRONICA-302	IMPERSONIFICAZIONE - Inserire combo user su menu principale	Alex Romero	Closed	→	May. 13, 2019		May. 16, 2019	Lorenzo De Gregorio

1 - 20 of 151 1 2 3 4 5 6 7 8 Next ▸ Batch Update

Figura 25: Lista de actividades personales Backlog

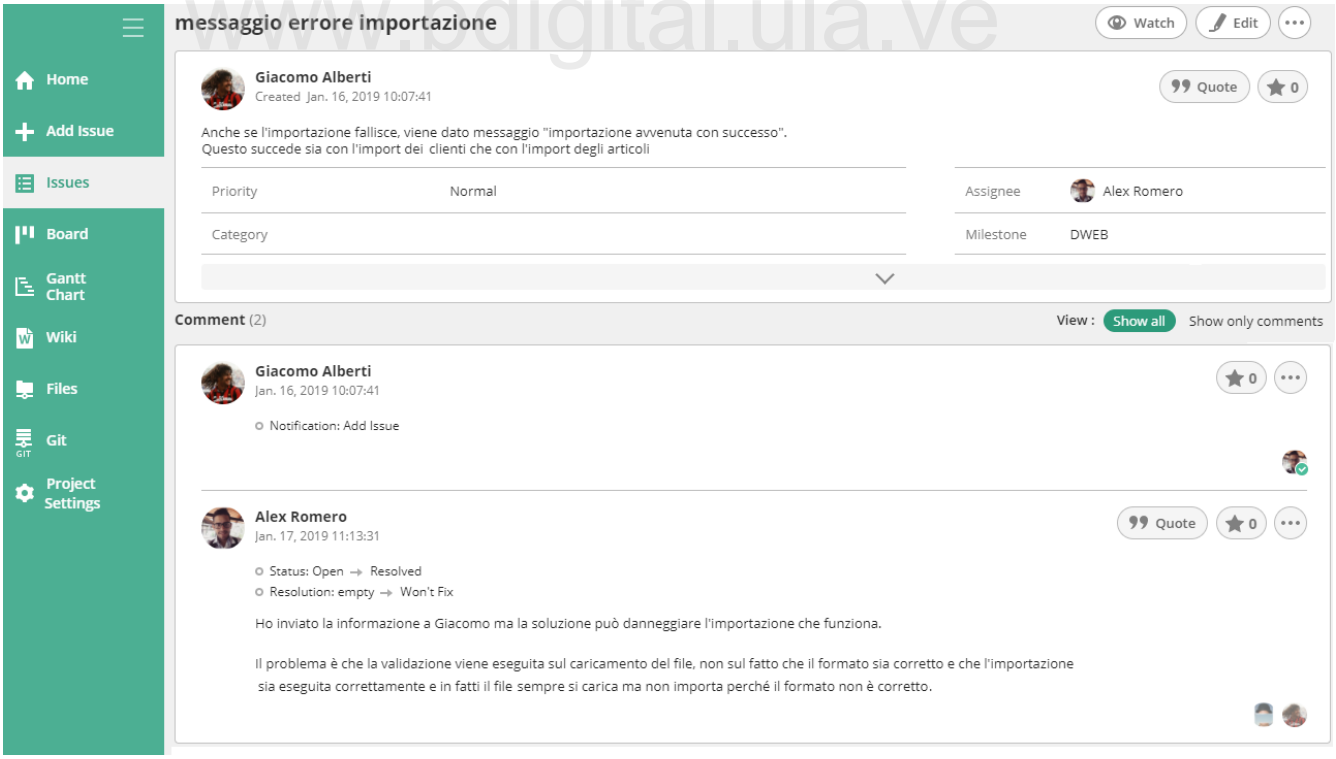


Figura 26: Ejemplo de error reportado por el reparto de test.

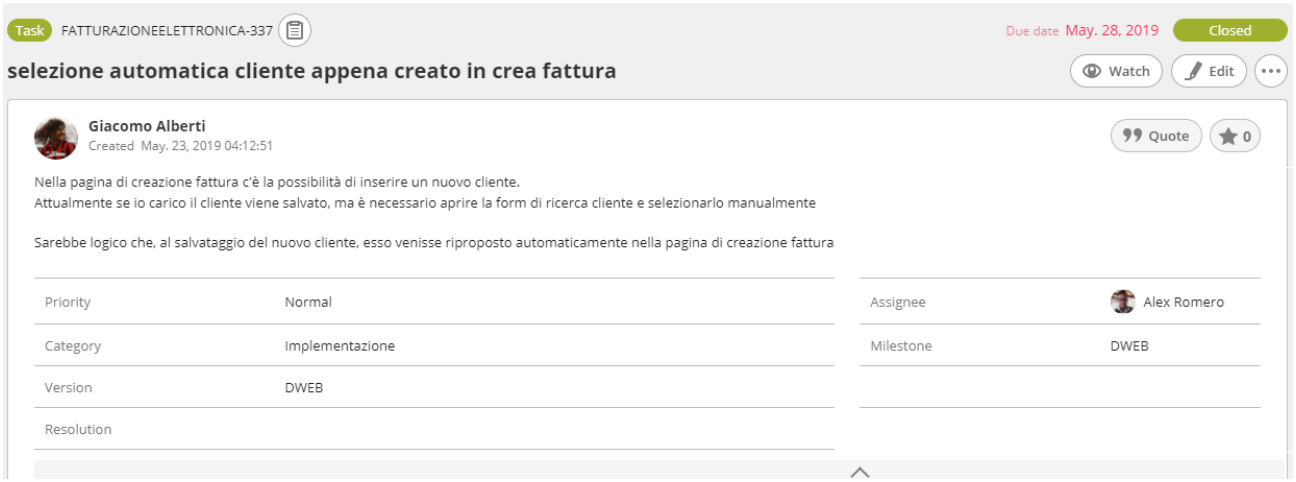


Figura 27: Ejemplo de actividad con fecha de vencimiento programada

Haciendo el uso correcto de la metodología se logra obtener una comunicación adecuada entre cada uno de los integrantes del grupo de trabajo, además permite obtener de forma eficiente y sencilla verificar el estado del proyecto en cada momento, así como el reporte de los errores o bugs que se presentaban en el programa al pasar por el área de testing, lo cual a su vez agiliza el proceso de resolución de errores y conlleva a la entrega de un producto final robusto y consistente a los requerimientos del software analizados.

5.1.2 Estudio y revisión de documentación

El desarrollo de este sistema en particular requirió de una fase de estudio, revisión y preparación teórica de la documentación referente a la facturación electrónica y los sistemas ya existentes con los cuales se deseaba interactuar para hacer uso de las funciones desarrolladas anteriormente por Datalog.

En cuanto comprende a la documentación fiscal, al no tener conocimientos previos sobre los procesos fiscales, legales y de realización de facturación en Italia, se requirió de un estudio profundo de las leyes y la normativa italiana, además Datalog puso a disposición un reparto conformado con personas expertas en el área, que estuvieron dispuesto a colaborar en la comprensión algún procedimiento o alguna normativa en particular si existían dudas o incomprensión de la misma.

Gran parte de esta investigación y estudio puede verse reflejada en el apartado Marco Teórico de este documento, otra parte como son los cálculos o procesos particular a realizar dependiendo de las variables del sistema (como tipo de documento o las normativas de cada campo) fueron plasmadas en el código del programa o conectadas con las funcionalidades ya existentes del KING, así como

también algunas otras validaciones y respuestas de los procesos fueron obtenidos por parte del sistema Datalog HUB.

Para esto se requirió también una fase de revisión de documentación de estos programas existentes, además de afortunadamente, contar con la presencia activa de los desarrolladores de estas herramientas de software en la empresa, los cuales también colaboraron en la explicación de procesos que podían ser complejos para la comprensión de una funcionalidad en particular.

5.2 Fase de diseño

La fase de diseño del subsistema realizado consta de dos partes fundamentales e importantes para el correcto funcionamiento del producto final propuesto, se realizó un análisis detallado de los elementos que comprenden el sistema integrado y la finalidad hacia la cual están destinados los elementos, esto permitió decidir los aspectos de diseño tanto a nivel de interfaz de usuario como de la estructuración del subsistema para la comunicación con todos los programas con los cuales debe integrarse.

En consecuencia, la fase de diseño puede ser dividida y representada, bajo dos categorías bien definidas como son la interfaz de usuario (front-end) y la estructura funcional a nivel de lógica de negocio del sistema (back-end), a continuación, procedemos con la definirlas de cada una y a la explicación sobre el análisis realizado, en base al cual se tomaron las decisiones de diseño visual y estructural.

5.2.1 Diseño de la interfaz de usuario

En el proceso del diseño de la interfaz de usuario entraron en consideración diferentes elementos necesarios para cumplir con los requerimientos del sistema integrado propuesto, entre los cuales destacan ciertos puntos requeridos por Datalog, entre los cuales destacan los siguientes:

- La interfaz de usuario propuesta debe presentar una armonía con la de los sistemas ya desarrollados, en particular con la plataforma DHUB, para que exista una armonización entre los programas que colaboran entre sí.
- Diseñar partiendo de la misma interfaz gráfica que presenta el DHUB, permitiendo a la rápida adaptación de los usuarios que ya usan este sistema para consultar el estado de las facturas enviadas al SDI.

- Se deben realizar cambios leves entre los colores y el logo para que los usuarios no se confundan en la plataforma en la cual se encuentran, visto que dentro del sistema propuesta se dispondrá de un acceso directo al Datalog HUB.
- La interfaz web de este subsistema estará directamente integrada con el subsistema de registro de datos, planificación y configuración. Por este motivo ambos proyectos deben ser desarrollados en simultaneo, lo que permitirá la reutilización de funcionalidades compartidas dentro de la creación de la factura.
- El diseño de la vista de creación de la factura deberá ser una vista con elementos bien definido, visto que, dependiendo de diversos factores, en los que resalta principalmente el tipo de documento, campos deben ser visibles o deberán esconderse sin que se pierda el orden de los elementos en el sistema.
- Cada sección de la vista creación factura debe ser independiente visualmente y claramente identificable para los usuarios visto la cantidad de campos que debe llenar el usuario en el formulario.
- Los campos de totalización de la factura deben ser siempre visibles y distinguibles en el proceso de creación.
- La línea de detalle o nota de la factura deberán variar en cantidad de columnas según la tipología de la factura, por lo tanto, la interfaz debe ser definida para soportar estos cambios sin perder la estructura.
- El sistema por la cantidad de usuarios estará principalmente estructurado para ser utilizado desde un ambiente desktop, aunque de igual forma la aplicación deberá contar con presentación diversa para dispositivos móviles y tablets, la cual sea flexible y adaptable, de forma que el usuario obtenga una visualización cómoda en la realización de la factura desde estos dispositivos. Se debe respetar el orden en los elementos de la factura en esta visualización.
- Existirán ciertas restricciones en la presentación Mobile, como puede ser la descarga de ciertos formatos del documento que no son soportados sobre el dispositivo móvil.

