Diseño OCR para la gestión de impagos en el ámbito empresarial

Jorge Gomila Álvarez
Grado en Ingeniería Informática

Xavier Escudero Sabadell

06/2022

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Diseño OCR para la gestión de impagos en el ámbito empresarial*
Nombre del autor:	*Jorge Gomila Álvarez*
Nombre del consultor:	*Xavier Escudero Sabadell*
Fecha de entrega (mm/aaaa):	*06/2022*
Área del Trabajo Final:	*Ingeniería del Software*
Titulación:	*Grado en Ingeniería Informática*

Resumen del Trabajo (máximo 250 palabras):

Este trabajo tiene el objetivo de diseñar una aplicación de sobremesa para la gestión de devoluciones de cobros para cualquier PYME. Por propia necesidad en el ámbito profesional, la dedicación exigida en tareas administrativas tras devoluciones de cobros es elevada. Por consiguiente, para minimizar tanto el coste económico como temporal, el diseño de dicha aplicación pretende ahorrar la introducción manual de los datos necesarios para registrar la devolución de una factura para gestionar su recobro. Para lograrlo, la aplicación se ha diseñado con funciones OCR, como lector inteligente, para que a partir de una imagen se capturen los caracteres y de forma automática se pueda realizar la introducción de los datos imprescindibles para dicha gestión. Por tanto, el trabajo se centra en el diseño de una API para ser utilizada como complemento de otro software.

El trabajo engloba el diseño de la arquitectura, la base de datos, así como el funcionamiento esperado a partir de una serie de requisitos. Mediante el lenguaje UML se crean diagramas de actividad, diagramas de clases. La tecnología elegida para la implementación es .NET y desarrollada bajo el paradigma de la programación orientada a objetos (POO). En principio, su funcionamiento está pensado para que funcione en la plataforma Windows. Se muestran prototipos de la interfaz gráfica aunque no se incluye la implementación pues no se encuentra entre los objetivos de este trabajo.

Abstract (in English, 250 words or less):

This work has the objective of designing a desktop application for the management of collection returns for any PYME. By own necessity in the professional field, the dedication required in administrative tasks after collection returns is high. Therefore, to minimize both the economic and time cost, the design of this application aims to save the manual input of the necessary data to record the return of an invoice to manage its recovery. To achieve this, the application has been designed with OCR functions, such as an intelligent reader, so that the characters are captured from an image and the essential data for said management can be entered automatically. So, the work focuses on the design of an API to be used as a complement to other software.

The work encompasses the design of the architecture, the database, as well as the expected operation based on a series of requirements. Using the UML language, activity diagrams and class diagrams are created. The technology chosen for the implementation is .NET and developed under the object-oriented programming (POO) paradigm. In principle, its operation is designed to work on the Windows platform. Prototypes of the graphical interface are shown, although the implementation is not included, since it is not among the objectives of this work.

Palabras clave (entre 4 i 8):
OCR, gestión de impagos, lector inteligente.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

El amplio uso de los sistemas informáticos ha llevado a que la gran mayoría de la información que recibimos, el inmenso bombardeo constante de información, esté digitalizado y lo recibamos vía dispositivo electrónico. Por ello, para poder hacer un procesamiento de los datos según nuestras necesidades, podemos recurrir a automatizar la captura de los mismos mediante el uso de software OCR[1] (reconocimiento óptico de caracteres).

Con tanto volumen de información y exigencias en el mundo empresarial a nivel de tiempo, conlleva relegar algunas gestiones importantes a un segundo plano con las posibles pérdidas económicas que pudieran generar. Debido a las reducidas estructuras empresariales como al reducido volumen de determinadas gestiones en las PYME[2], hacen inapropiado el contar con personal con una dedicación exclusiva a ciertas tareas administrativas (por ejemplo, la gestión de impagos).

La idea de la automatización parte del objetivo de minimizar el coste que genera la intervención humana, tanto en coste económico como temporal. Desde una perspectiva personal, la motivación para realizar esta temática se debe a la necesidad en la propia vida laboral de disponer de una herramienta de dichas características. Los conocimientos adquiridos a lo largo de la carrera me han permitido obtener la suficiente base para poder llevar pequeños proyectos que ante todo, sean útiles en mí día a día. Espero poder mejorar el diseño del software en cuestión para que pueda en el futuro llevarlo a cabo.

1.2 Objetivos del Trabajo

En el desarrollo del software enfocado al mundo empresarial se desea realizar el diseño y análisis de un software capaz de capturar texto susceptible de usar en la gestión de cobros de una empresa. La información recibida por vía electrónica, mediante archivos PDF[3], hace necesaria la intervención humana. El objetivo es lograr minimizar el tiempo/recursos utilizados en las tareas administrativas de recopilación de datos y gestión de impagos a partir de las comunicaciones enviadas por la entidad bancaria. Para lograrlo mediante un OCR y su posterior tratamiento, se pretende diseñar una API[4] capaz de reportar y comunicar los datos necesarios para integrarlos en la aplicación propia de la empresa, permitiendo automatizar el proceso.

En base a lo anterior, se pueden extraer una serie de necesidades a cubrir:

- Ofrecer de una solución para la gestión de impagos resultado de devoluciones de recibos domiciliados.

- Diseñar una aplicación de aspecto sencillo pero a la vez, potente en su funcionalidad.
- En base a los conocimientos adquiridos en el Grado de Ingeniería Informática, analizar y diseñar con la tecnología más idónea.

1.3 Enfoque y método seguido

A partir de los datos facilitados por la entidad bancaria, se deberá obtener la información necesaria para poder procesarla para llevar a cabo la gestión de impagados y su posterior recobro.

Teniendo en cuenta el uso de un OCR y por tanto, la necesidad del reconocimiento de caracteres, es crucial que la tasa de error en el mismo sea mínima y se logre la mayor exactitud posible para minimizar así mismo, la intervención humana, logrando reducir al máximo el coste de un departamento que a fin de cuentas, no reporta ningún valor añadido al cliente final pero que resulta necesario en cualquier organización empresarial.

Se incluirá en el proyecto software de terceros. La motivación de dicha decisión es la minimizar el coste de implementación, garantizando además una alta calidad gracias al estudio e investigación de otros desarrolladores en el campo de los OCR inteligentes. El software que se incluirá, de todas las alternativas posibles, es el conocido Tesseract[5], un motor de reconocimiento óptico de caracteres para varios sistemas operativos. Se trata de un software libre, liberado bajo la licencia Apache[6].

Por tanto, aunque se pretende diseñar un producto nuevo, se hará uso del software Tesseract como "librería" o motor para la captura de los caracteres. Para el tratamiento de la información se diseñará una interfaz de usuario.

Para lograr dichos objetivos, en el desarrollo se utilizará una metodología ágil [7] frente a una clásica o tradicional. El motivo de esta decisión se basa en la clasificación del proyecto según Wysocki (2009), que debido a que el objetivo a alcanzar resulta claro y la solución es poco conocida, puesto que no conocemos todos los detalles de la solución, es posible que se deba corregir ciertos diseños por que se necesita cierta flexibilidad en el proceso de desarrollo. Además, la metodología ágil permite un desarrollo iterativo e incremental y repercute positivamente al producto final que se obtendrá. Dentro de la metodología ágil, pienso que Scrum[8] podría ser el método utilizado puesto que es ligero y minimiza al máximo lo innecesario, centrándose únicamente en aquello que aporta un valor importante.

1.4 Planificación del Trabajo

Las tareas que se deberán realizar a lo largo del proyecto están relacionadas con el análisis y diseño de la aplicación. Para una correcta progresión se han establecido 3 entregas a lo largo del curso lectivo y que corresponden con la PAC1, PAC2 y PAC3. Finalmente, se tendrá que

desarrollar una memoria del proyecto así como una presentación. A continuación se muestra una planificación:

Planificación y Diagrama de Gantt

	Días	Fecha de inicio	Fecha límite
PAC1 - Alcance y planificación	**15**	**16/02/2022**	**01/03/2022**
Alcance	2	16/02/2022	18/02/2022
Objetivos	3	18/02/2022	21/02/2022
Decisión metodología de desarrollo	1	21/02/2022	22/02/2022
Planificación proyecto	3	22/02/2022	25/02/2022
Entrega PAC1	**5**	**26/02/2022**	**01/03/2022**
PAC2 - Análisis	**29**	**02/03/2022**	**01/04/2022**
Definición de requisitos	4	02/03/2022	06/03/2022
Casos de uso	3	07/03/2022	10/03/2022
Identificación de actores	4	11/03/2022	15/03/2022
Clases	7	16/03/2022	23/03/2022
Otras tareas de Análisis	5	24/03/2022	29/03/2022
Entrega PAC2	**1**	**30/03/2022**	**01/04/2022**
PAC3 - Diseño	**51**	**02/04/2022**	**23/05/2022**
Modelado UML	18	02/04/2022	20/04/2022
Modelado casos de uso	9	21/04/2022	30/04/2022
Modelado Interfaz	19	01/05/2022	20/05/2022
Entrega PAC3	**2**	**21/05/2022**	**23/05/2022**
Memoria y presentación	**15**	**24/05/2022**	**09/06/2022**
Revisión y desarrollo memoria final	4	25/05/2022	29/05/2022
Preparación presentación y exposición final	**9**	**30/05/2022**	**09/06/2022**

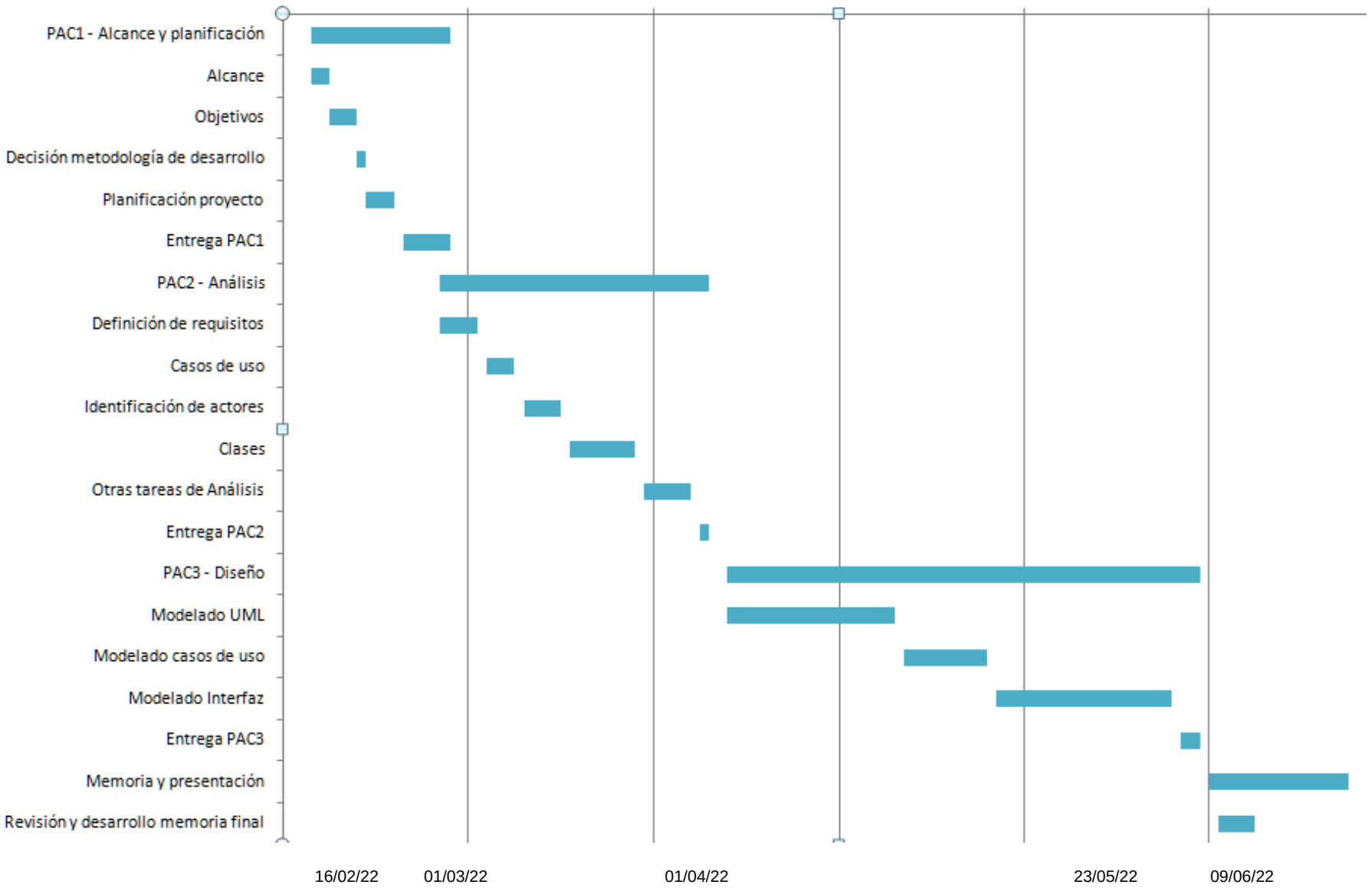


Ilustración 1. Diagrama de Gantt

1.5 Breve resumen de productos obtenidos

Se desea obtener el diseño de una aplicación enfocada al mundo empresarial, capaz de realizar la captura de caracteres susceptibles de ser utilizados en el proceso de la gestión de impagos.

A priori, aunque se diseñará con unas características concretas (con un software concreto y conocido), para poder ser utilizado de forma genérica se buscará que pueda ser adaptado al otro software por medio del desarrollo de una API.

1.6 Breve descripción de los otros capítulos de la memoria

En un siguiente capítulo, se desea poder desarrollar más ampliamente los requisitos, mediante la identificación de los mismos y la forma en que pueden obtenerse. Para ello, se realizará una posible documentación de los requisitos necesarios para poder completar el diseño, analizando los diferentes casos de uso, los actores y clases necesarias.

En un siguiente capítulo, se llevará a cabo el diseño propiamente dicho de la aplicación, haciendo uso de UML[9]. A partir de este, se buscará modelar en la medida de lo posible los casos de uso principales y un modelado de la interfaz.

En ambos casos, las diferentes tareas a realizar en el desarrollo de este software se relacionan con las primeras fases necesarias para el diseño de una aplicación y resulta imprescindible para poder obtener el producto final según los requisitos obtenidos de los stakeholders[10] o personas interesadas en el proyecto.

2. Requisitos

2.1 Requisitos del software

La aplicación que se pretende desarrollar será del tipo cliente-servidor, ubicada en local o en una intranet. Su implementación se llevará a cabo mediante el paradigma de POO[11] (Programación orientada a objetos) en el lenguaje Visual.Net.

Para el desarrollo de una forma ordenada, la arquitectura se basará en una arquitectura de 2 o 3 capas. La capa persistente, es decir, la base de datos donde se almacenarán los datos será PostgreSQL[12] también llamado Postgres. Se trata de un sistema de gestión de bases de datos relacional orientado a objetos y de código abierto, publicado bajo la licencia PostgreSQL.