A continuación, se presentan capturas de pantalla de la aplicación DHUB, estas permiten ver el modelo de interfaz gráfica usado como punto de inicio para el diseño de la interfaz de nuestro subsistema, en el apartado correspondiente a la implementación de la interfaz podremos apreciar las similitudes entre ambos sistemas y los elementos que permiten diferenciarlos uno del otro.

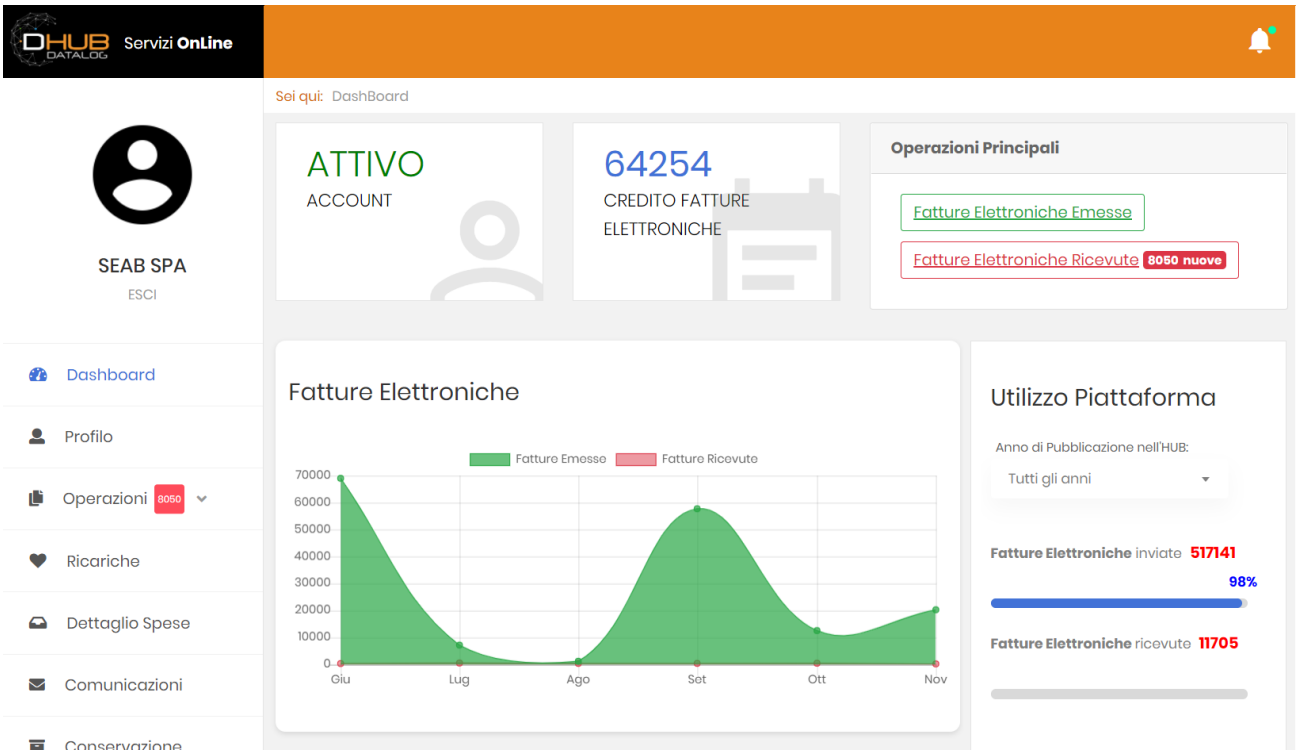


Figura 28: Dashboard DHUB

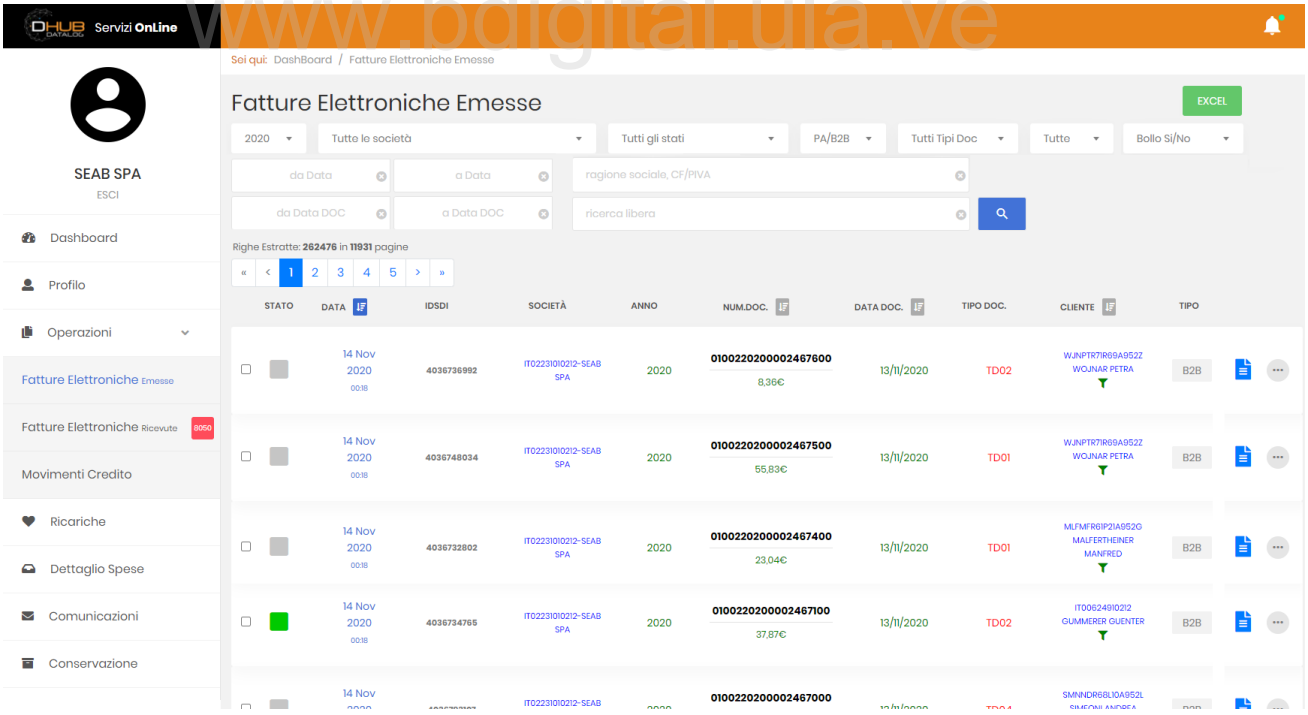


Figura 29: Listado de facturas DHUB

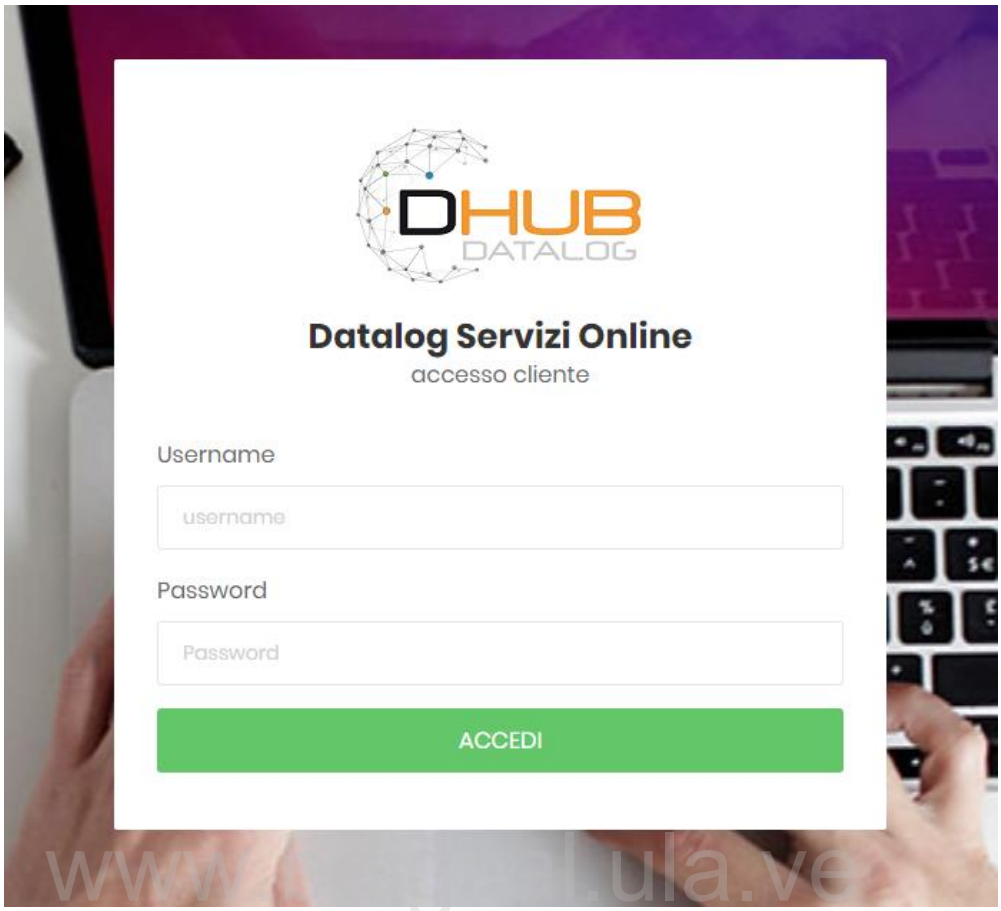


Figura 30: Login DHUB

Teniendo el conocimiento y la clara base plasmada en estas consideraciones, por su parte Datalog estableció una política de libertad de diseño gráfico sobre las ideas que pudiéramos presentar siempre y cuando no irrespetaran en forma o concepto la lista expuesta anteriormente. Esta clara guía (plantilla visual DHUB) y partiendo de las condiciones se decidió el diseño final de la interfaz de usuario que podremos observar en el apartado de implementación del sistema, en la fase de desarrollo.

5.2.2 Diseño de la estructura funcional

El diseño de la estructura funcional del sistema represento un enfoque mayor y más complejo por diversos factores tomados en cuenta para el desarrollo del sistema, las decisiones fueron tomadas principalmente por parte de los actores principales en los proyectos KING y DHUB, además participaron activamente los altos cargos de la empresa por motivos de políticas empresariales.

Para lograr una definición en la que todos los elementos de la empresa estuvieran de acuerdo, se realizó una participación activa de nuestra parte (desarrolladores de los subsistemas) aconsejando sobre las herramientas que podían proporcionar las soluciones que mejor se adaptaran para satisfacer sus necesidades, entre las cuales destacan:

- El uso de C# como lenguaje de programación por políticas de migración de la empresa de sus sistemas (se encontraban en proceso de actualización de sus programas de VB6, VB .NET a C#).
- La estructura de desarrollo final deberá soportar la comunicación con los sistemas KING, desarrollado en VB6 y el sistema DHUB, desarrollado en VB .NET.
- El sistema debe proveer una comunicación segura entre los programas del entorno haciendo uso de tokens de acceso privados a las funciones compartidas.
- El sistema DWEB deberá ser publicado tanto en un ambiente de test que tendrá su propia base de datos, como posteriormente en un ambiente de producción que compartirá datos con los clientes de producción que ya están registrados en Datalog.
- El sistema DWEB deberá contar con la base del paradigma de la programación orientada a objetos, desarrollando una estructura clara que permite definir los elementos de la factura, usando la abstracción como regla principal.

Esta fase del diseño del sistema requirió un estudio profundo de los sistemas existentes (DHUB y KING), el lenguaje en el cual fueron desarrollados, el tipo de comunicación a establecer con cada uno de ellos y de las tecnologías que podían establecer el tipo de conexión que se adapte a los requerimientos presentados.

Una vez realizado el estudio profundo de los sistemas y las tecnologías se estableció el diseño del sistema DWEB el cual es la base compartida entre la integración del subsistema propuesto y el subsistema de registro de datos, planificación y configuración.

Las decisiones tomadas a nivel del diseño de la estructura del sistema, para el desarrollo y comunicación las enumeramos a continuación:

1. Desarrollo del sistema usando el Framework de desarrollo de Windows MVC 4 en .NET C#/WinForm.
2. Definición de los modelos usando la abstracción de los elementos y el paradigma POO.

3. Integración con la base de datos ya existente de clientes en Datalog y gestión de los registros a través de las funciones presentes en el sistema KING.
4. Desarrollo de una DLL compilada del programa (KING), agregada al proyecto que permite la comunicación con el KING y sus bases de datos internos haciendo uso de las funciones expuestas en esta DLL.
5. Desarrollo por parte del encargado en el DHUB de una serie de WebMethods expuestos en el sistema públicos con validación a través de tokens de seguridad para establecer la comunicación con este sistema.
6. Construcción de una API DWEB por parte de uno de los responsables del proyecto que se encargara de integrar las DLL (KING) y funciones API (DHUB).

El diseño de esta estructura funcional permite al sistema propuesto cumplir con el objetivo principal de este proyecto, satisfaciendo las necesidades de la empresa y asegurándose además de no irrespetar las condiciones presentadas por la empresa Datalog Italia Srl. Permitiéndoles la oportunidad de ofrecer un nuevo servicio a sus clientes, obteniendo el máximo provecho de sus programas ya existentes e integrándolos permitiéndoles una mejora estructural entre sus productos comerciales.

5.3 Fase de Implementación

En esta sección se procede con la descripción de la fase de implementación del subsistema propuesto, tomando en cuenta todas las consideraciones realizadas durante el análisis de los requerimientos, las fases de planificación semanales y las fases de diseño tanto de interfaz como de estructura del subsistema.

El objetivo principal de este segmento es mostrar los resultados de las actividades planificadas y realizadas en el proceso de implementación del nuestro subsistema, estas actividades en forma resumida podemos detallarlas en la siguiente lista:

1. **Preparación de la organización**: Datalog Italia Srl, se encargó de reunir y disponer de todos los recursos necesarios para la realización de este proyecto.

2. **Planificación del sistema**: Todos los integrantes del equipo se encargaron de analizar y planificar las actividades para llevar a cabo el desarrollo del proyecto.

3. **Comunicación e información**: Cada una de las partes se mantuvo en constante comunicación para manejar la información de forma efectiva, cada uno de los equipos de desarrollo compartió información acerca de los avances que realizaba y los requerimientos, mientras que por su parte los elencos de especialistas y testing compartían información esencial de los procesos fiscales y los errores que podían encontrarse en el sistema, para esta actividad la organización a través de la plataforma backlog fue fundamental.

4. **Diseño y elaboración de la documentación**: De forma organizada el grupo de trabajo interno en Datalog presento la documentación necesaria para este desarrollo y los grupos de desarrollo presentaron las propuestas de diseño que fueron aprobadas posteriormente por los responsables en la empresa.

5. **Implementación y seguimiento**: Constantemente en el proceso de desarrollo del proyecto se realizó un seguimiento constante y evaluación continua del proyecto lo que permitió obtener un producto de alta calidad.

6. **Certificación**: Una vez culminado el proyecto fue sometido a diversos test tanto a nivel interno en Datalog, como a nivel externo, con pruebas de usuario, dispuestos a los clientes de la empresa que colaboraron activamente al final del proyecto con la verificación de la correcta implementación del software realizando un test final sobre el producto, la mayoría de sus observaciones fueron graficas o requerimientos de nuevas funcionalidades.

7. **Mantenimiento**: Posteriormente a la fase de desarrollo de este proyecto, fueron recomendados nuevas implementaciones para continuar con el proceso de mejora del producto, además que Datalog Italia Srl propuso agregar nuevas funcionalidades para ofrecer nuevos servicios a sus clientes, cabe destacar que el proceso de mantenimiento es importante visto que al ser un sistema que se rige por las leyes fiscales italianas, si cambia alguna ley, el programa deberá ser adaptado para cumplir con la misma.

Una vez detallado cada punto del proceso de implementación realizado procedemos a detallar los procesos de implementación de la interfaz gráfica y de la estructura del sistema.

5.3.1 Implementación de la interfaz de usuario

La interfaz de usuario es el vínculo entre el usuario y el programa de computadora. Una interfaz es un conjunto de comandos o menús a través de los cuales el usuario se comunica con el programa. Esta es una de las partes más importantes de cualquier programa ya que determina que tan fácilmente es posible que el programa haga lo que el usuario quiere hacer.

La creación de las interfaces de usuario ha sido un área del desarrollo de software que ha evolucionado dramáticamente a partir de la década de los setentas. Un programa muy poderoso con una interfaz pobremente elaborada tiene poco valor para un usuario no experto y puede obtener como resultado el rechazo de este hacia el sistema, todos estos elementos fueron tomados en cuenta en la fase de diseño de la interfaz.

Con la ventaja de tener una interfaz de usuario definida en el sistema DHUB el desarrollo de esta interfaz de usuario fue más sencillo, visto que se contaba con un punto de inicio ya establecido, aunque de igual forma durante el desarrollo de la interfaz de nuestro sistema, encontramos requerimientos específicos como podían ser validaciones de ciertos campo en tiempo real, guardado y recarga de los campos en automático y la complejidad de esconder o mostrar algunos campos dinámicamente sin perder la estructura de la visualización, motivo por el cual también se realizó un estudio de cada pantalla presentada al usuario.

Para el desarrollo se eligió utilizar el Framework de desarrollo de Windows MVC 4 en .NET C#/WinForm, el cual mediante su motor de vistas Razor contribuyo enormemente con la filosofía de reutilización del código, permitiendo mediante el uso de las vistas parciales incluir todos los elementos comunes de cada vista en modelos visuales parciales que después eran agregados en cada página donde eran requeridos.

En cuanto a el proceso de las validaciones se decidió utilizar en lenguaje de programación Jquery el cual permite interactuar con la vista y sus campos de forma inmediata, lo cual proporciono una ventaja en el proceso de validación de los campos y las llamadas a las funciones del controlador de cada vista, también permitió manejar los datos obtenidos por parte de los controladores (formato Json de cada modelo) y la serialización de los objetos.

En el desarrollo de la interfaz fue elegida como prioridad la visualización desktop y tablets de grande dimensiones por encima de los dispositivos móviles, las tablets pequeñas y las laptops de baja resolución, puesto que si ciertamente era importante la presentación Mobile para aquellos que desearan facturar a través de estos dispositivos, el proceso de creación de la factura es complejo y cuenta con muchos campos, lo cual hace realmente difícil el proceso de la creación de la factura desde un dispositivo móvil.

Esta premisa se mantuvo, al punto de que requirió el desarrollo una funcionalidad específica, en la cual el usuario puede configurar un campo para permanecer con la vista clásica (desktop) aun cuando hace uso de los dispositivos móviles o tablets.

Para el desarrollo de estos requerimientos se usó la construcción de la interfaz haciendo uso de CSS, en particular como base el Framework Boostrap en su versión 4, para poder adaptar cada vista a los requerimientos particulares de cada usuario y los diferentes dispositivos con los cuales accedan al sistema.

Posteriormente para los usuarios que realizaban el acceso a través de dispositivos móviles y que requerían una mejor presentación de ciertas áreas del sistema, se desarrollaron vistas particulares apoyándose siempre en boostrap para mostrar los campos a través de modales o tablas modificadas.

Para obtener un mejor manejo de estas tablas modificadas y aprovechando al máximo las ventajas de obtener representación de los datos a través de objetos fueron utilizados también los archivos Handles Bars con extensión .hbs de Jquery el cual permite formatear una estructura HTML de un simple objeto y posteriormente cargar todos los datos con un array de este tipo de objeto.

A continuación, se presentan capturas de pantalla del sistema DWEB ya desarrollado y en estado del proyecto en producción, se aconseja atención a nivel grafico para encontrar las similitudes entre el sistema DWEB y el sistema DHUB, también se recuerda que nuestro subsistema forma parte del DWEB por lo tal de todo el proyecto se presenta solo la parte correspondiente a la creación y el listado de las facturas.

En cuanto a la codificación del mismo no podemos compartir grandes fragmentos por motivo del acuerdo de confidencialidad, recibimos autorización de compartir el proceso de uso de los Handles Bars para ser presentado como una técnica que simplifica el manejo de los datos según la estructura propuesta.