Para la obtención de los requisitos se llevan a cabo entrevistas y reuniones con el cliente para determinar y recoger de forma detallada las funcionalidades que tendrá el programa. Para el correcto desarrollo posterior se detallan de forma minuciosa para que existan las mínimas diferencias entre el diseño y el producto final.

Los requisitos son características observables de un sistema que expresan una necesidad o restricción que un stakeholder tiene sobre él. Nos permiten delimitar las posibles soluciones a un problema y si son adecuadas.

En este apartado nos centraremos en la obtención, gestión y documentación de los requisitos. La validación como la verificación no entra dentro de los objetivos de este análisis. Entre las diferentes categorías de los requisitos, nos centraremos en los requisitos funcionales y no funcionales.

2.2 Identificación de stakeholders

Un stakeholder sería el cliente o empresa receptora del software y que desea satisfacer unas necesidades que cubren el software propuesto. Como principal interés es disponer de una herramienta sencilla pero a la vez potente que le permita llevar una correcta gestión de los impagados al menor coste posible a partir de la información que hace disponible la entidad bancaria.

Teniendo en cuenta que la aplicación tiene que cumplir con una serie de parámetros legales y de seguridad, otro posible stakeholder serían las administraciones públicas. En especial, la aplicación debe cumplir con la ley en los términos de la LOPD[13], en la protección de datos personales. Además, también debe cumplir los términos para ser considerado un software que cumple con los términos de la nueva ley contra el fraude fiscal 11/2021, de 9 de julio, de medidas de prevención y lucha contra el fraude fiscal.

2.3 Descripción del proceso

Ante una devolución bancaria, el usuario descarga desde la web de la entidad, información detallada de la devolución, normalmente generada en formato PDF. A continuación, al arrancar el software se realiza de forma automática una importación con los registros de las facturas, en el caso de que se disponga de la estructura específica de datos o bien a partir de un fichero que los contenga en formato Excel. Seguidamente, se podría realizar la selección del fichero de devoluciones, para la captura mediante OCR, aunque también la aplicación dispondrá de la opción para realizarlo manualmente.

2.4 Funciones principales

Entre la funcionalidad de la aplicación, las principales son las que se especifican a continuación:

- Importación de datos de facturas.
- Selección de facturas devueltas.
- Generación de remesa de devoluciones.
- Captura de devoluciones mediante OCR.
- Envío automático de email de aviso de impago.
- Enlace contable de devoluciones y recobro.

2.5 Requisitos funcionales

Los requisitos funcionales son aquellos que hacen referencia a la funcionalidad del sistema y los datos que tiene que conocer y guardar. Por ello, se distinguen en esta categoría, los de funcionalidad y de datos.

Acceso	**Gestión de acceso e identificación de usuarios.**
Mantenimiento	Gestión de facturas. Gestión de usuarios. Gestión de devoluciones y recobros.
Comunicaciones	Gestión de comunicaciones vía email.
Capturas	Gestión de capturas vía OCR.

2.5.1 Funcionalidad

Describen el comportamiento esperado del sistema (respuesta observable) ante los estímulos que llegan del exterior. Algunos de estos requisitos aplicables al software que se está diseñando serían:

“Como usuario quiero poder visualizar una lista completa de las facturas”.

“Como usuario quiero vincular un fichero de devoluciones con la correspondiente factura registrada”.

2.5.2 De datos

Los requisitos funcionales de datos describen qué datos tiene que conocer el sistema.

“El sistema tiene que conocer el número de cliente, su nombre o razón social, número de factura, fecha e importe total”.

“El sistema tiene que conocer el estado actual de las facturas”.

2.6 Requisitos no funcionales

Son aquellos que implican cualidades esperadas del sistema, como la usabilidad, fiabilidad, rendimiento o mantenibilidad. Están relacionados con la calidad del sistema y que lo convierten en requisitos que mejoran la interacción del usuario y aportan el cumplimiento de unas características concretas, como son legales, de seguridad, etc.

Entre los diferentes requisitos que pudieran diferenciarse, se han obtenido los siguientes:

Requisito	**Plataforma. La aplicación será desarrollada para funcionar en el sistema operativo Windows.**
Descripción	La aplicación correrá en el sistema operativo Windows.
Tipo	Usabilidad, operacionales y entorno.
Criterios de aceptación	Se hará una prueba para comprobar que el sistema funciona en Windows.
Stakeholders	Cliente.

Requisito	**La aplicación tiene que ser sencilla de utilizar.**
Descripción	El sistema tiene que ser sencilla de usar para minimizar tanto el coste de aprendizaje como su futuro mantenimiento.
Tipo	Usabilidad, mantenimiento, soporte.
Criterios de	Se hará una prueba con un usuario sin previa experiencia en

aceptación	la aplicación para poder determinar el grado de aprendizaje sin ningún tipo de formación.
Stakeholders	Cliente.

Requisito	**Tiene que cumplir con los términos legales y de seguridad mínimos.**
Descripción	El sistema tiene que cumplir con las leyes de protección de datos actuales y deben estar protegidos contra accesos no autorizados.
Tipo	Seguridad y legales.
Criterios de aceptación	Se realizarán diversas pruebas para comprobar el nivel de protección y seguridad, como la protección de accesos no autorizados a las bases de datos.
Stakeholders	Administración pública.

Requisito	**Entorno intuitivo y amigable.**
Descripción	El sistema tiene que resultar cómodo a la vista, fácil de usar e intuitivo.
Tipo	Presentación, usabilidad y humanidad.
Criterios de aceptación	Se hará una prueba con un usuario sin previa experiencia y se le pedirá una valoración sobre los aspectos mencionados (comodidad, facilidad y uso intuitivo).
Stakeholders	Cliente.

2.7 Identificación de los actores

A diferencia de los stakeholders, los actores participan en el comportamiento. Por eso, mientras que los stakeholders participan en el caso de uso mediante sus intereses, los actores interactúan con el sistema, entendiéndose para tal fin como el conjunto de roles que una persona/organización puede tener en relación el caso de uso.

Entre estos, podríamos destacar principalmente:

- Administrador del sistema.

- Usuario final.
- Empresario.

2.7.1 Descripción

Administrador. Se trata del usuario que gestiona la aplicación. Puesto que debe controlar todas las operativas, tiene acceso a la totalidad de las funcionalidades. Así mismo, es el encargado de gestionar a los usuarios que se les permite el acceso.

Entre sus funciones se encuentran:

- Dar acceso a los usuarios que da de alta y la asignación como usuario final.
- Modificación y baja de usuarios.
- Revisión/mantenimiento de los datos.
- Operativa habitual de la gestión de impagados.
- Captura OCR.

Usuario final. Se trata del usuario que por excelencia hará uso de las funcionalidades generales de la aplicación. Por un aspecto de seguridad, los usuarios tendrán que loguearse/identificarse para acceder y realizar una trazabilidad de las operaciones que realicen.

Entre las funcionalidades generales se encuentran:

- La importación de los registros de facturas.
- La selección manual de los registros de impago.
- La captura de ficheros externos para la captura OCR.
- La exportación para la vinculación con los ficheros externos de contabilidad (generación de ficheros de exportación).
- Gestión de impagados y recobro.
- Envío de emails.

Usuario superior. Se trata de un usuario de revisión de las operativas realizadas, pudiendo tener alguna operativa especial para facilitar la comprobación de la correcta gestión de los impagados llevada a cabo por los usuarios finales.

2.8 Identificación de los casos de uso

Se enumeran diferentes casos de uso, siendo una descripción de un conjunto de secuencias de acciones, incluyendo variantes, que ejecuta un sistema para producir un resultado observable y de valor para un actor *(VVAA (2010) Unified Modeling Language (UML). Object Management Group).*

Recoge el contrato entre el sistema y los stakeholders mediante la descripción del comportamiento observable del sistema. Para su modelización se hará uso del lenguaje UML, ampliamente utilizado.

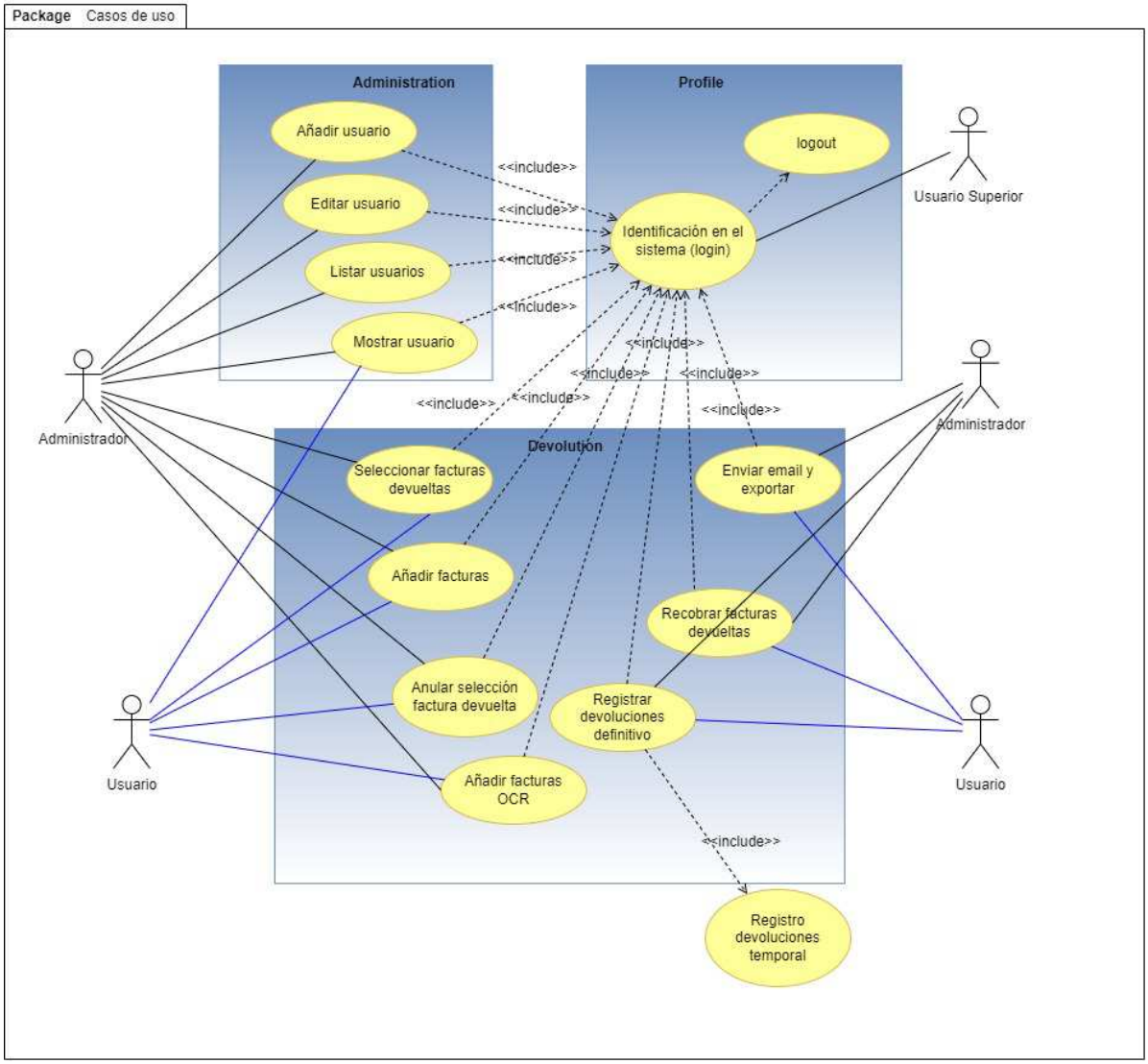


Ilustración 2. Casos de uso

CU001: Acceder a la aplicación

Actor principal: Usuario.
Ámbito: Pantalla principal.
Nivel de objetivo: General.
Stakeholders e intereses:
Administrador: quiere acceder.
Usuario final: quiere acceder.
Usuario Superior: quiere acceder.

Precondición: El usuario se debe haber identificado en el sistema.

Garantías mínimas: El sistema grabará el intento de entrada.

Garantías en caso de éxito: El sistema permite el acceso a las diferentes funcionalidades.

Escenario principal de éxito:

1. El usuario accede mediante sus datos registrados.
2. El sistema comprueba y graba el acceso, validándolo.
3. El sistema muestra la pantalla principal dándole al usuario el acceso al resto de funcionalidades.

Extensiones:

1.a El usuario no introduce correctamente sus datos.

1.a.1 Se borran los datos introducidos y se muestra un mensaje de error, permitiendo introducir de nuevo sus credenciales.

1.b El usuario cancela el acceso.

1.b.1 El sistema cancela la operación y cierra la aplicación.

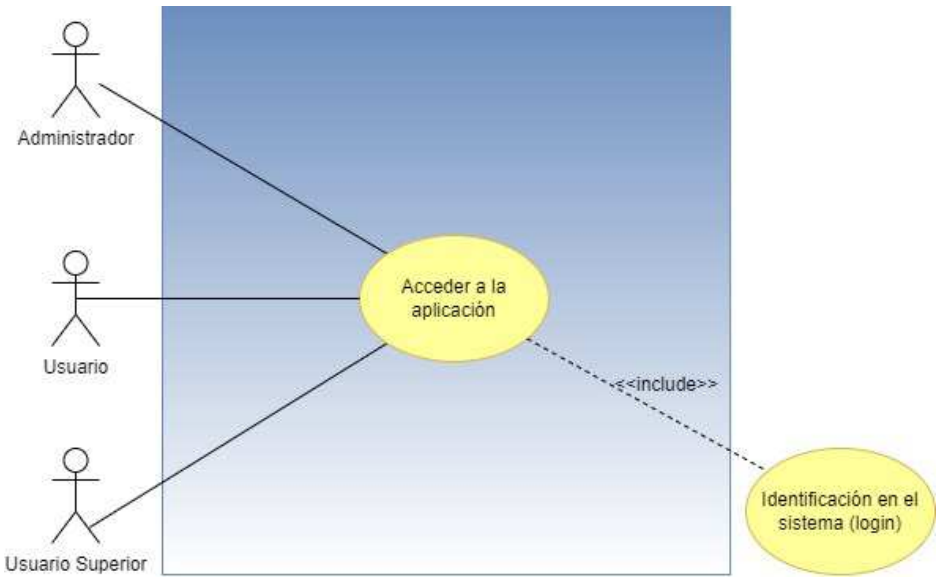


Ilustración 3. Caso de uso *Acceder a la aplicación*.

CU002: Añadir registros facturas

Actor principal: Usuario final.
Ámbito: Pantalla principal.
Nivel de objetivo: General.
Stakeholders e intereses:

Usuario final: quiere añadir los registros de facturas de ficheros externos.

Administrador: quiere añadir los registros de facturas para comprobar si están en su totalidad.

Precondición: El usuario se debe haber identificado en el sistema.

Garantías mínimas: El sistema revisará el último número existente añadiendo el resto de facturas, registrando finalmente la factura final añadida al sistema.