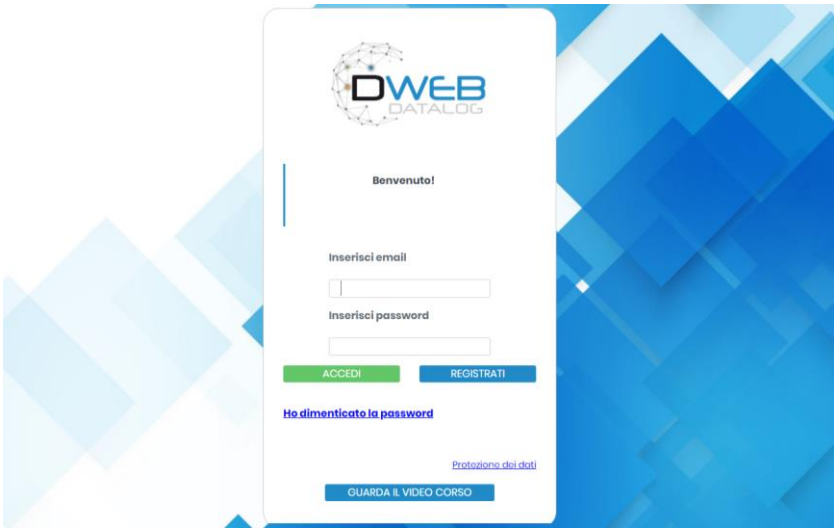


Figura 31: Login DWEB.

La siguiente imagen representa la segunda página de inicio (una vez que el cliente ha seleccionado una empresa sobre la cual desea trabajar) en la misma se listan las facturas por enviar y las ultimas 10 facturas enviadas al DHUB, se puede apreciar igualmente, las similitudes de interfaz en el sistema DWEB visto que estos datos son cargados a través de la Api expuesta en el sistema DHUB.

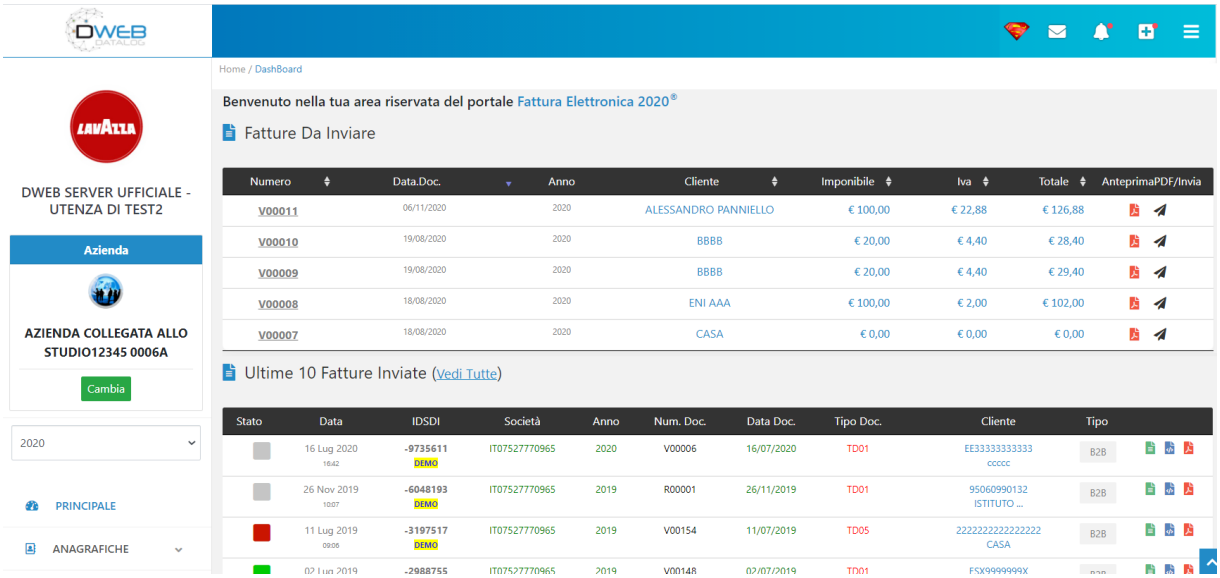


Figura 32: Pagina Home (Azienda)

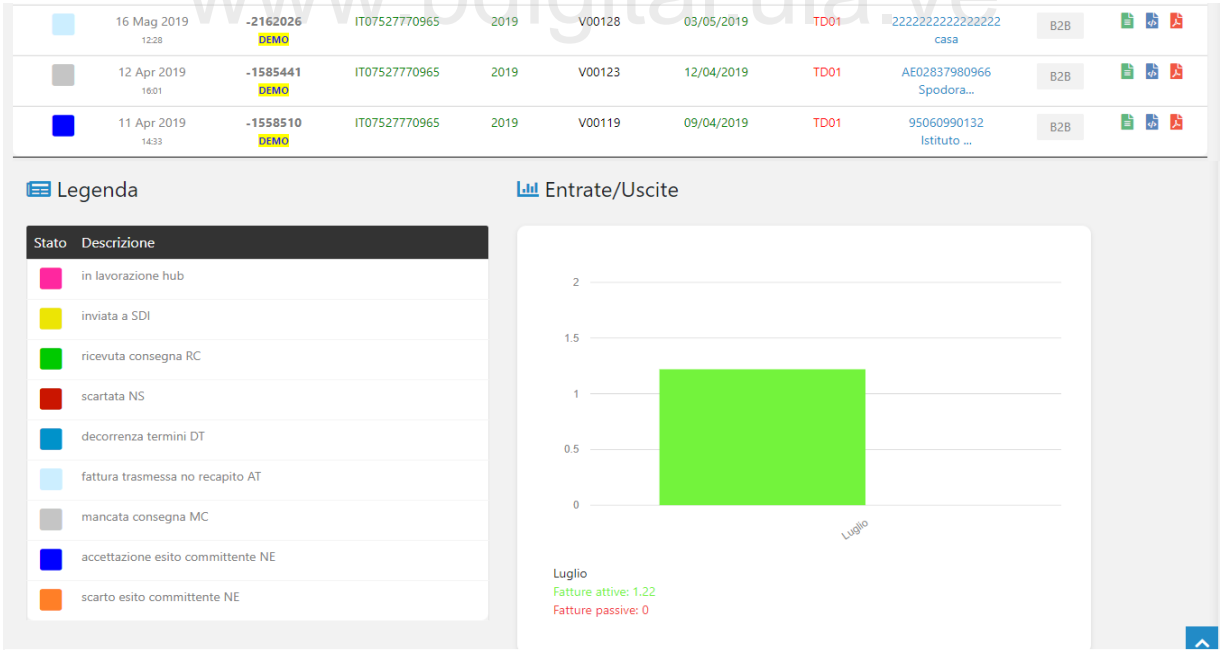


Figura 33: Pagina Home (Azienda) – Legenda modelo DHUB

Existe una página home principal, la cual interactúa directamente con el subsistema de registro de datos, planificación y configuración, por lo que no nos concierne mostrarla.

Por otra parte, en las siguientes imágenes se puede apreciar como la interfaz fue desarrollada con la capacidad de cambiar dinámicamente en base a la tipología del documento sin perder la estructura base de la página.

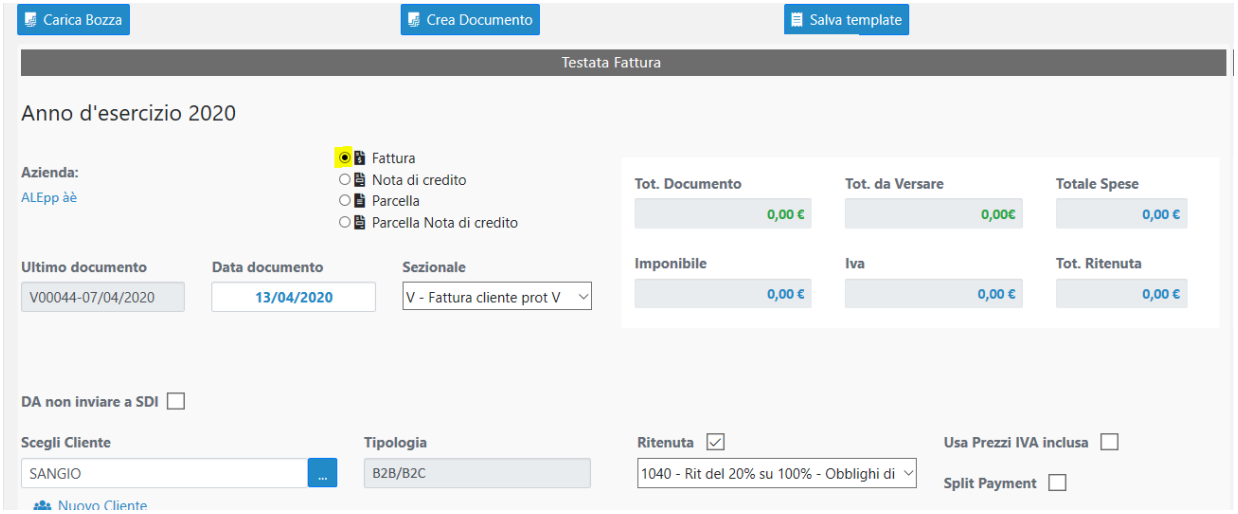


Figura 34: Vista creación factura – Tipo Documento Fattura

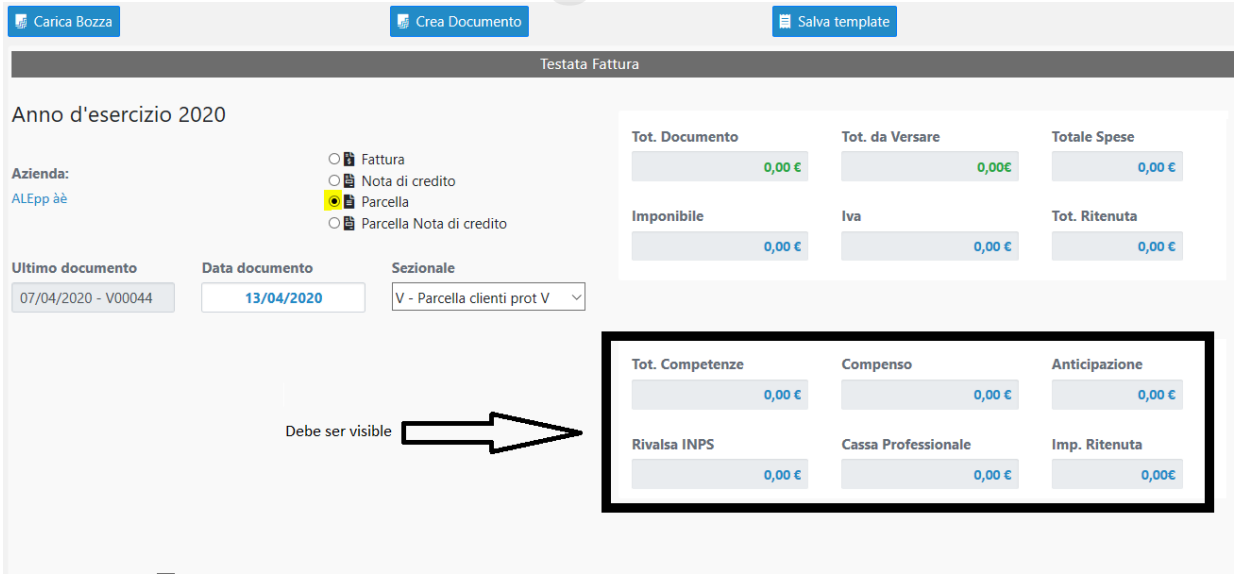


Figura 35: Vista creación factura – Tipo Documento Parcella

En la siguiente se aprecia el área de la vista de creación de factura que contiene los datos pertenecientes a la carga de datos del cliente de la factura, sea a través del formulario o de las funciones del subsistema de registro de datos, planificación y configuración.