Garantías en caso de éxito: El sistema mostrará todas las facturas que se extraigan de los ficheros de importación.

Escenario principal de éxito:

1. El usuario pulsa en el botón que actualiza el registro de facturas.
2. El sistema valida la última factura registrada y si existen nuevos registros pendientes de añadir.
3. El sistema registra las facturas pendientes, posteriores al último registro.

Extensiones:

3.a El sistema no tiene ninguna factura registrada.
1.a.1 El sistema registra todas las facturas provenientes del fichero a importar.

3.b El sistema está completamente actualizado con los últimos registros.
3.b.1 El sistema no agrega ningún nuevo registro.

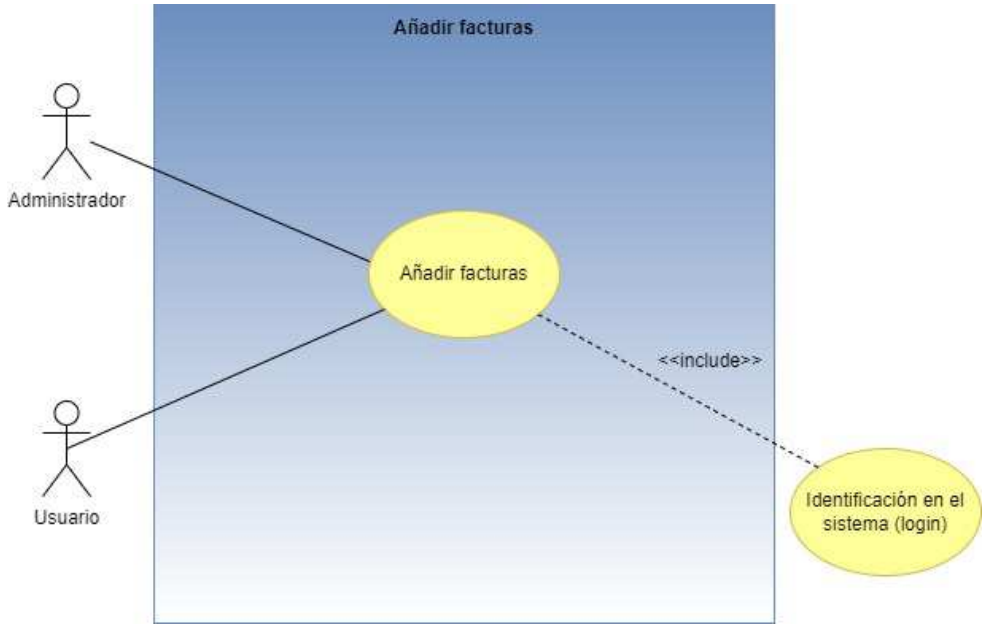


Ilustración 4. Caso de uso *Añadir facturas.*

CU003: Seleccionar facturas devueltas

Actor principal: Usuario final.
Ámbito: Pantalla principal.
Nivel de objetivo: General.
Stakeholders e intereses:

Usuario final: quiere registrar nuevas facturas devueltas.

Administrador: quiere actualizar el estado actual para corregir las devoluciones existentes.

Precondición: El usuario se debe haber identificado en el sistema.

Garantías mínimas: En el caso de existir una devolución, el sistema la añade a una lista para su registro.

Garantías en caso de éxito: El sistema muestra la selección en una lista temporal para su registro definitivo.

Escenario principal de éxito:

1. El usuario selecciona un registro, indicando también la fecha de devolución.
2. El sistema agrega un nuevo registro de la factura seleccionada a la lista de devoluciones temporal.

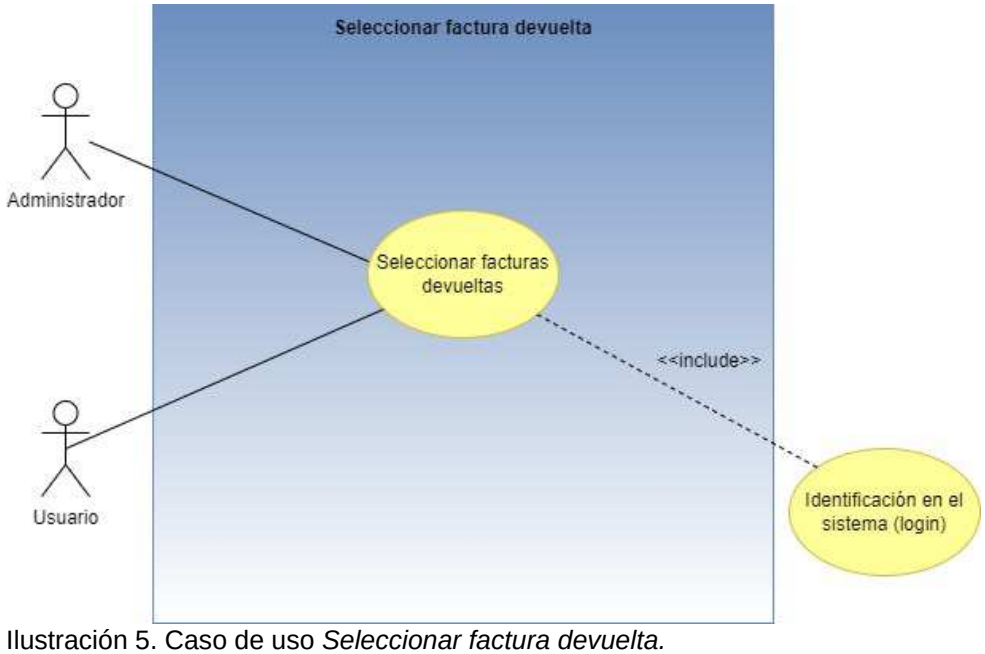


Ilustración 5. Caso de uso *Seleccionar factura devuelta.*

CU004: Retroceso-Anular factura devuelta

Actor principal: Usuario final.
Ámbito: Pantalla principal.
Nivel de objetivo: General.
Stakeholders e intereses:

Usuario final: quiere registrar anular un registro de facturas devueltas de la lista temporal.

Administrador: quiere actualizar el estado actual para corregir las devoluciones existentes.

Precondición: El usuario se debe haber identificado en el sistema y deben existir facturas devueltas en la lista temporal.

Garantías mínimas: En el caso de existir una devolución, el sistema la elimina de la lista temporal.

Garantías en caso de éxito: El sistema muestra elimina la selección de la lista temporal y cambia el estado al original.

Escenario principal de éxito:

1. El usuario selecciona una factura devuelta de la lista temporal.
2. El sistema elimina el registro seleccionado de la lista de devoluciones temporal.

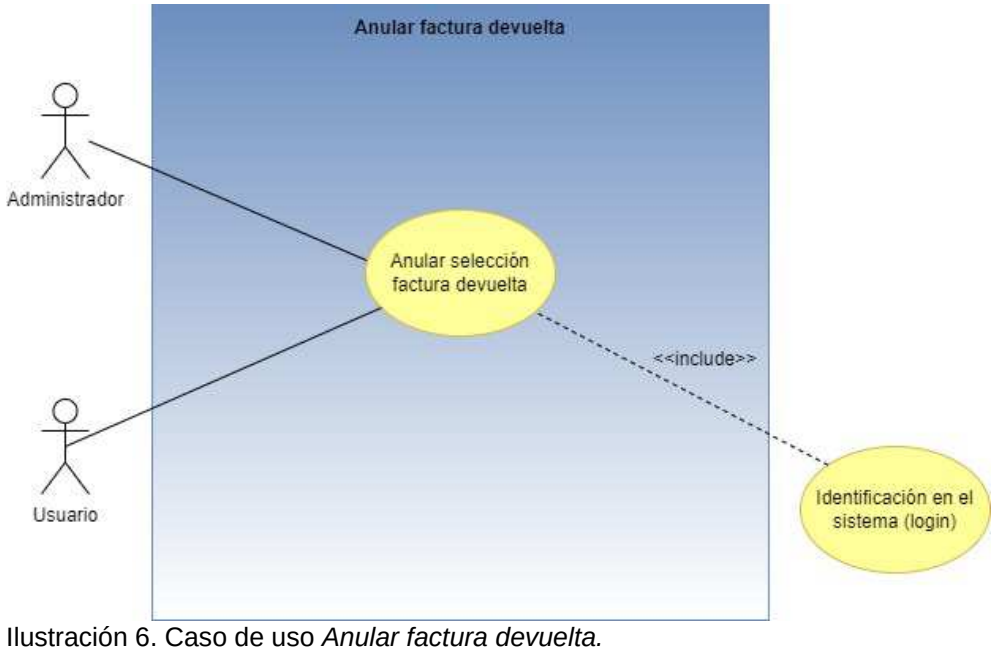


Ilustración 6. Caso de uso *Anular factura devuelta.*

CU005: Añadir devoluciones mediante OCR

Actor principal: Usuario final.
Ámbito: Pantalla secundaria.
Nivel de objetivo: General.
Stakeholders e intereses:

Usuario final: quiere registrar nuevas facturas devueltas mediante la lectura OCR a partir de un documento externo.

Administrador: quiere comprobar la correcta lectura de un fichero mediante OCR.

Precondición: El usuario debe haberse identificado en el sistema. El fichero externo puede estar en formato PDF o de imagen (JPG, TIFF, …).

Garantías mínimas: El sistema realiza la lectura del fichero externo, capturando diferentes parámetros.

Garantías en caso de éxito: El sistema revisará la correcta lectura de los datos del fichero y si cumplen con los parámetros exigidos, con unos campos mínimos según la estructura de datos necesaria.

Escenario principal de éxito:

1. El usuario pulsa en el botón que realiza la captura OCR, seleccionando el fichero a leer.
2. El sistema realiza la lectura capturando los datos e incorporándolos en los campos requeridos.
3. El usuario acepta los datos encontrados y pulsa el botón que permite la exportación.
4. El sistema importa a la pantalla principal la lista de devoluciones y los coloca en la lista temporal.

Extensiones:

2.a El sistema captura datos de forma incompleta o inexacta.
2.a.1 El sistema no procesa los datos y permite realizar una nueva captura de datos o modificarlos manualmente.
2.a.2 El usuario modifica y realiza una nueva captura.
2.a.3 El sistema captura correctamente los datos.

2.b El usuario cancela la captura OCR.
3.b.1 El sistema se dirige a la pantalla principal de la aplicación.

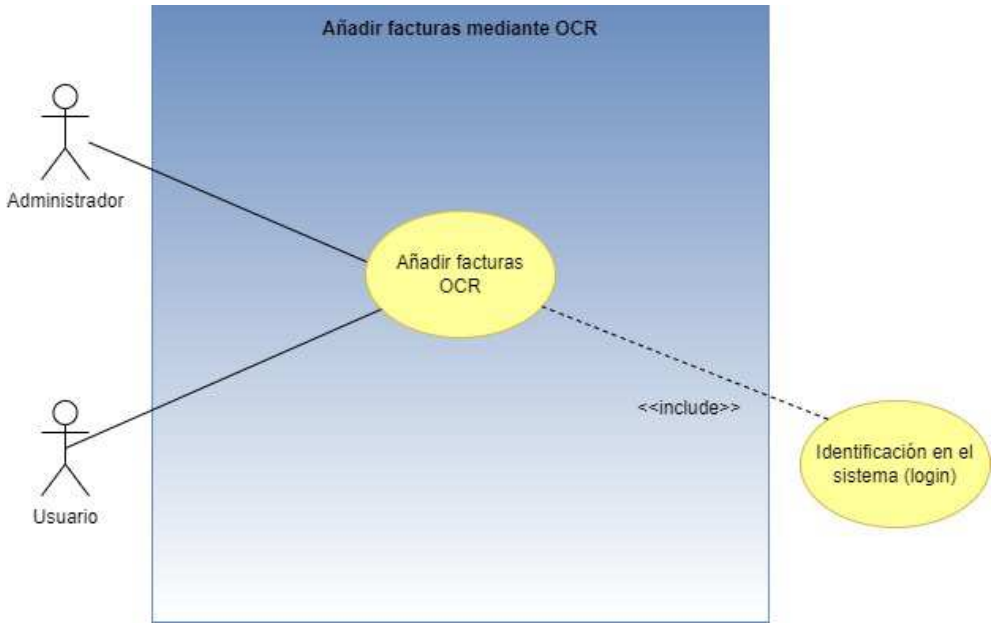


Ilustración 7. Caso de uso *Añadir facturas mediante OCR.*

CU006: Registro definitivo de devoluciones

Actor principal: Usuario final.
Ámbito: Pantalla principal.
Nivel de objetivo: General.
Stakeholders e intereses:

Usuario: quiere registrar de forma definitiva el conjunto de facturas que se encuentran en la lista temporal de devoluciones.

Precondición: El usuario debe haber registrado las devoluciones de facturas en la lista temporal.

Garantías mínimas: El sistema ha registrado en la lista temporal las facturas devueltas y es posible generar la devolución de forma definitiva. El botón está activo para realizar el proceso.

Garantías en caso de éxito: El sistema mostrará la lista de devoluciones temporal que tras pulsar el botón de registro definitivo eliminará del conjunto de facturas y quedarán registradas con un cambio de estado “devuelta”.

Escenario principal de éxito:

1. El usuario pulsa en el botón que realiza el registro definitivo.
2. El sistema escribe dentro de una tabla de devoluciones las nuevas facturas devueltas y cambia el estado de las facturas a “devuelta”.

Extensiones:

1.a El sistema cancela la operación.

1.a.1 El sistema no procesa los datos y los registros quedan en la lista temporal de devoluciones.

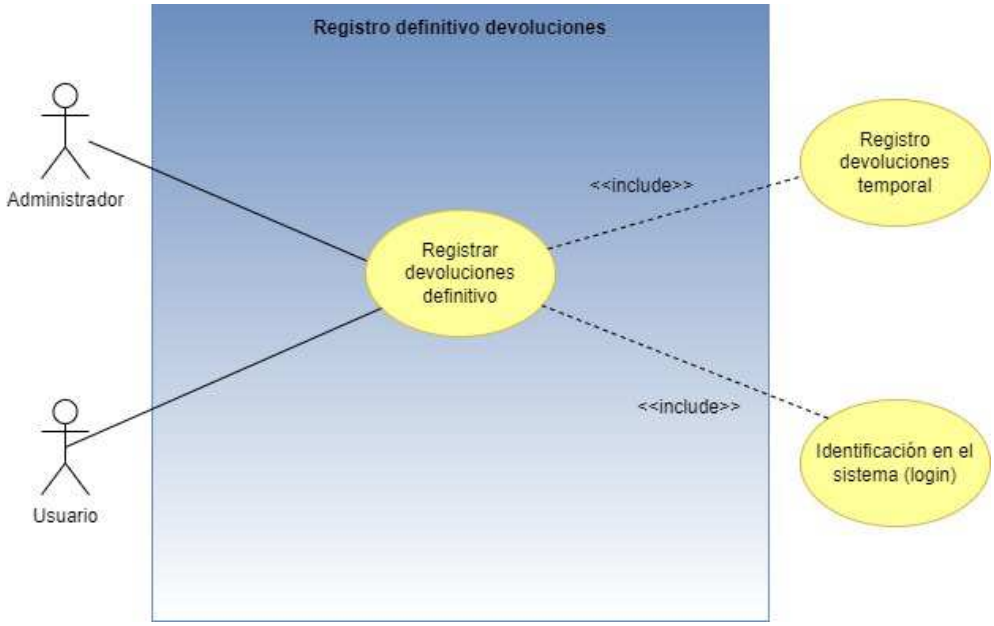


Ilustración 8. Caso de uso *Registro definitivo devoluciones.*

CU007: Recobro de facturas

Actor principal: Usuario final.
Ámbito: Pantalla secundaria.
Nivel de objetivo: General.
Stakeholders e intereses:

Usuario: quiere registrar el recobro de facturas devueltas.

Precondición: El usuario debe haberse identificado en el sistema y deben existir facturas devueltas en la tabla de devoluciones.

Garantías mínimas: El sistema muestra el conjunto de facturas devueltas.

Garantías en caso de éxito: Si el usuario selecciona una factura, el sistema permite su recobro, cambiando el estado de “devuelto” a “recobro”.

Escenario principal de éxito:

1. El usuario selecciona la factura que desea realizar el recobro.
2. El sistema solicita la fecha de recobro.
3. El usuario acepta la solicitud de recobro.

4. El sistema cambia el estado de la factura devuelta a factura “recobrada”.

Extensiones:

3.a El usuario cancela la solicitud.

3.a.1 El sistema no gestiona la solicitud, dejando el estado actual de la factura devuelta.

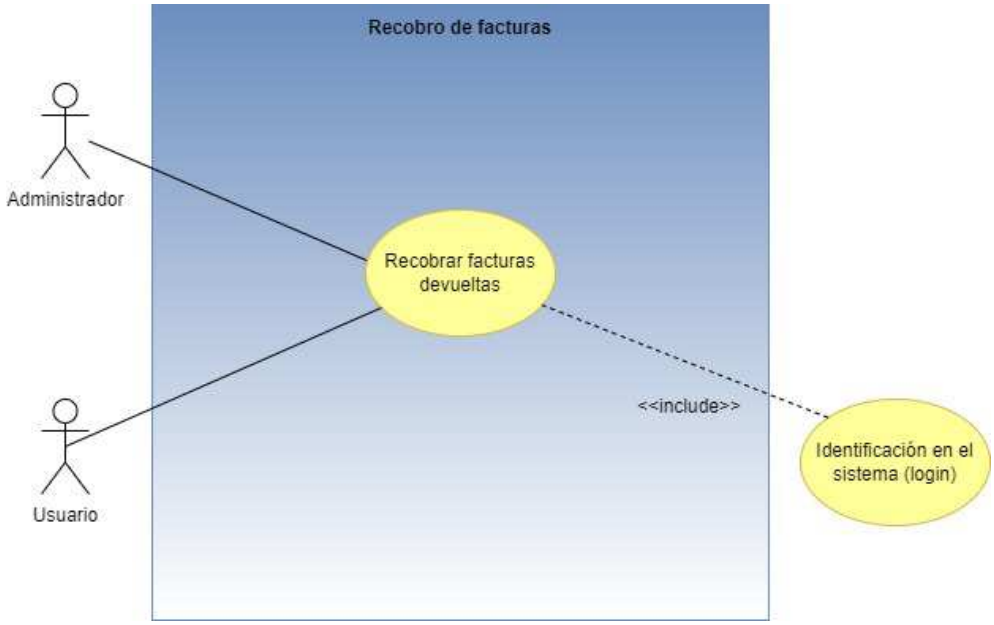


Ilustración 9. Caso de uso *Recobro de facturas.*

CU008: Envío emails de devolución y exportación contable

Actor principal: Usuario final.
Ámbito: Pantalla secundaria.
Nivel de objetivo: General.
Stakeholders e intereses:

Usuario final: quiere enviar emails de devolución a clientes y realizar la exportación contable.

Precondición: El usuario debe haberse identificado en el sistema y deben existir facturas devueltas y pendientes de enviar un email y realizar la exportación contable.

Garantías mínimas: El sistema permite seleccionar las facturas de las que se enviará el email y se realizará la exportación.

Garantías en caso de éxito: El sistema envía un email informando al cliente de la devolución y se realizará la exportación contable con la información necesaria para su registro.

Escenario principal de éxito:

1. El usuario selecciona la/s factura/s de las que desea enviar un email informando la devolución y registrarla contablemente.
2. El sistema solicita información necesaria para su registro.
3. El sistema comprueba que no se haya enviado ningún email y tampoco esté registrada la devolución de la factura seleccionada.
4. El usuario confirma la solicitud.

Extensiones:

3.a El sistema informa de que ya existen los registros de envío de email y enlace contable. El sistema no realiza el envío del email y anula la solicitud.

4.a El usuario cancela la solicitud.

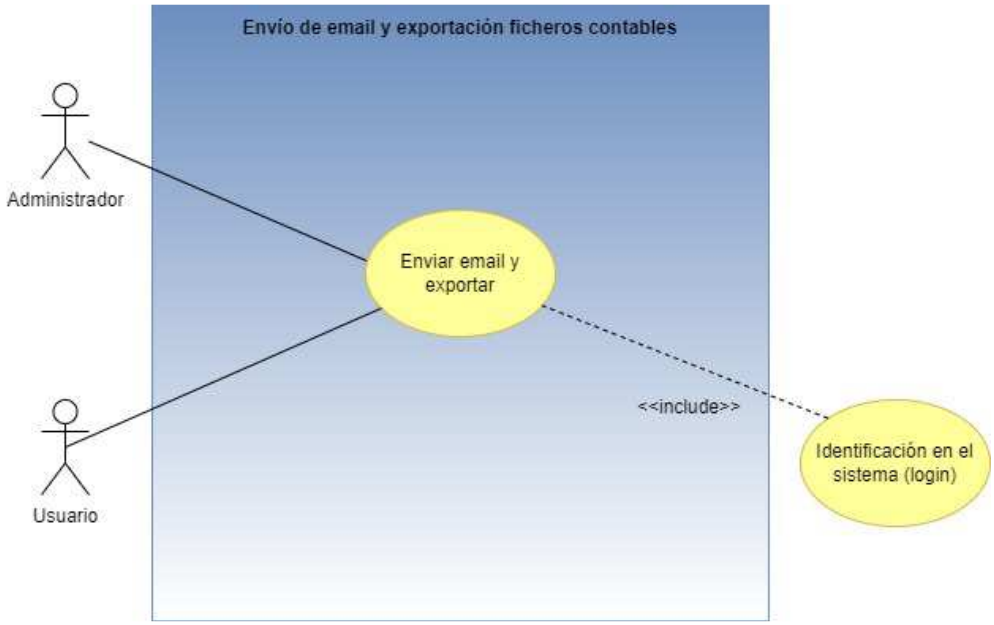


Ilustración 10. Caso de uso *Envío de email y exportación ficheros contables.*

3. Diseño y arquitectura

3.1 Análisis y diseño del sistema

Se modelan todas las especificaciones obtenidas en el análisis de requisitos. El objetivo de esta fase de diseño es la de modelar desde un punto de vista lógico, todo el sistema en vía de desarrollo. Por tanto, se trata de definir qué ha de hacer el sistema, cómo y los datos que necesita para lograrlo.

Teniendo en cuenta que el desarrollo se realizará bajo el paradigma de la programación orientada a objetos (POO), es necesario utilizar un modelo

conceptual de datos. De forma independiente a la tecnología elegida, en el desarrollo se ha tenido en cuenta la información necesaria y se ha expresado por medio de un modelo relacional de datos (entidad-relación) donde se muestra tanto las entidades más relevantes como sus distintas relaciones y propiedades.

3.2 Modelo conceptual de datos

3.2.1 Diseño relacional de datos (Tablas y datos)

Para continuar con el diseño de clases, necesitamos en primer lugar conocer los datos que necesitaremos almacenar para realizar las diferentes funciones del sistema.

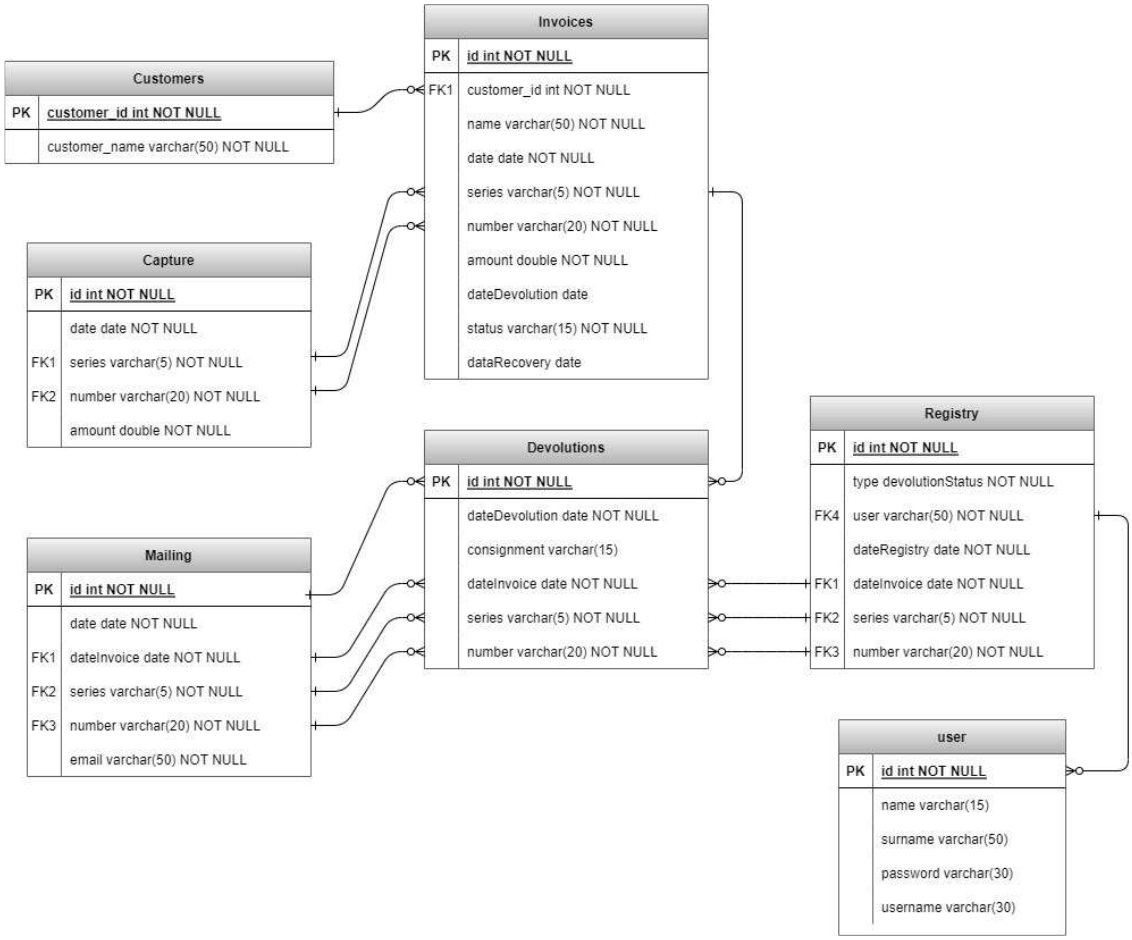


Ilustración 11. Diseño relacional de datos.

3.2.2 Modelo de dominio / Diagrama de clases / Punto de vista de la información

El punto de vista de la información describe con detalle la información que va a tratar el sistema, incluyendo las estructuras de datos que maneja, sus posibles valores y las relaciones entre ellos.

Esta forma de representar la información nos ofrece una visión global y nos permite garantizar la coherencia de los datos del sistema que manejan las demás vistas. En esta fase, el lenguaje de información específica tanto la información como sus restricciones, y su estado y estructura viene restringido por esquemas estáticos, dinámicos e invariantes.

En concreto, los esquemas invariantes especifican los predicados que se han de cumplir en cualquiera de los estados por los que atraviese el sistema. Estos incluyen los tipos de objetos que encapsulan la información manejada por el sistema, sus posibles estados, las relaciones existentes entre ellos y las posibles restricciones entre los tipos y relaciones.

Este esquema invariante desde el punto de vista de la información, lo expresaremos mediante un esquema UML.

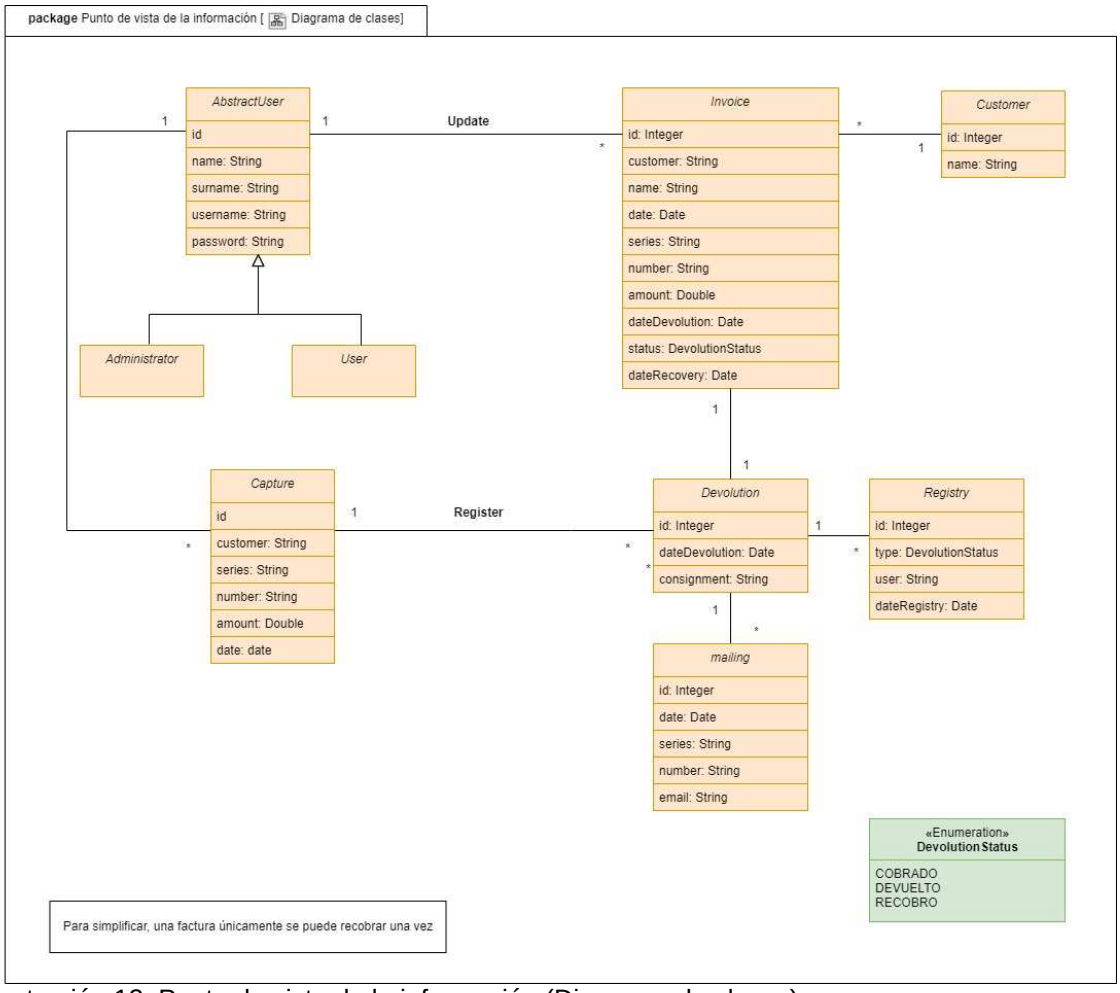


Ilustración 12. Punto de vista de la información (Diagrama de clases).