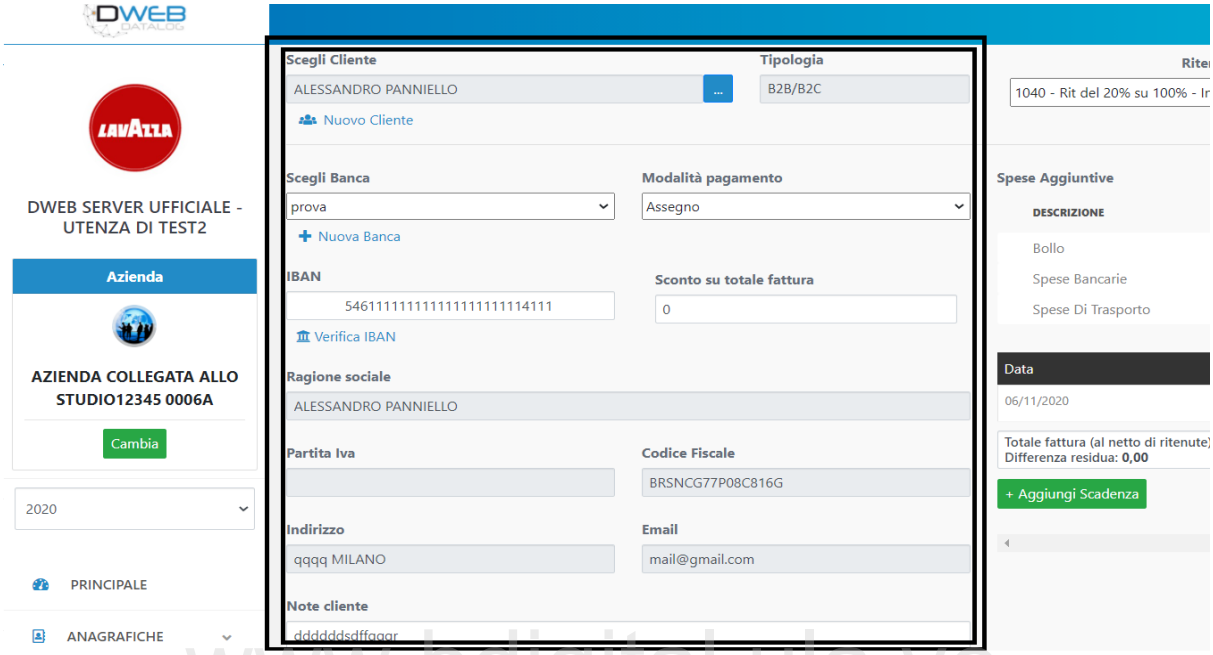


Figura 36: Vista creación factura – Datos del Cliente

Mientras que para la sección de los datos del pago (modalidad y planificación de los pagos) se puede apreciar la siguiente sección, que comparte algunos campos con los datos del cliente registrado, pero pueden ser modificables. Esta sección además contiene el área de configuración de otros gastos adicionales, como pueden ser marcas de sellos, costos de transporte o gastos bancarios.

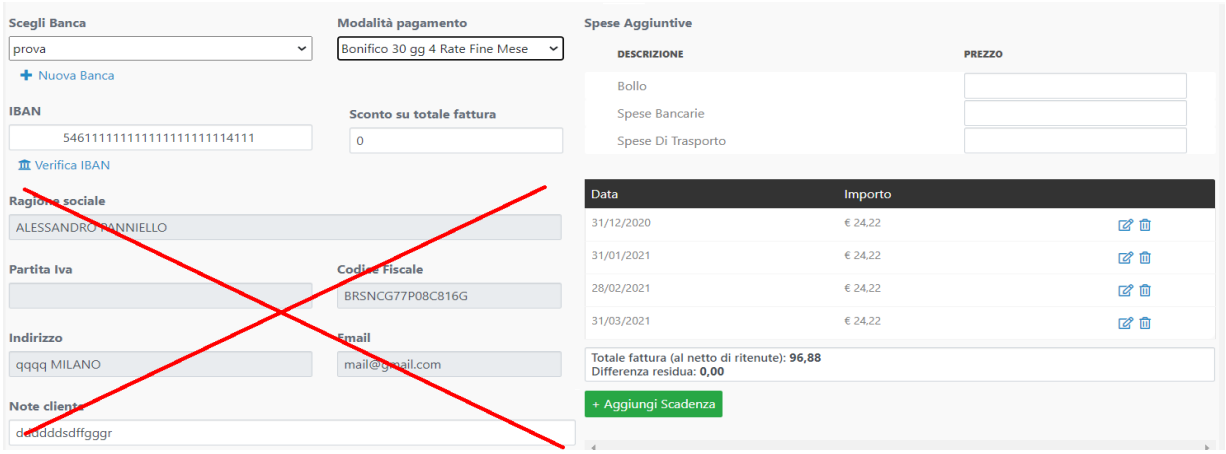


Figura 37: Vista creación factura – Datos de pago y gastos adicionales

En la siguiente imagen se muestra el área de detalle de la factura con diversos productos y la función de agregado de productos a la factura en vista clásica (Desktop).

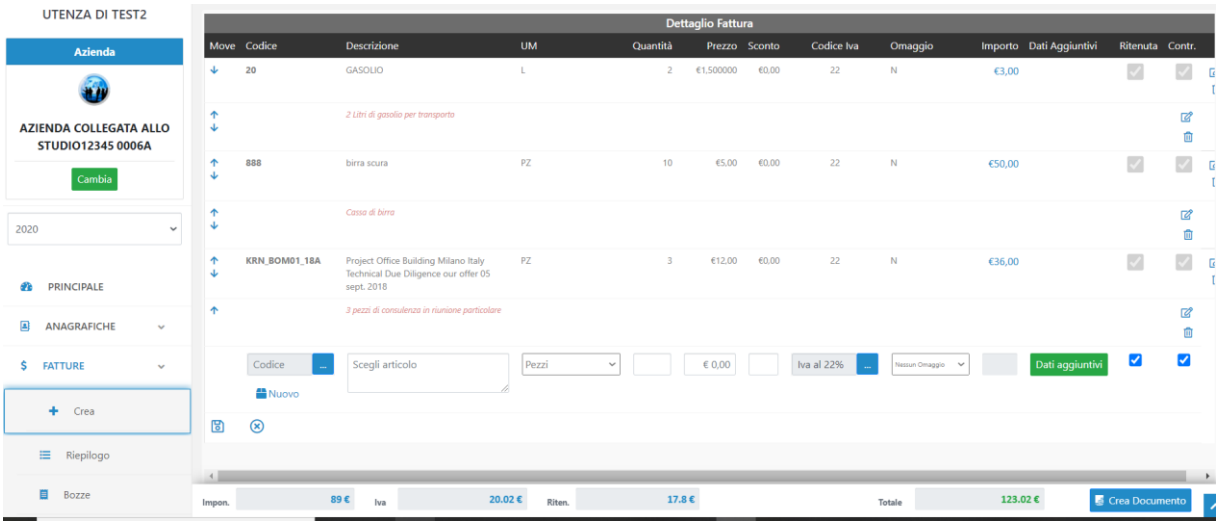


Figura 38: Vista creación factura – Detalle de la factura

En la siguiente pantalla puede apreciarse la vista del resumen de las facturas realizadas, en esta vista el cliente puede realizar las diversas funciones de envió y exportación para las facturas, como también las funciones de modificación, eliminación y duplicación de las facturas.

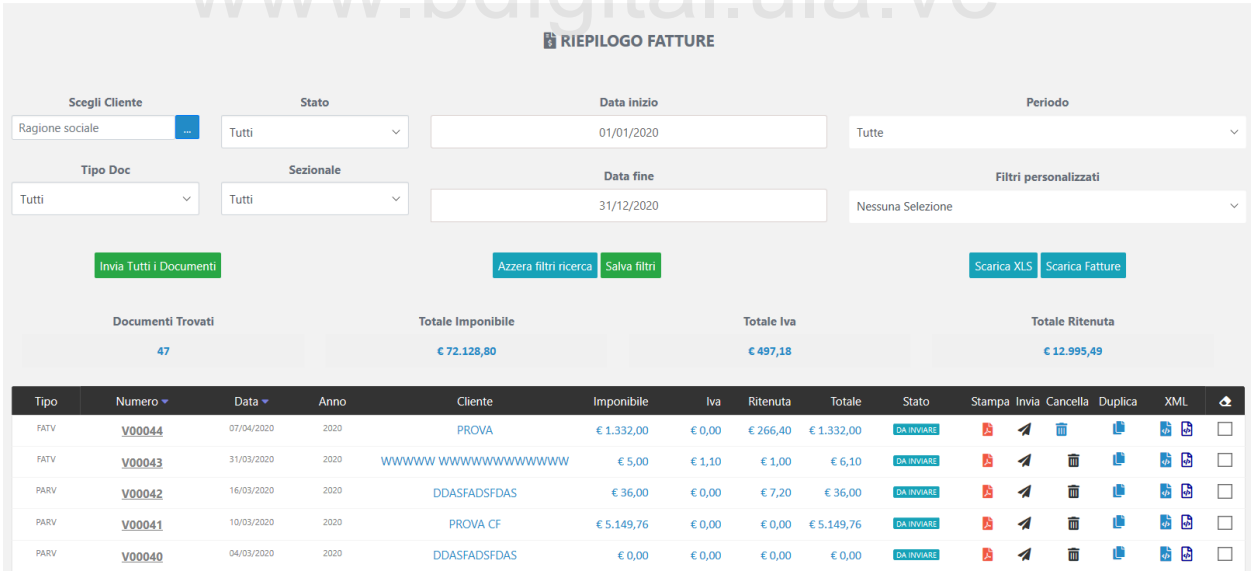


Figura 39: Vista resumen de facturas – Lista de las facturas creadas

También contamos con una vista en la cual se pueden listar los borradores de las facturas que aún no han sido creadas, desde esta opción el cliente puede recuperar el trabajo de cualquier factura que se encuentra en proceso de realización.

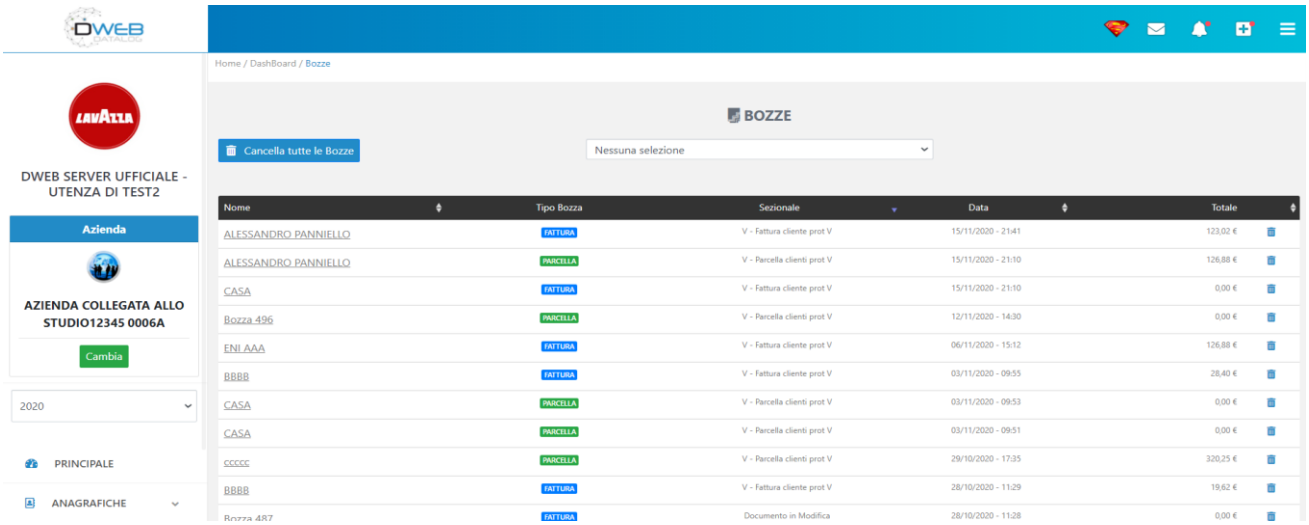


Figura 40: Borradores de facturas no creadas

A continuación, presentamos las vistas de previsualización de la factura y los formatos en los cual puede ser exportada, estos formatos son oficiales, proporcionados por las normal de generación del XML y un formato estandart de AssoSoftware (Asociación de desarrolladores de software de facturación).