3.3 Arquitectura

La arquitectura elegida y adecuada para el proyecto en cuestión es una arquitectura de flujo de datos. Puesto que el uso de datos en diferentes unidades es la base de este proyecto, la arquitectura acorde para el proyecto será del tipo Cliente-servidor basada en capas. A la hora de su instalación, se decidirá donde

residirá la capa de datos, preferiblemente en un servidor, aunque también se ha decidido que pueda funcionar únicamente a nivel local.

La aplicación estaría distribuida mediante 3 capas[14]:

- Capa de la aplicación o presentación.
- Capa lógica o de negocio.
- Capa lógica de datos.

Las capas proporcionan diferentes servicios a los elementos de la capa inmediatamente anterior, sirviéndose de los servicios que le brindan los elementos de la capa posterior. Por otro lado, la principal ventaja de una arquitectura cliente-servidor es que **se permite de manera natural la distribución de sus componentes**. Además de la ventaja añadida de la posibilidad de **ampliar funcionalidades de forma sencilla** debido a la fácil **escalabilidad**[15] de este tipo de arquitectura. Por tanto, teniendo en cuenta la combinación de arquitecturas, se estaría utilizando una arquitectura heterogénea de sistemas cliente-servidor organizados en capas.

A pesar de las desventajas que pudiera tener una aplicación local, las ventajas en este caso las superan. En concreto, se busca en la aplicación las siguientes:

- **Tanto el programa como los datos se encuentren ubicados en el mismo ordenador.** Más que por una ventaja, permite un funcionamiento más fluido y rápido.

- **Rapidez en la ejecución de las tareas.** Puesto que no depende de servidores externos, la ejecución de las tareas es más rápida.

- **Interfaz más eficiente.** Tanto por el funcionamiento de aplicaciones externas, como Tesseract, así como la ejecución de la propia interfaz, al tratarse de forma local será más eficiente que si fuera remota.

3.3.1 Arquitectura lógica y arquitectura física

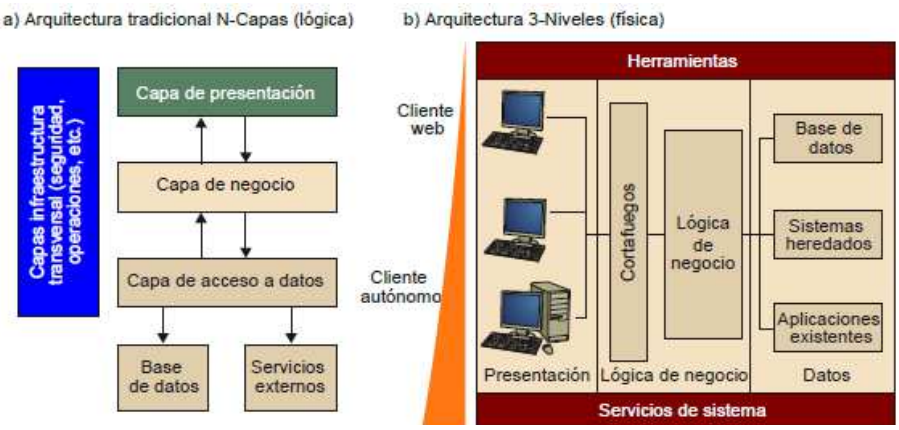


Ilustración 13. Ejemplos arquitectura lógica y arquitectura física.
(Fuente: Josep Maria Camps Riba – Java EE. Figura 3, pág. 19)

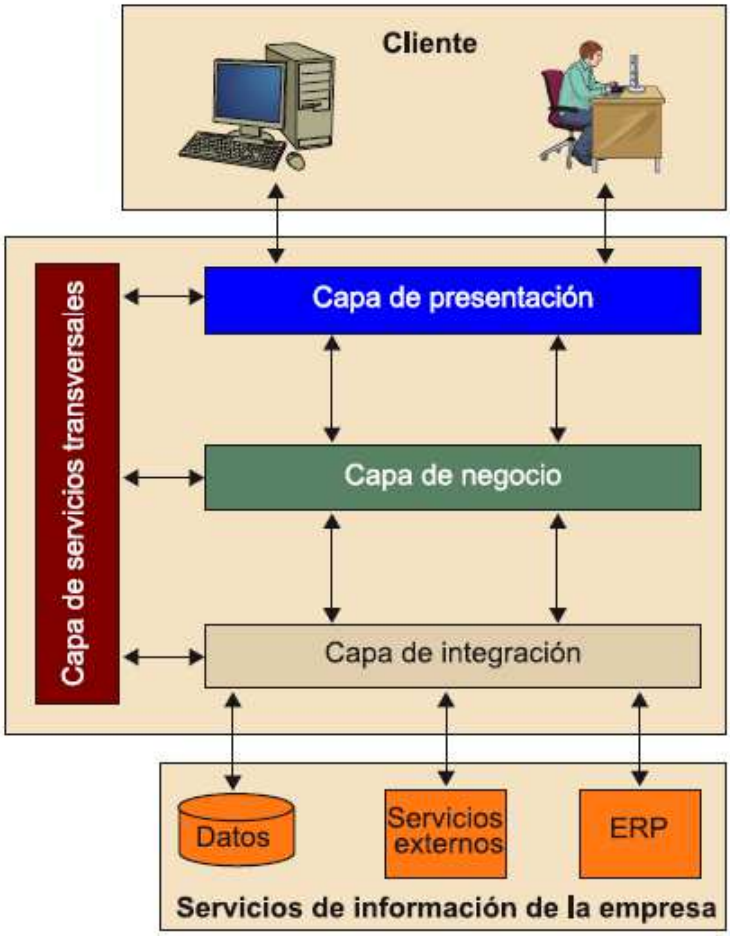


Ilustración 14. Ejemplo arquitectura lógica 4 capas.
(Fuente: Josep Maria Camps Riba – Java EE. Figura 4, pág. 20)

En la imagen se observan 4 niveles o capas lógicas, donde además de la capa de presentación, negocio y de integración, se añaden una capa de servicios transversales (siendo estos la vía para permitir un acceso a unos servicios comunes que pudieran necesitar todos los componentes de las diferentes capas).

Para la aplicación diseñada será una aplicación multicapa, siendo el número elegido la de 3 capas (presentación, negocio e integración). De esta manera, al lograr una correcta distribución alcanzada por las capas incrementa la mantenibilidad, la robustez, la extensibilidad, la escalabilidad y la seguridad de la aplicación.

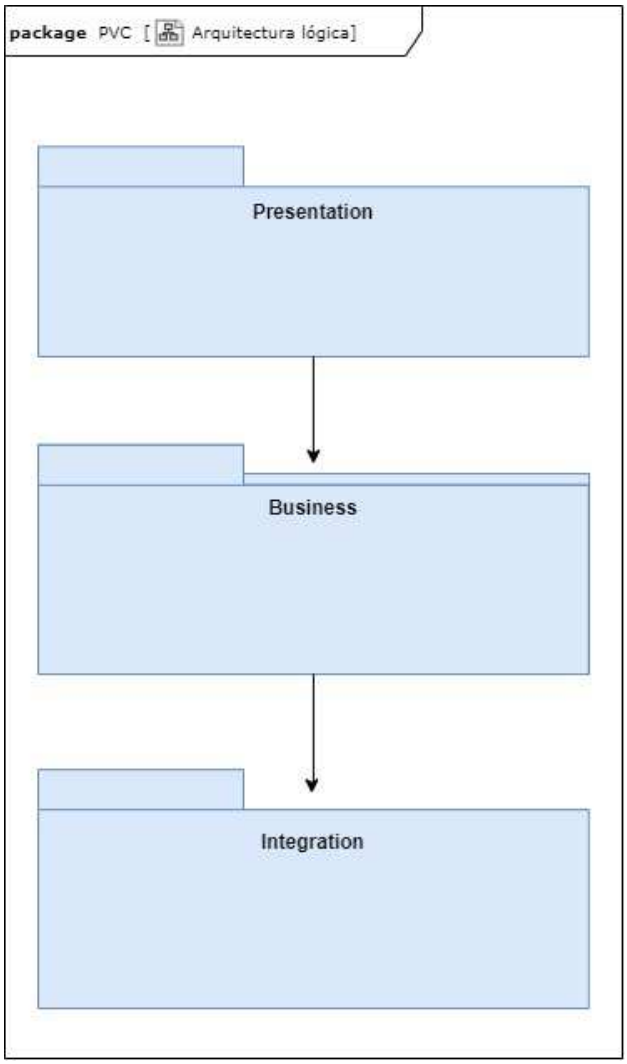


Ilustración 15. Punto de vista de la computación (Arquitectura lógica).

Para la arquitectura física se pudiera utilizar una arquitectura de dos niveles (cliente-servidor) como de 3 niveles como las que se muestran en las dos imágenes a continuación.

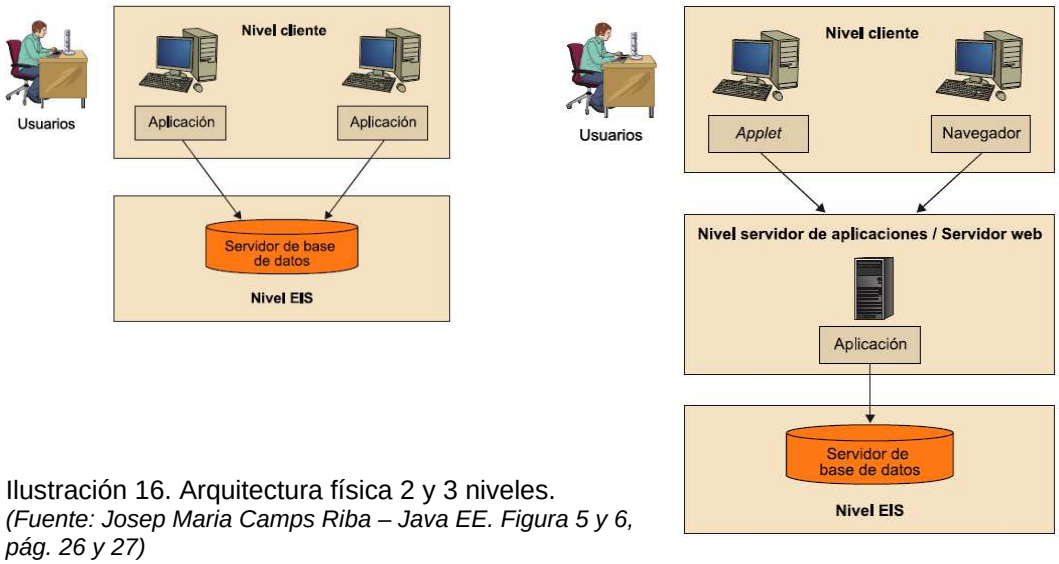


Ilustración 16. Arquitectura física 2 y 3 niveles.
(Fuente: Josep Maria Camps Riba – Java EE. Figura 5 y 6, pág. 26 y 27)

Para la aplicación, la decisión más idónea y posible solución para la arquitectura física sería la de dos niveles, pudiendo ser como sigue:

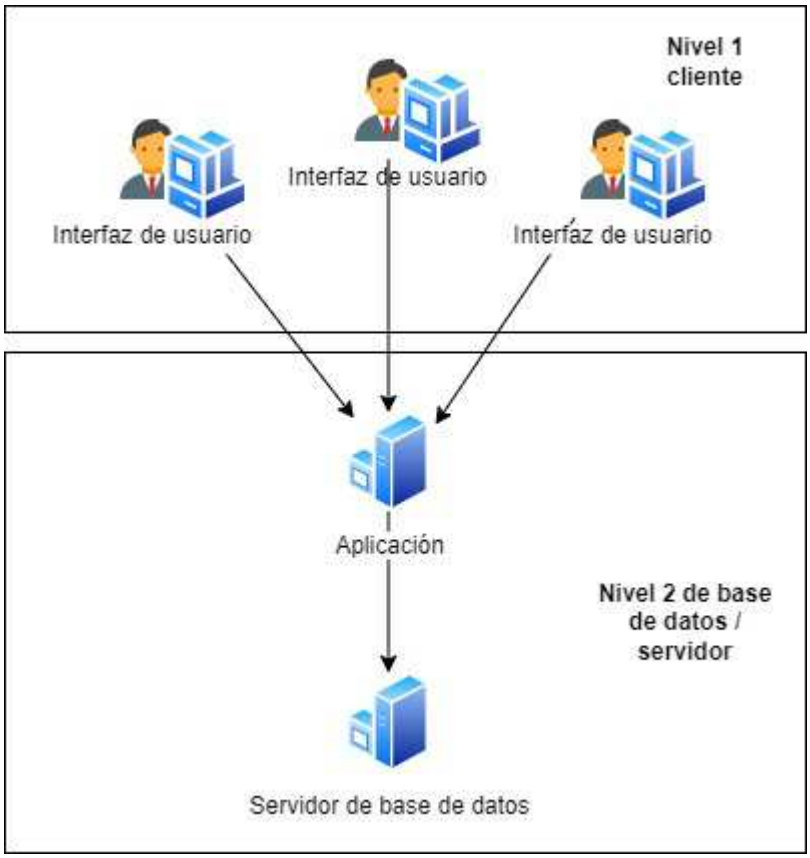


Ilustración 17. Arquitectura 2 niveles.

3.3.2 Diagrama de despliegue

El diagrama de despliegue nos permite modelizar la distribución física, en tiempo de ejecución, de los diferentes artefactos de software. Gracias a este diagrama se puede mostrar el tipo de conexión o comunicación entre los diferentes nodos/dispositivos. De esta forma, se puede observar cómo el sistema se desplegará físicamente en el hardware.

El sistema se ha diseñado para funcionar a nivel local, pudiendo estar instalado localmente, tanto para la interfaz de usuario como la base de datos, como también para una intranet, en el que la base de datos se encuentra ubicada en el servidor, tal y como se muestra en la siguiente imagen.

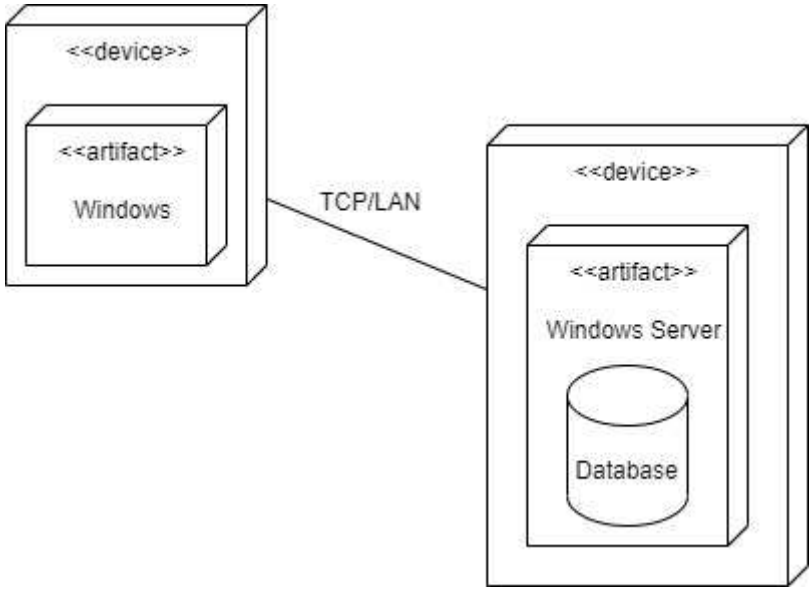


Ilustración 18. Diagrama de despliegue.

Su funcionamiento está basado para funcionar en Windows para el lado del cliente y en Windows/Windows Server para el lado del servidor. Las comunicaciones se realizan vía TCP/LAN por tratarse de una Red de Área Local (LAN).

3.4 Diseño Arquitectura lógica. Punto de vista de la computación

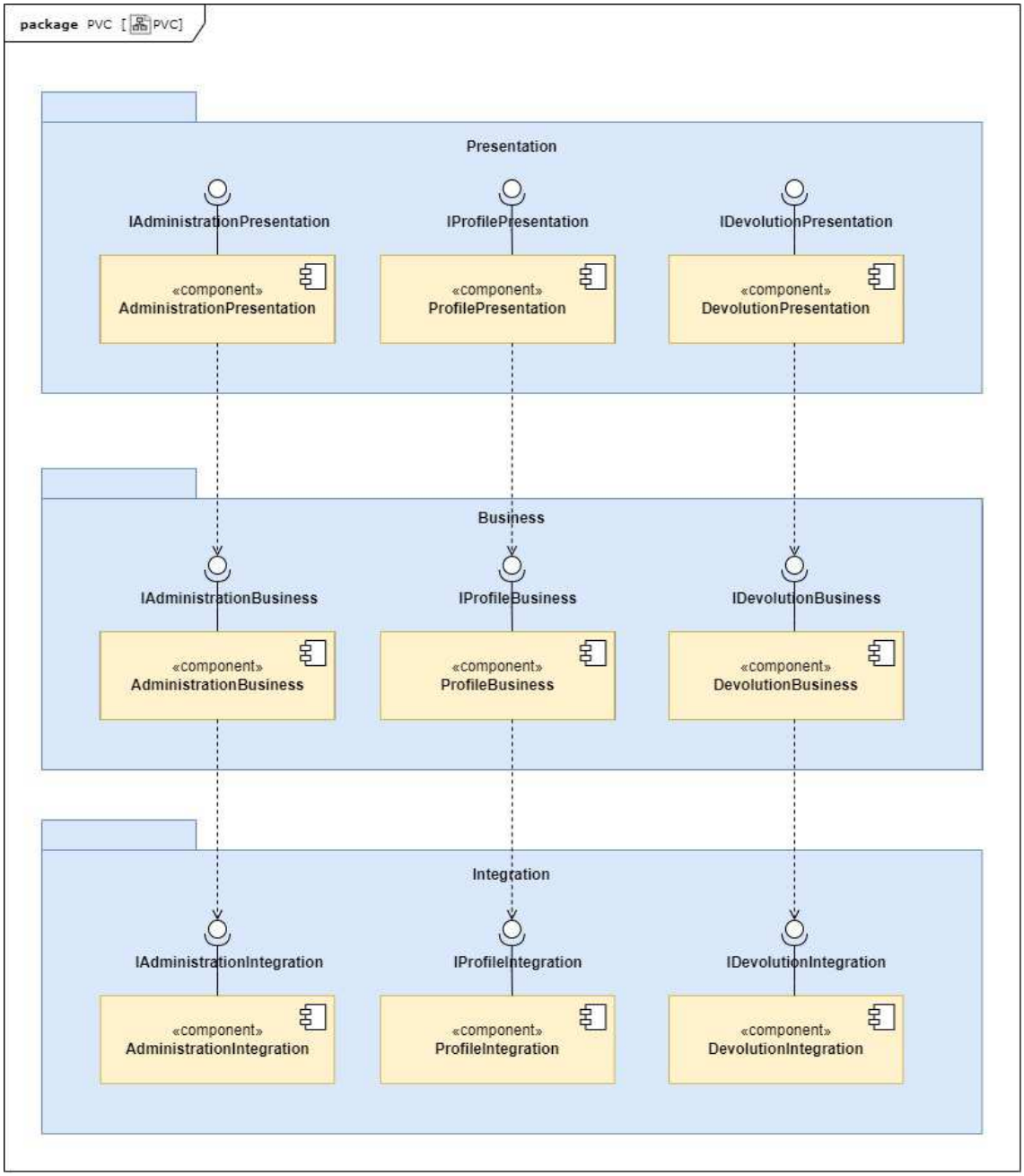


Ilustración 19. Punto de vista de la computación.

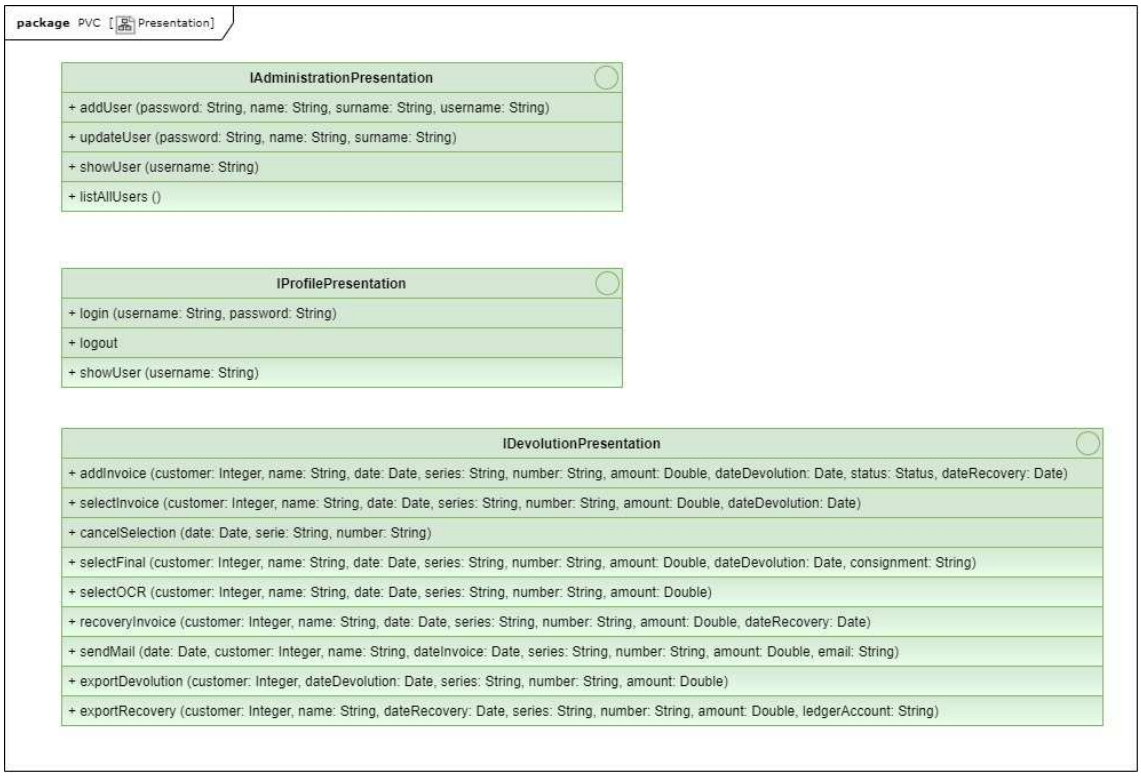


Ilustración 20. Punto de vista de la computación (Presentación)

Ilustración 21. Punto de vista de la computación (Business)

Ilustración 22. Punto de vista de la computación (Integration).

En cuanto al diseño de las capas y de acuerdo con sus características, se ha decidido aplicar el patrón MVC[17] (modelo, vista y controlador) separando de esta forma los diferentes componentes en objetos separados y por tanto, que cada componente no se pueda combinar dentro de una misma clase.

El patrón MVC tradicionalmente se utilizaba para interfaces gráficas de usuario de escritorio, por lo que al tratarse nuestra aplicación de una aplicación de escritorio, es idónea. Cabe decir que en la actualidad, el patrón MVC se ha hecho muy popular para el diseño de aplicaciones Web y el resto de aplicaciones.

Es en la capa de presentación, donde se mostrarán las vistas, la capa que necesita de un mayor refinamiento. El siguiente apartado muestra el refinamiento de cada una de las capas propuestas donde se especifica mayor detalle.

3.5 Refinamiento de las capas

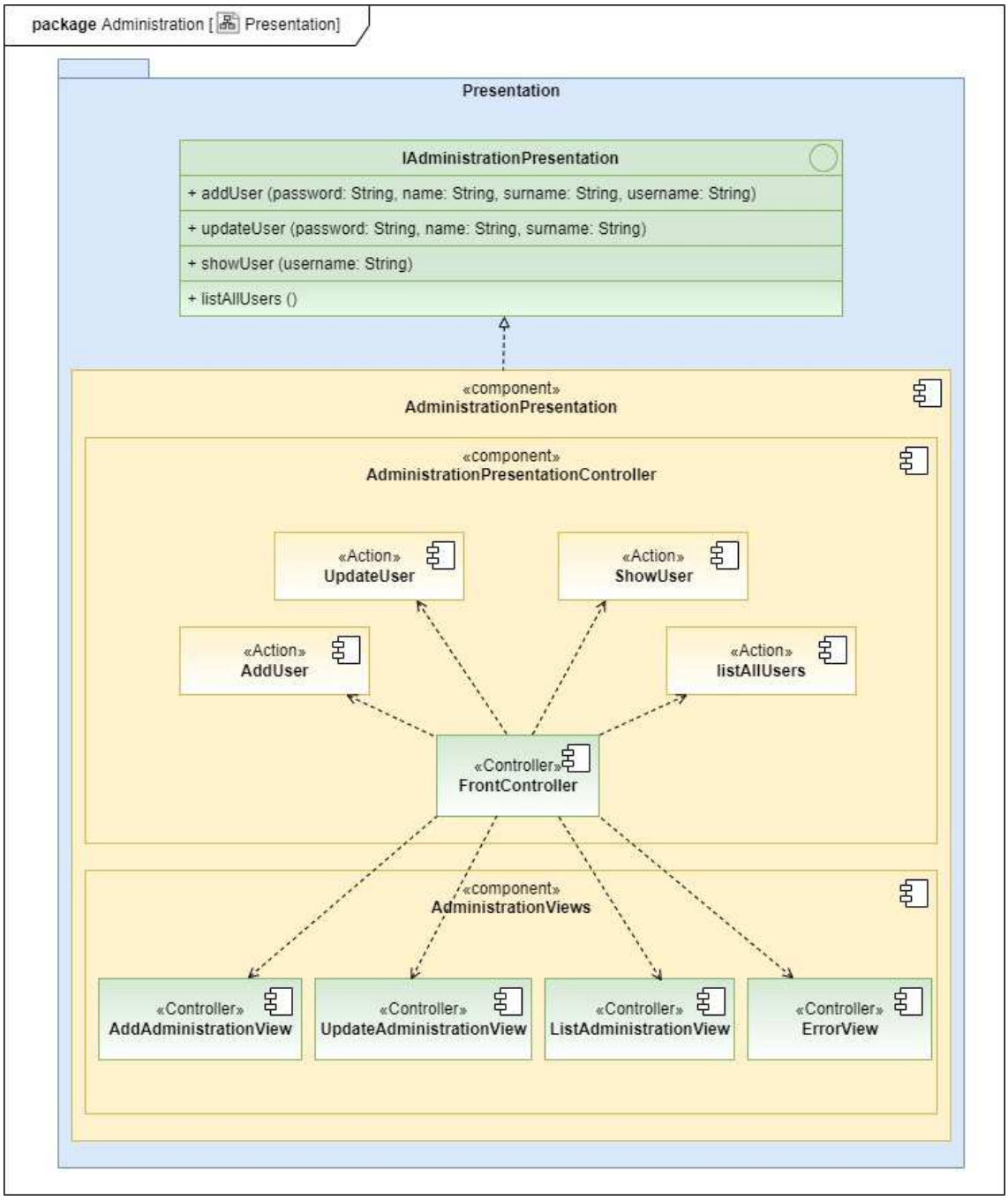


Ilustración 23. PVC módulo Administration (Presentation).