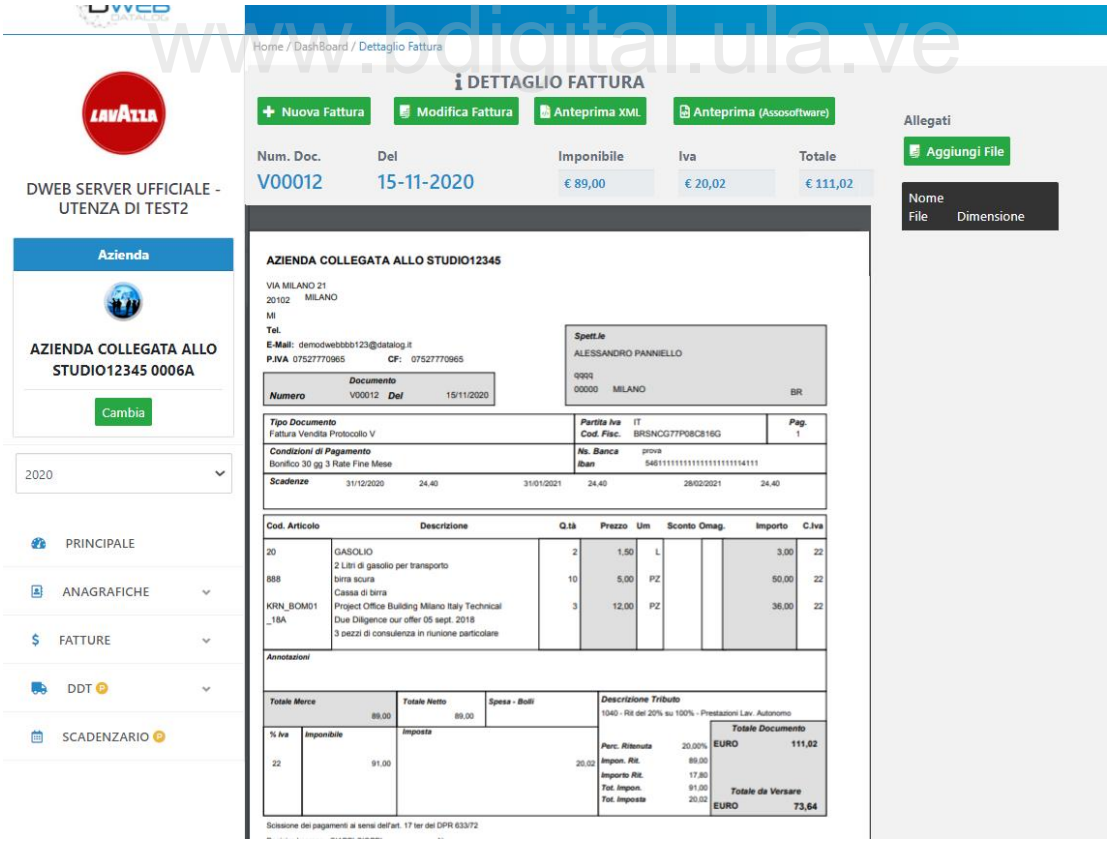


Figura 41: Vista previa de factura

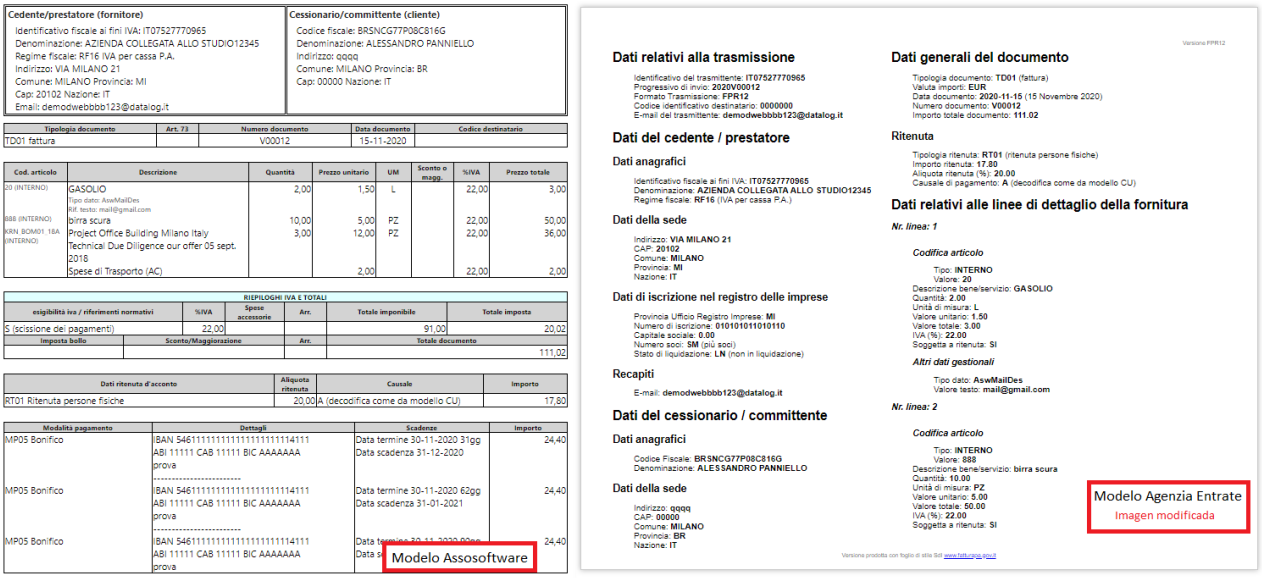


Figura 42: Vista previa factura - modelo Assosoftware y Agenzia delle Entrate

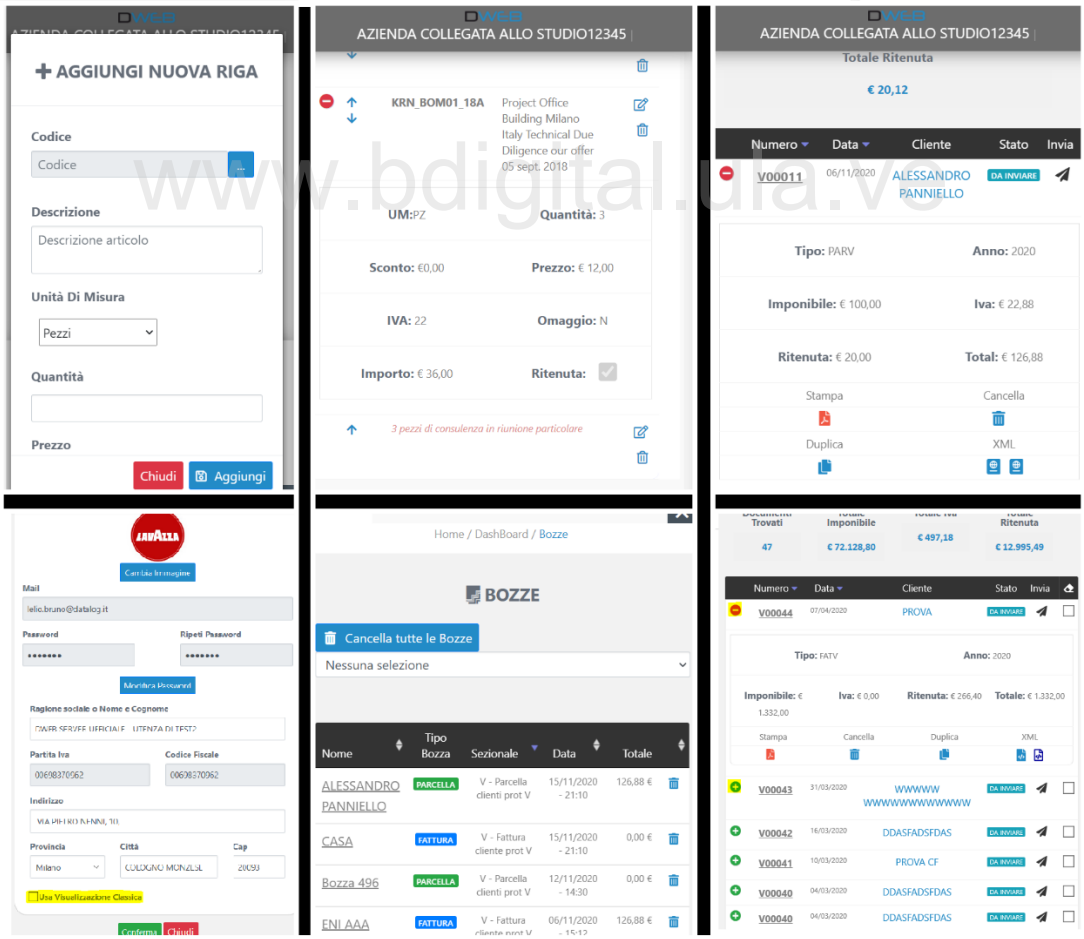


Figura 43: Capturas de Visualización Móvil

5.3.2 Implementación de la estructura funcional

En cuanto al desarrollo de la estructura funcional del sistema se eligió una metodología de desarrollo que permitiera el funcionamiento integrado de los subsistemas que conforman el sistema DWEB, además se eligió utilizar una estructura que permitiera la comunicación con los sistemas externos DHUB y KING como fue descrito en la fase de diseño del sistema.

Para lograr estos objetivos se realizó un tercer sistema externo API DWEB, este sistema tiene previsto el desarrollo de la comunicación entre las funciones del DWEB y los sistemas DHUB y KING, esto se logra a través del uso de una DLL compilada del programa KING para aprovechar sus funcionalidades internas y la comunicación con la base de datos para los registros, mientras que para el DHUB se diseñó una API que permite la comunicación con este servicio.

Por lo tanto, a nivel del sistema integrado DWEB se realizaron modelos de clases "servicio" las cuales se encargan de realizar las respectivas gestiones para establecer la comunicación de la forma:

DWEB ←→ API DWEB ←→ (KING/DHUB)

Esta Api fue desarrollada por los creadores de los sistemas KING y DHUB, por lo tanto, fueron ellos quienes pusieron a nuestra disposición los prototipos de las funciones necesarias para realizar la funcionalidad del sistema, así que, aunque no comprenden parte del desarrollo de este proyecto, es clave en la funcionalidad y la estructura del mismo.

Iniciando con los modelos se estableció una estructura de clases que permitía manejar los datos de forma más sencilla a través del concepto de la abstracción de datos y la programación orientada a objetos, este desarrollo permitió estructurar los datos de una forma simple y fácil de manejar, utilizando la serialización de las clases en formato Json.

Todas estas clases fueron implementadas en paralelo con las clases implementadas en el API DWEB y a su vez en los programas KING y DHUB.

Mientras tanto por su parte los controladores del sistema se encargaban de realizar la conexión entre la vista (mediante Jquery), los modelos y la API DWEB destinada a interactuar con los otros subsistemas propuestos.

Mediante el uso de todos los elementos descritos anteriormente se representa la estructura del framework de trabajo utilizado MVC (Modelo-Vista-Controlador) y se realizó la comunicación interna de los datos en nuestro sistema.

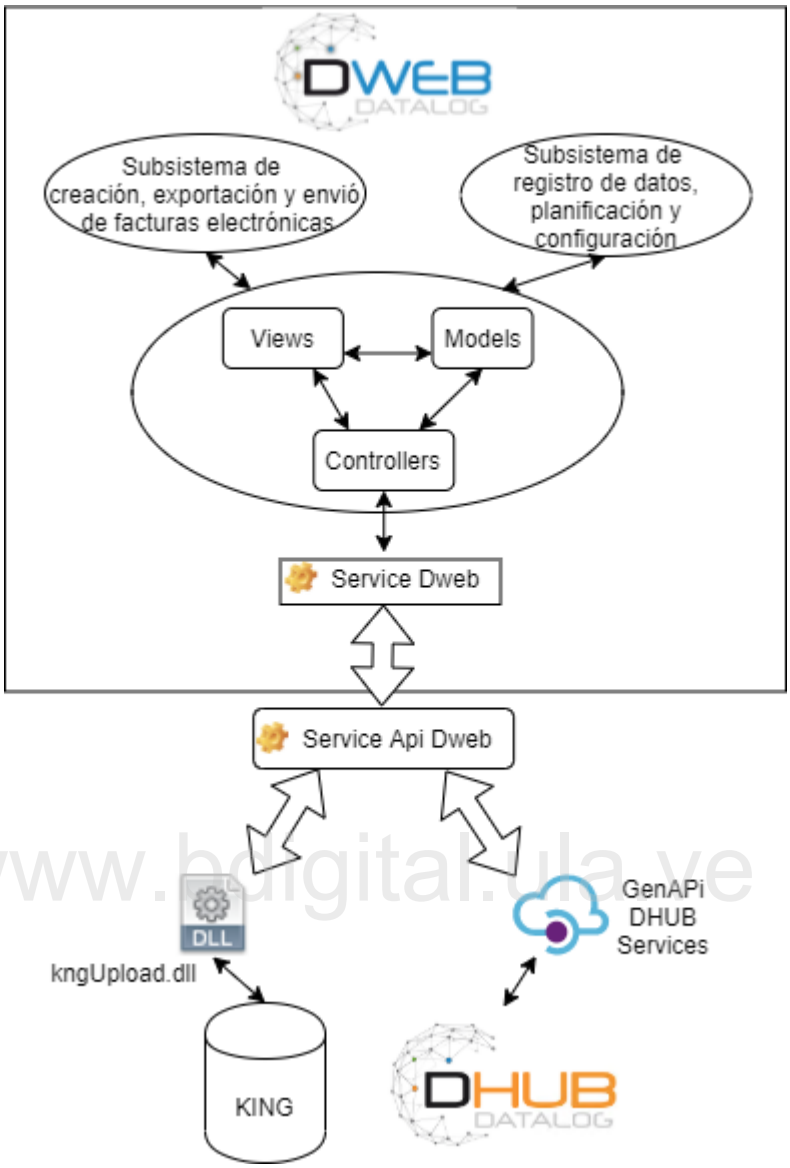


Figura 44: Representación gráfica de la estructura funcional

Para la seguridad entre las llamadas y conexión de las Api y las Dll se utilizó una estructura de tokens en cada llamada, las cuales utilizaban tokens de seguridad fijos del sistema para las funciones en las cuales el usuario no se encontraba autenticado dentro del sistema, mientras que cuando el usuario estaba autenticado al sistema se realizaba un token calculado en base al código del usuario, de forma de autenticar que usuario realizaba las peticiones a los servicios.

Adicionalmente para incrementar las medidas de seguridad se estableció una política de validación de datos a diversos niveles, partiendo del nivel de interfaz de usuario (front-end) usando Jquery, posteriormente los campos son validados a nivel del controlador (back-end), una vez pasados estos niveles los tokens de usuario o del sistema son calculados a nivel del servicio de comunicación

con los programas DHUB y KING donde, cada uno a su vez haciendo uso del token autorizaban el acceso a las plataformas.

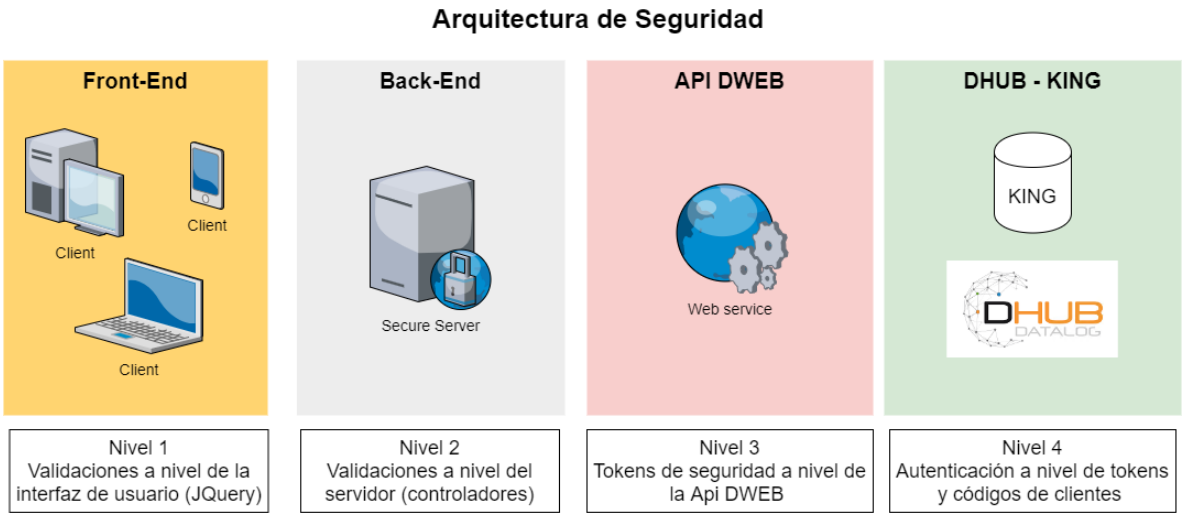


Figura 45: Arquitectura de seguridad del sistema

Con esta estructura como se puede apreciar en la imagen precedente se pueden garantizar al menos 4 niveles de seguridad en el procesamiento de los datos, implementados a partir de lógicas particulares a cada nivel, lo que implica que el manejo y seguridad de los datos esten garantizados y a su vez proporciona un nivel de confiabilidad del sistema alto.

Es importante tambien mencionar que como en el entorno de la facturacion electronica el manejo de la informacion es considerado como manejo de datos sensibles, esta arquitectura de seguridad proporciona ademas un manejo de los datos a partir de los tokens de seguridad y las tecnicas propias usadas en cada nivel, para realizar verificaciones a traves de los mismos a nivel de usuarios, garantizando de este modo que Datalog como garante de la privacy de cada uno de sus clientes realiza de forma correcta el manejo de los datos sensibles verificando en diversos niveles la identidad de los clientes en cada proceso del sistema antes de proporcionar acceso a las funciones o plataformas conectadas o de realizar la carga de cualquiera de estos datos.