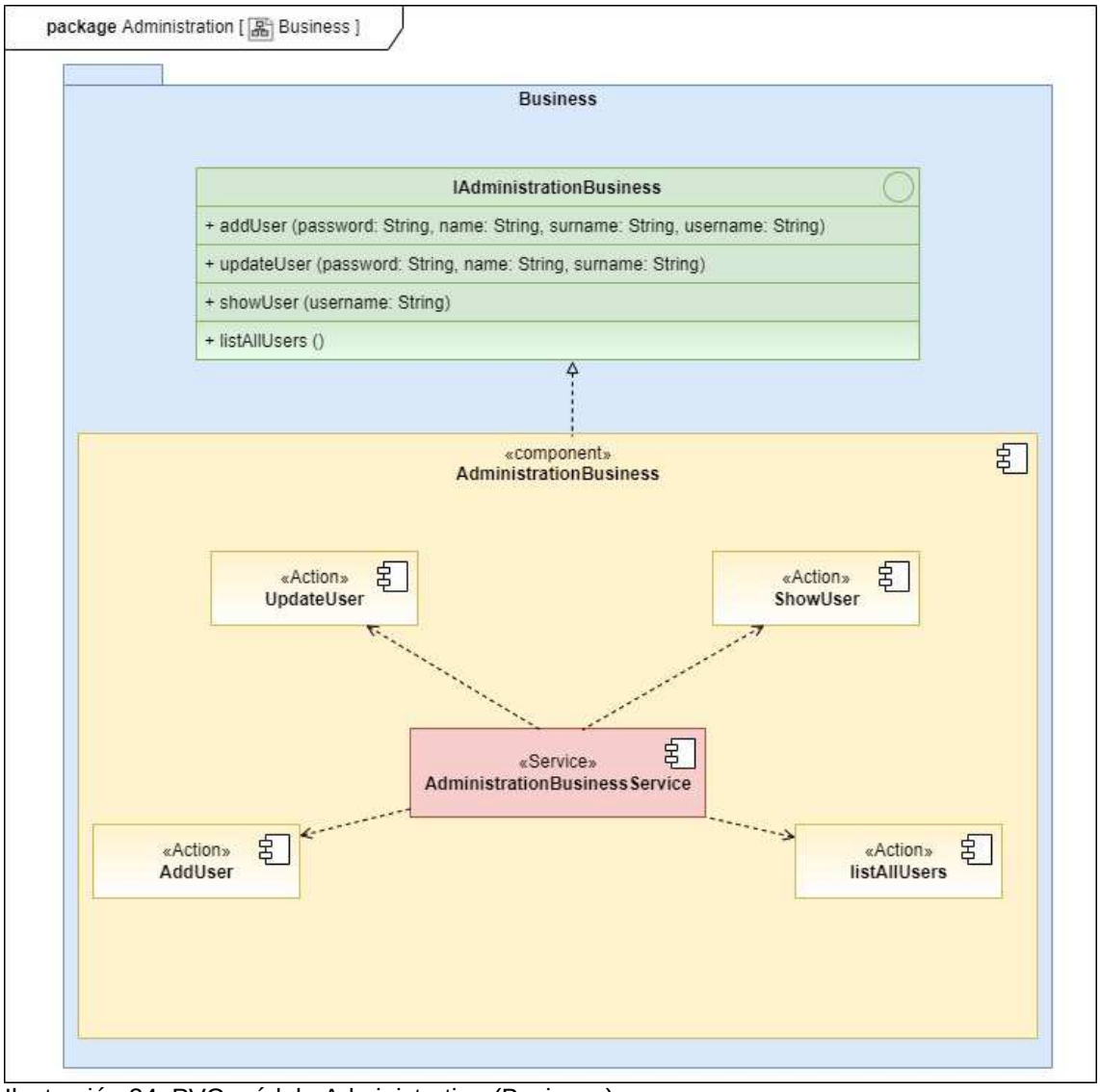


Ilustración 24. PVC módulo Administration (Business).

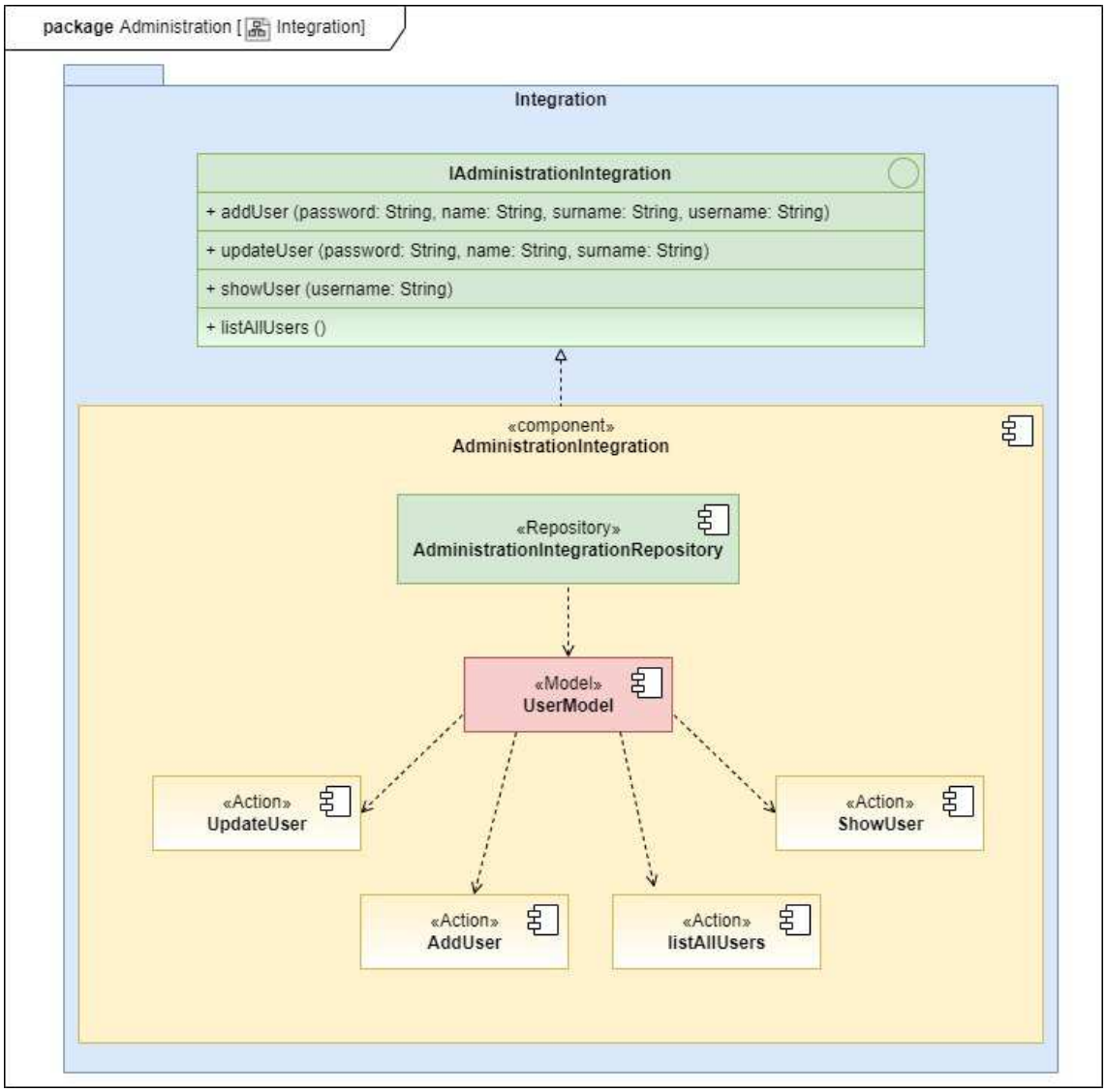


Ilustración 25. PVC módulo Administration (Integration).

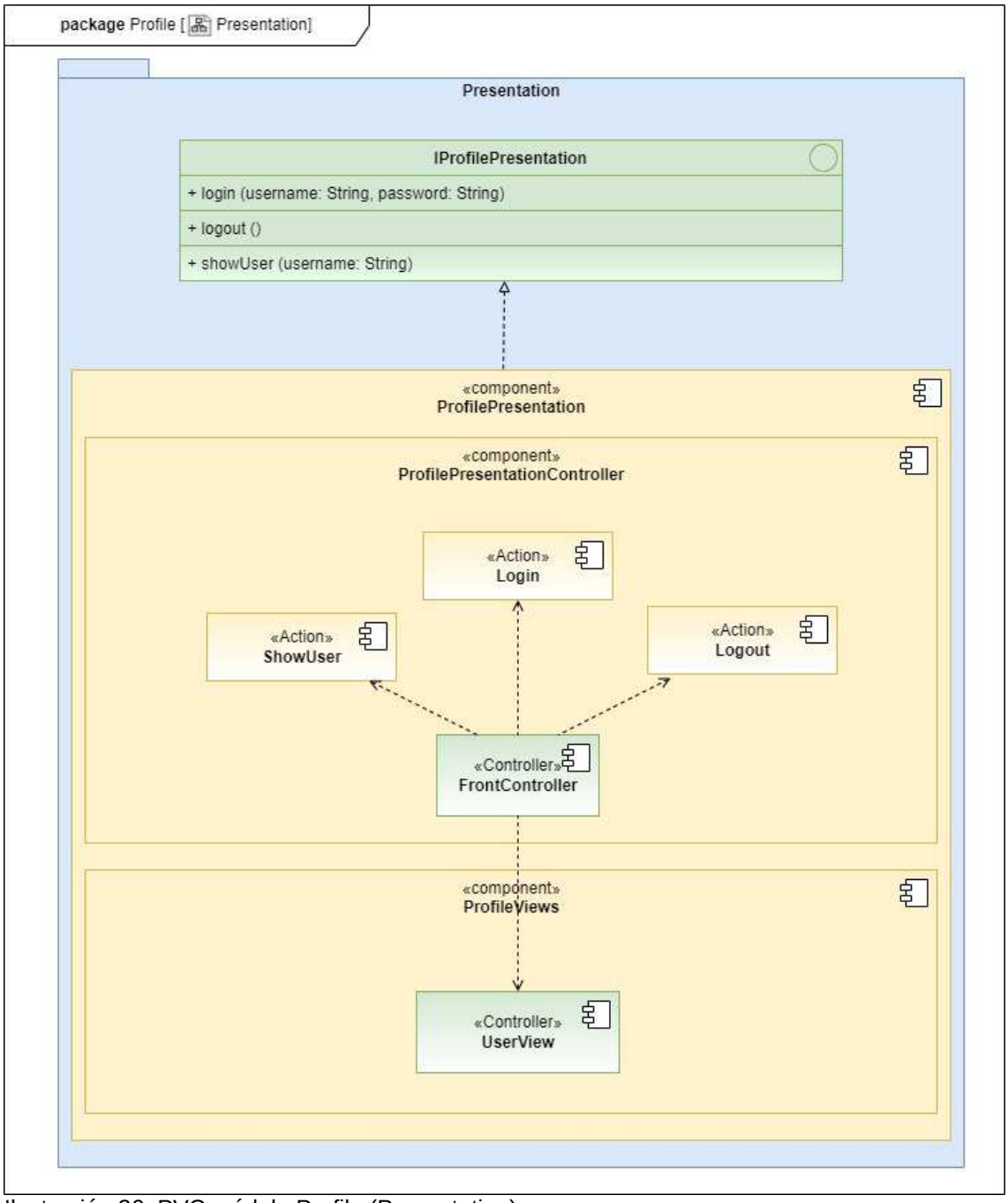


Ilustración 26. PVC módulo Profile (Presentation).

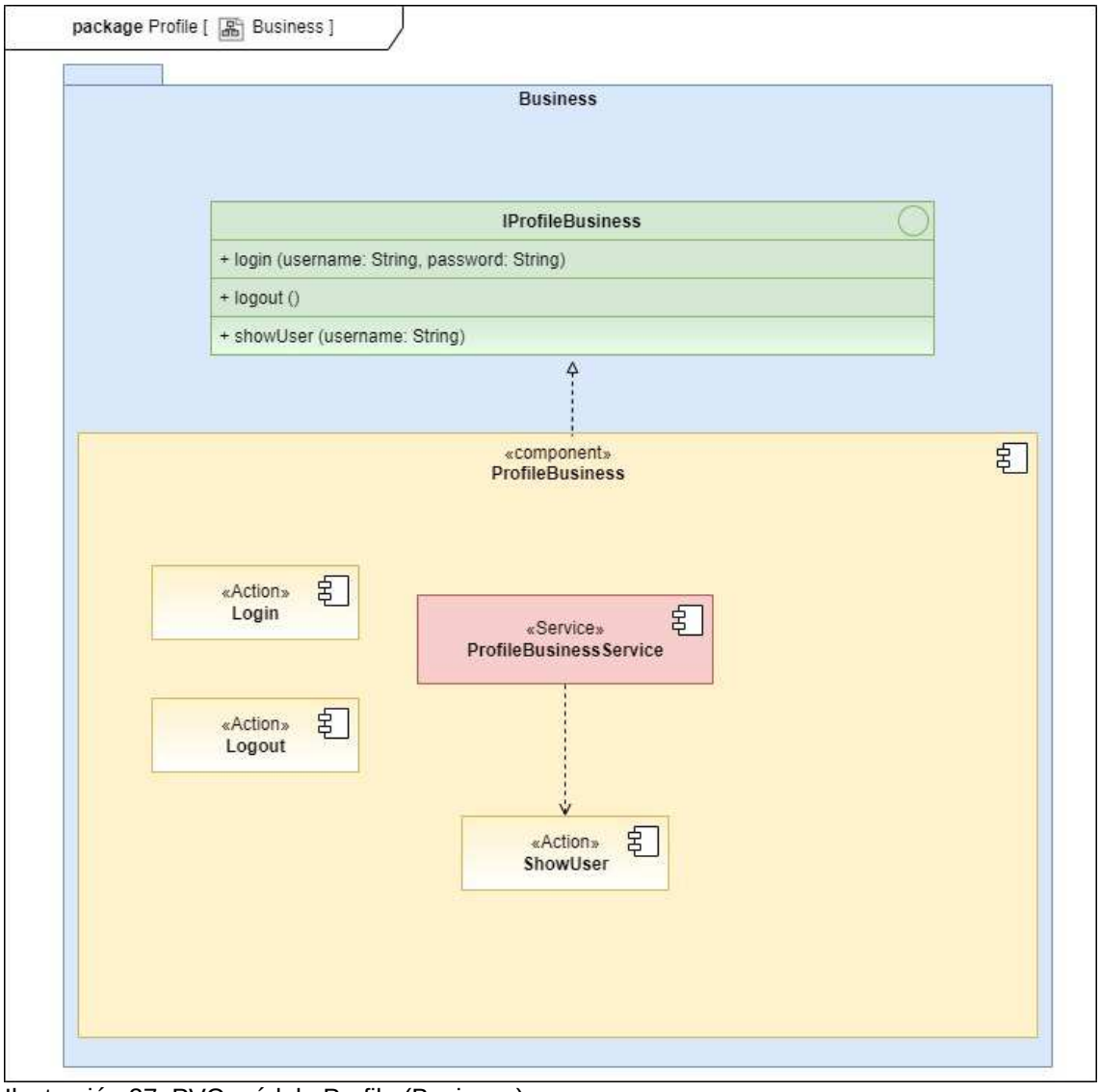


Ilustración 27. PVC módulo Profile (Business).