CAPITULO 6

Conclusiones y Recomendaciones

Una vez terminado el desarrollo del subsistema realizado y de haber procedido con su integración en el entorno DWEB para la empresa Datalog Italia Srl, hemos tenido la oportunidad de observar durante un periodo de tiempo prolongado la calidad del producto proporcionado, esto nos permite realizar las conclusiones del proyecto con datos reales más allá de una proyección estimada, a continuación, podrá encontrar una descripción detallada de los resultados de nuestro producto.

6.1 Conclusiones

La factura electrónica es una innovación en diversos campos, es una aplicación del uso de los avances tecnológicos que, desarrollados correctamente, permiten avanzar en los procesos de modernización de las empresas, abarcando todas las ventajas explicadas en el capítulo 3 de este documento, cabe resaltar que una prueba clara de la modernización es que hoy en día, en Europa, así como en otras partes del mundo, es una parte fundamental de los procesos de auditoría ya que habilita la búsqueda y localización rápida de los documentos de soporte del análisis en cuestión.

Adicionalmente para la administración tributaria resulta muy importante que las empresas utilicen la facturación electrónica debido a que les facilita los mecanismos de control de cumplimiento tributario para evitar la evasión fiscal, tanto para revisar las operaciones del contribuyente como para verificar el cumplimiento de sus obligaciones impositivas que resultan indispensables para la integración de datos de todas las declaraciones informativas y de pago que se está obligado a presentar.

Las empresas que empiezan a operar con la facturación electrónica alcanzan un mayor grado de eficacia y eficiencia en diferentes áreas, ya que facilita la conciliación contable, a la vez ayuda a reducir los costos administrativos, ahorro de gastos de envío y de insumos, debido a que se disminuye el consumo de papel, cosa que adicionalmente fomenta la protección del medio ambiente.

El nivel de uso de la facturación electrónica en Italia hoy en día es alto y obligatorio por ley, por este motivo, se pueden apreciar las claras ventajas en comparación a la facturación tradicional.

El desarrollo y madurez de los sistemas que existen en el mercado es una clara respuesta de los empresarios del área tecnológica que cuentan con una disposición empresarial favorable que

permite avanzar en los procesos de modernización tanto en el sector público como privado, además de aperturar las opciones de innovación a través de la oferta de nuevos servicios.

Para Datalog Italia Srl como empresa, al ser proveedor de estos servicios a través del producto realizado, el crecimiento ha sido contundente, verificando los datos que presentaremos a continuación, podemos concluir que sencillamente, el desarrollo de este producto significa un antes y un después en la historia de la empresa:

1. Antes del desarrollo del sistema, la empresa Datalog Italia contaba con una cartera de clientes aproximada a un numero de 6.000 clientes, hoy en día cuenta con registros de 20.000 clientes de los cuales gran parte han ingresado gracias a la necesidad de realizar la facturación electrónica, pero además de esto por la oferta comercial que proporciona el sistema DWEB y sus servicios adicionales.
2. DWEB ha recibido diversas modificaciones, las cuales han estado motivadas por:
    a. Cambios en las normas de facturación (modelos, formatos, campos, cambios fiscales, entre otros).
    b. Iniciativa de la empresa Datalog Italia Srl de ofrecer nuevos servicios adicionales a sus clientes.
    c. Modificaciones requeridas por los clientes para la realización de personalizaciones en el sistema.
    d. Integración de otros sistemas existentes en la empresa Datalog al sistema DWEB además de los ya conocidos (DHUB y KING).
3. El proyecto hasta la fecha de hoy cuenta con una cantidad de once millones de facturas transitadas a través del sistema DWEB y otros sistemas conjuntos.

6.2 Aportes

Dentro de los aportes que proporciona la realización de este sistema, se presentan diversos aportes tangibles y no tangibles, para diversas entidades que participan directa e indirectamente en este producto.

Entre los aportes no tangibles podemos mencionar:

- Se presenta una experiencia exitosa de cómo abordar un problema de la ingeniería en todas sus etapas, definición del problema, análisis de requerimientos, diseño, documentación, implementación e integración.
- La aplicación práctica de los conocimientos adquiridos por parte de los integrantes del grupo de desarrollo, los cuales involucran diversas disciplinas de la ingeniería.

- La nueva adquisición de conocimientos obtenidos durante el desarrollo del producto como son la normativa fiscal, la normativa legal y el idioma italiano.
- La experiencia adquirida en el desarrollo de software y las herramientas del framework de desarrollo utilizado.
- Conocimientos avanzados en la integración de sistemas existentes con un sistema nuevo, demostrando que la premisa de reutilización de código, es una premisa útil y que además de evitar trabajo extra puede conllevar a la realización de nuevos sistemas complejos, facilitando su implementación, disponiendo de algunos procesos o funciones ya desarrollados, lo que deriva en una disminución del grado de complejidad del nuevo sistema.
- El reconocimiento del personal de la empresa con los conocimientos adquiridos en nuestra formación profesional y del alto nivel de enseñanza de nuestra universidad.

Entre los aportes tangibles se pueden mencionar:

- El producto desarrollado e implementado en el sistema integrado DWEB, el cual, a pesar de ser realizado bajo la filosofía de acuerdo de confidencialidad, deja el estudio realizado y las decisiones tomadas en cuenta, como una guía clara de los pasos a seguir, además de un gran marco teórico disponible al público para la implementación de este tipo de sistemas.
- La construcción de un sistema funcional, que sigue todas las especificaciones del diseño y que además hoy en día se mantiene en uso y bajo constantes actualizaciones.
- El manual de usuario del sistema, el cual facilita a los clientes de Datalog Italia Srl., una guía rápida con una explicación acerca del sistema y sus reglas principales, permitiendo que el producto sea de fácil uso.
- La apertura de nuevos puestos de trabajo dentro de la empresa Datalog Italia Srl., para la realización del proceso de asistencia y reporte de posibles problemas en el sistema.
- Un crecimiento importante a nivel de clientes, lo cual es una muestra clara de las ventajas y de resolución de problemas que proporciona el servicio del producto presentado a nivel de automatización de empresas.

El presente documento también puede ser considerado como un aporte tangible de este trabajo.

6.3 Recomendaciones

En este apartado se mencionan algunas recomendaciones y trabajos futuros propuestos que posteriormente ayudaron a consolidar y a mejorar el sistema DWEB como plataforma web que ofrece servicios de facturación electrónica.

- Realizar una limpieza y optimización del código que pueda ser centralizado o que pueda aumentar la velocidad en los procesos, debido al alto volumen de facturas que son generadas por el sistema.
- Seguir con la implementación de servicios adicionales, como la gestión de nuevos documentos fiscales que no son obligatorios, pero que sirven de apoyo a la facturación electrónica.
- Desarrollo de un nuevo subsistema del tipo BackOffice que puede permitir a usuarios especiales (internos en Datalog) gestiones de mensajes y alertas para los usuarios.
- Automatización de un Test Case para realizar las pruebas del sistema de forma automática, permitiendo no repetir al reparto de testing, el compilado de todos los formularios y la generación automática de facturas, de esta forma solo deberán validar que la factura sea generada correctamente.
- El diseño del marketing para la comercialización de planes adicionales (premium) que permiten acceso a ciertas funcionalidades del sistema, de esta forma los servicios no obligatorios podrán ser contratados por los usuarios que los requieran y no serán impuestos (ya que no son obligatorios fiscalmente).
- Registro de las acciones realizadas por el cliente obtenidas a través de las llamadas realizadas al sistema (Http Request) lo cual permite obtener datos importantes para la fase de resolución de errores (como pueden ser dispositivo, navegador, fecha, respuestas del sistema, entre otros) estos datos adicionalmente a un sistema de logs bien planificado podrá ayudar a la identificación y resolución de errores de una forma más eficiente.

Bibliografía

VILLEGAS, José. *Comprobantes fiscales digitales y facturación electrónica* [en línea]. Universidad Católica del Táchira, Revista Derecho y Tecnología Nº 15/2014 71-84.
Disponible en < http://bdigital.ula.ve/documento/30350>
[Consulta: 01 Septiembre 2019]

Microsoft Corporation. *Introducing Visual Studio 6* [en línea]. Microsoft Visual Studio, 1997.
Disponible en < https://www.microsoft.com/msj/0597/visualstudio97.aspx >
[Consulta: 01 Septiembre 2019]

Microsoft Corporation. *Microsoft Visual Studio 16.1.6* [en línea]. Microsoft Visual Studio, 2019.
Disponible en < https://visualstudio.microsoft.com/es/ >
[Consulta: 01 Septiembre 2019]

NONAKA & TAKEUCHI, *The New New Product Development Game. SCRUM* [en línea]. 1986.
Disponible en < https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software) >
[Consulta: 01 Octubre 2019]

SCHWABER, Ken. *Scrum Development Process* [en línea]. OOPSLA 95 (Object-Oriented Programming Systems & Applications conference),1995.
Disponible en < https://www.scrum.org/about >
[Consulta: 06 Noviembre 2019]

Proyectos Ágiles. Que es SCRUM [en línea] 4 de agosto de 2008
Disponible en <https://proyectosagiles.org/que-es-scrum/ >
[Consulta: 06 Noviembre 2019]

SCHWABER, Ken. *Agile Project Management with Scrum*. [en línea] 163pp, ISBN 0-7356-1993-X, 2004 and *Scrum et al,* Youtube GoogleTechTalks, 2016.
Disponible en
<https://www.youtube.com/watch?v=IyNPeTn8fpo >
[Consulta: 12 Noviembre 2019]

Ole-Johan Dahl and Kristen Nygaard, Simula Software Development, [en línea] IBM System 360/370 Compiler and Historical Documentation, 1967
Disponible en
<https://web.archive.org/web/20171011181559/http://edelweb.de/Simula/>
[Consulta: 12 Noviembre 2019]

Senato della Repubblica, Legge di Bilancio 2018. [en línea] Pubblicato in Gazzetta Ufficiale n. 302 (Art 1 – 411, 909,920,995,1011,1012,1026,1028,1033,1123) del 29 de diciembre del 2017.
Disponible en
<http://www.gazzettaufficiale.it/do/gazzetta/downloadPdf?dataPubblicazioneGazzetta=20171229&numeroGazzetta=302&tipoSerie=SG&tipoSupplemento=SO&numeroSupplemento=62&estensione=pdf&edizione=0 >
[Consulta: 11 Diciembre 2019]

COCKBURN, Alistair. *Writing Effective Use Cases*. [en línea]
Disponible en < https://www.infor.uva.es/~mlaguna/is1/materiales/BookDraft1.pdf >
[Consulta: 03 Marzo 2020]