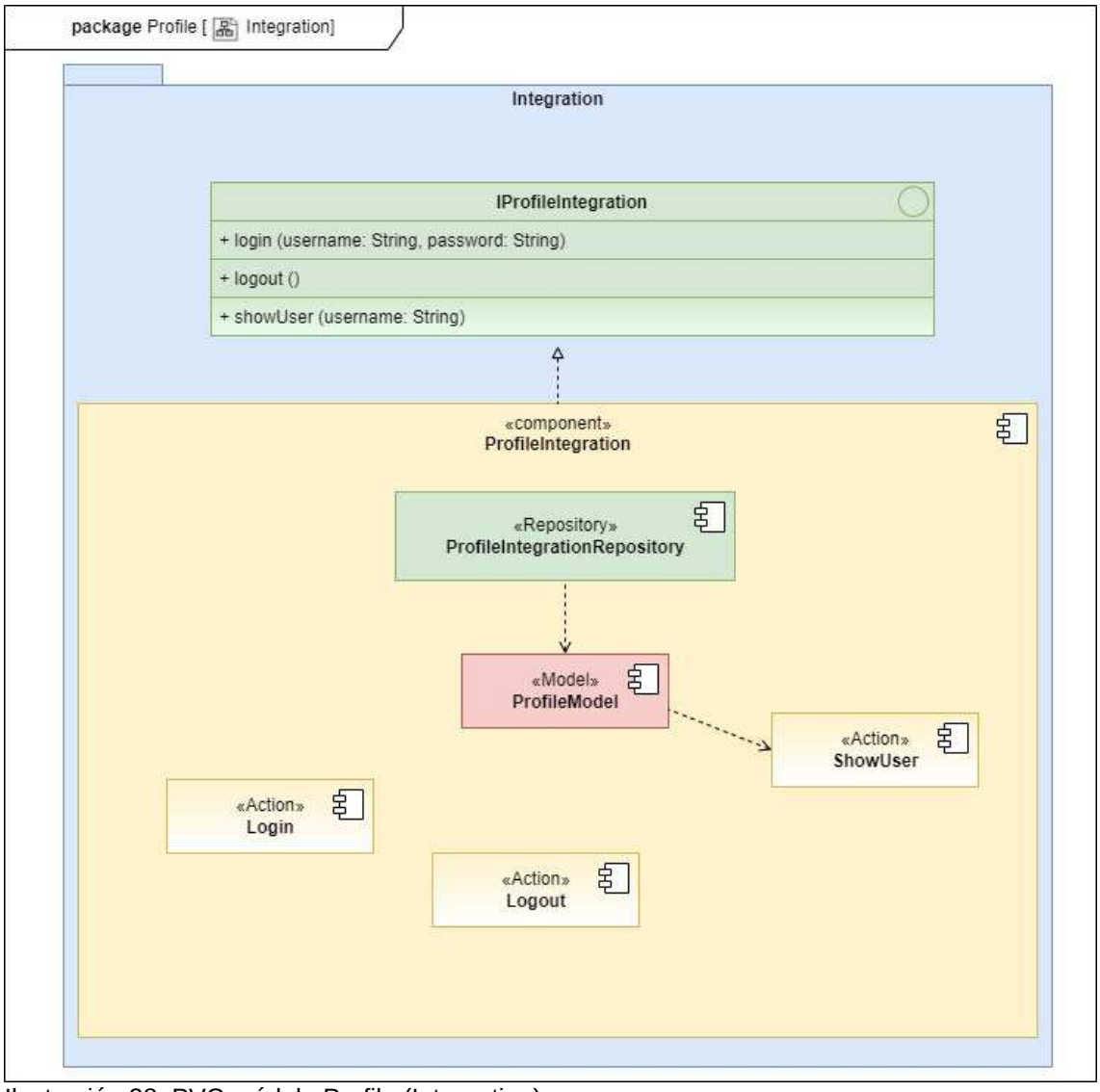


Ilustración 28. PVC módulo Profile (Integration).

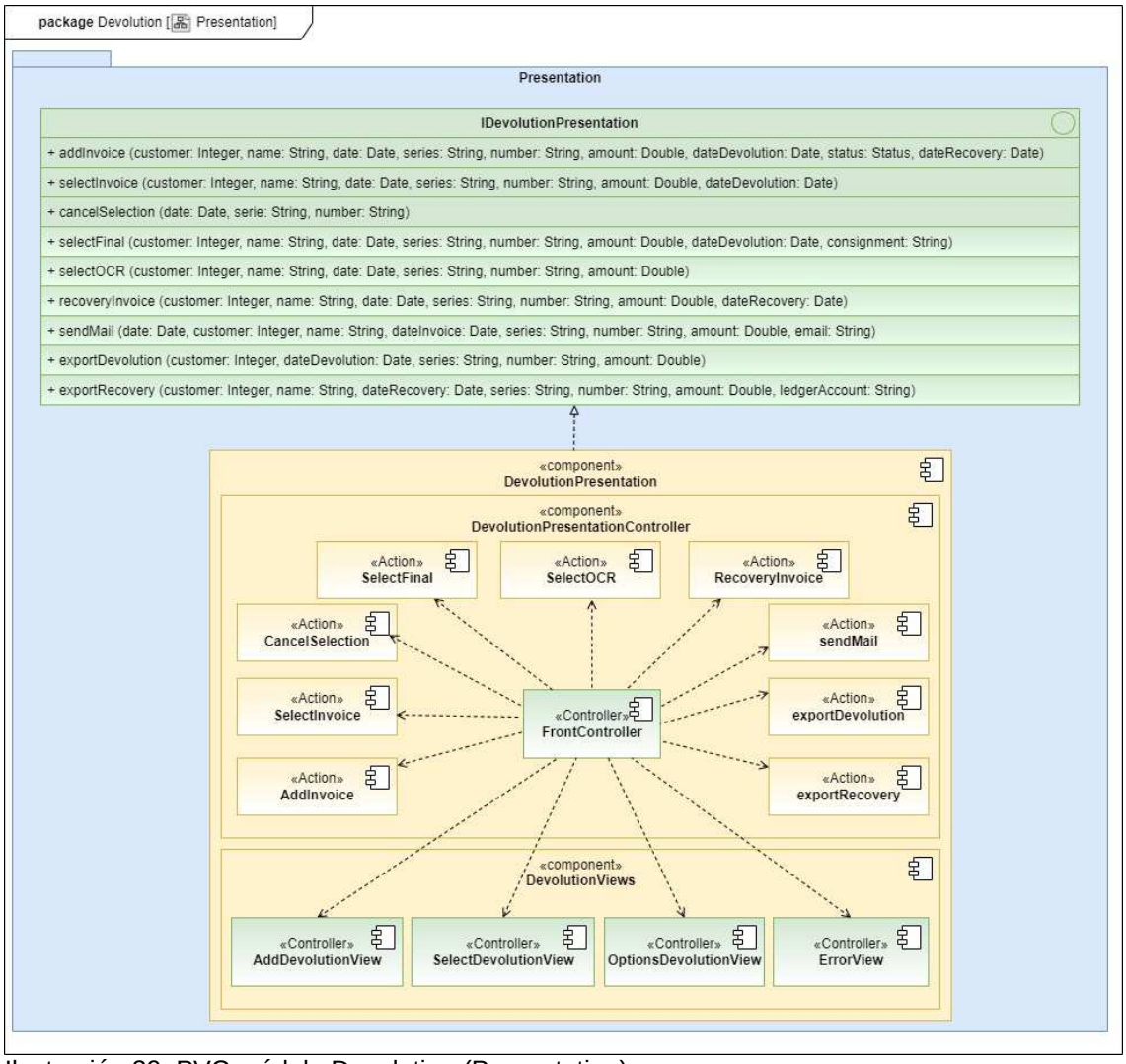


Ilustración 29. PVC módulo Devolution (Presentation).

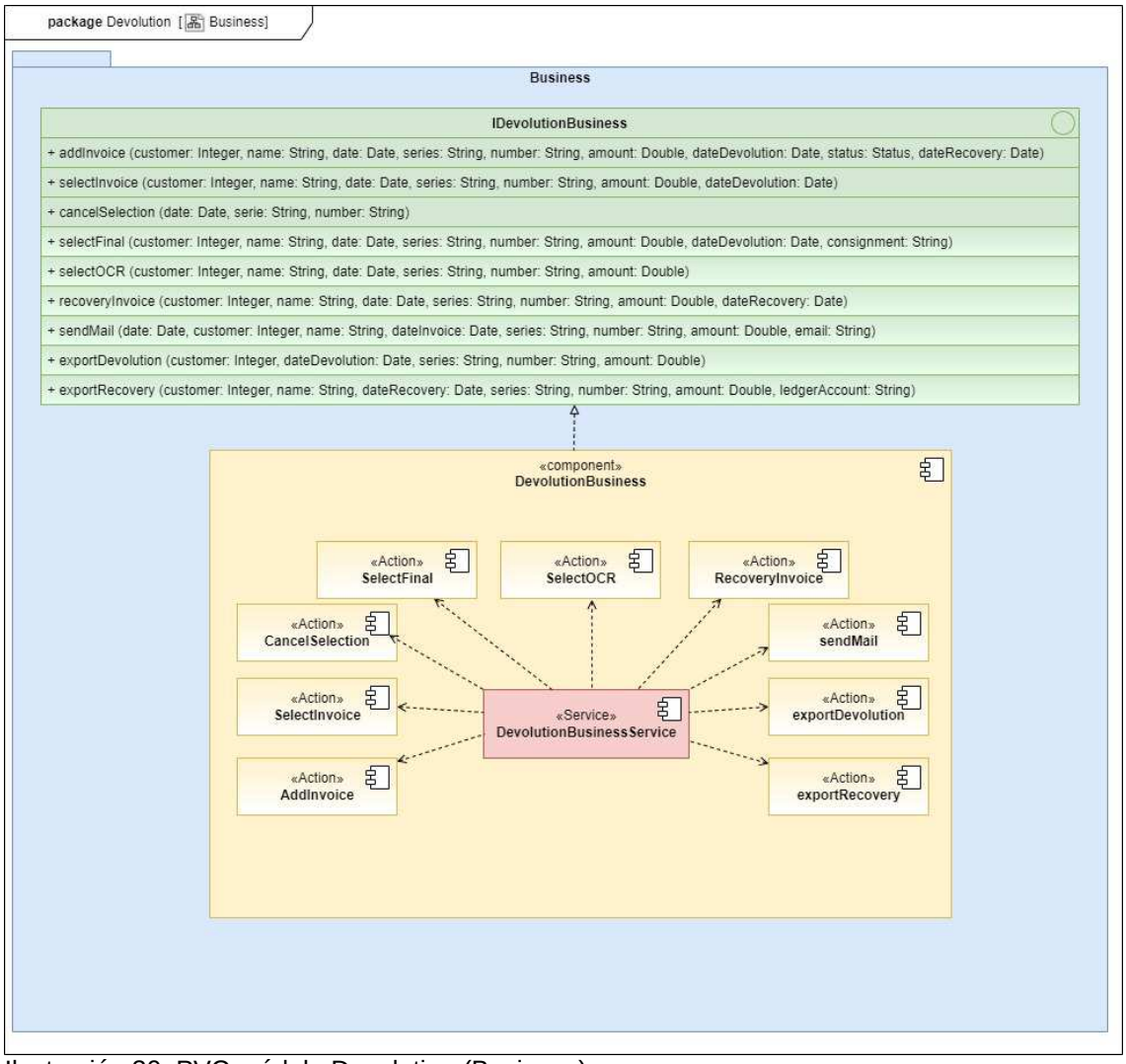


Ilustración 30. PVC módulo Devolution (Business).

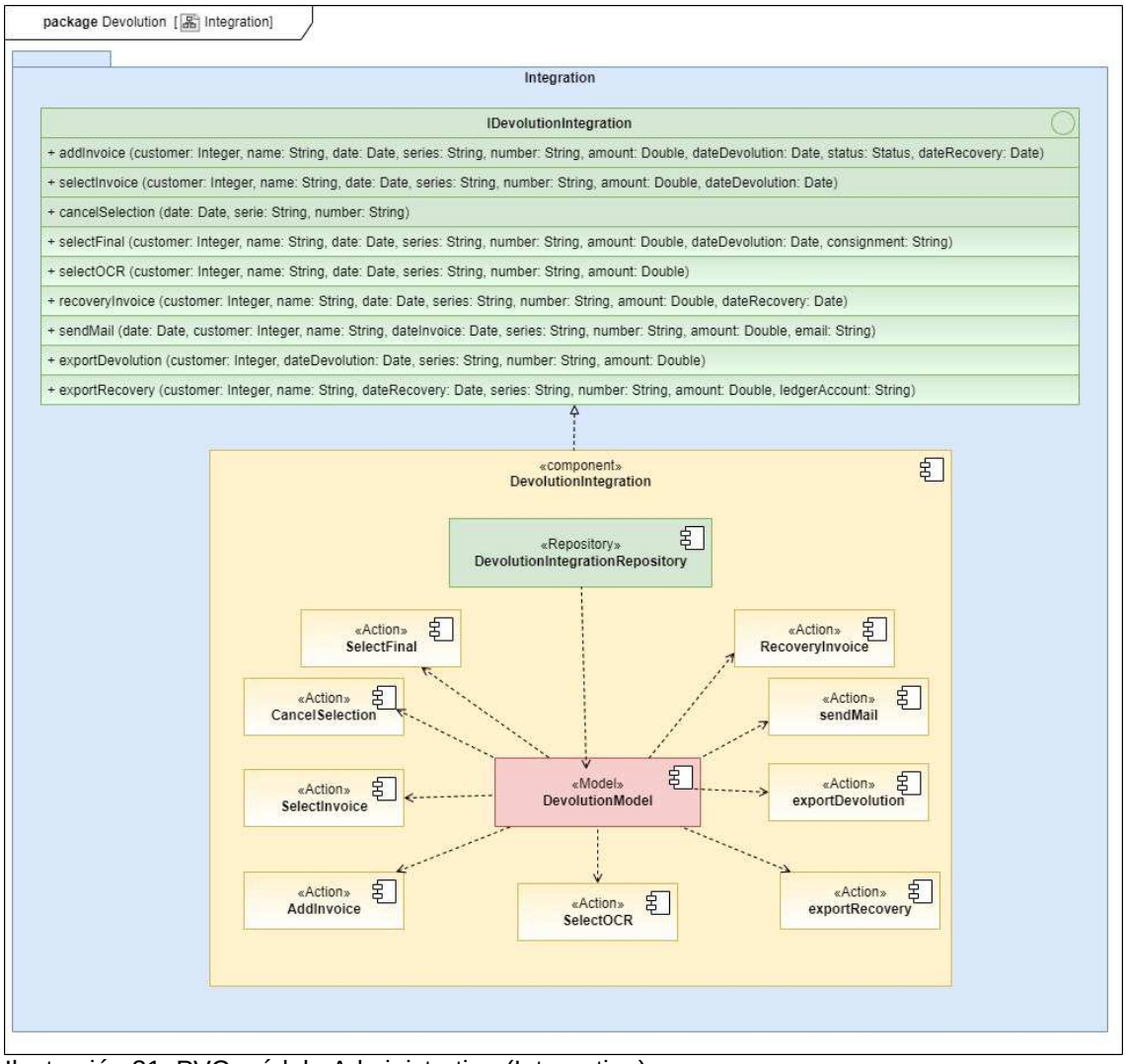


Ilustración 31. PVC módulo Administration (Integration).

En el diseño se ha procedido a la creación de módulos, además de componentes, servicios y modelos. A continuación, se explica la función de cada uno de ellos:

- Módulo: Los módulos permiten la modularización de la aplicación, creando partes bien diferenciadas de forma que queden bien delimitadas ciertas funciones frente a otras.
- Componente: Los componentes son las diferentes partes de la aplicación y que permiten realizar las diferentes acciones para un funcionamiento fluido del usuario.
- Servicios: A partir de los servicios se busca la conexión entre los componentes y los datos.
- Modelo de datos: Se tratarían de los ficheros con los atributos de los datos con los que se trabajará.

En el caso de mi proyecto, los módulos se distribuyen en 3: Administration, Profile, Devolution.

3.6 Capa de integración

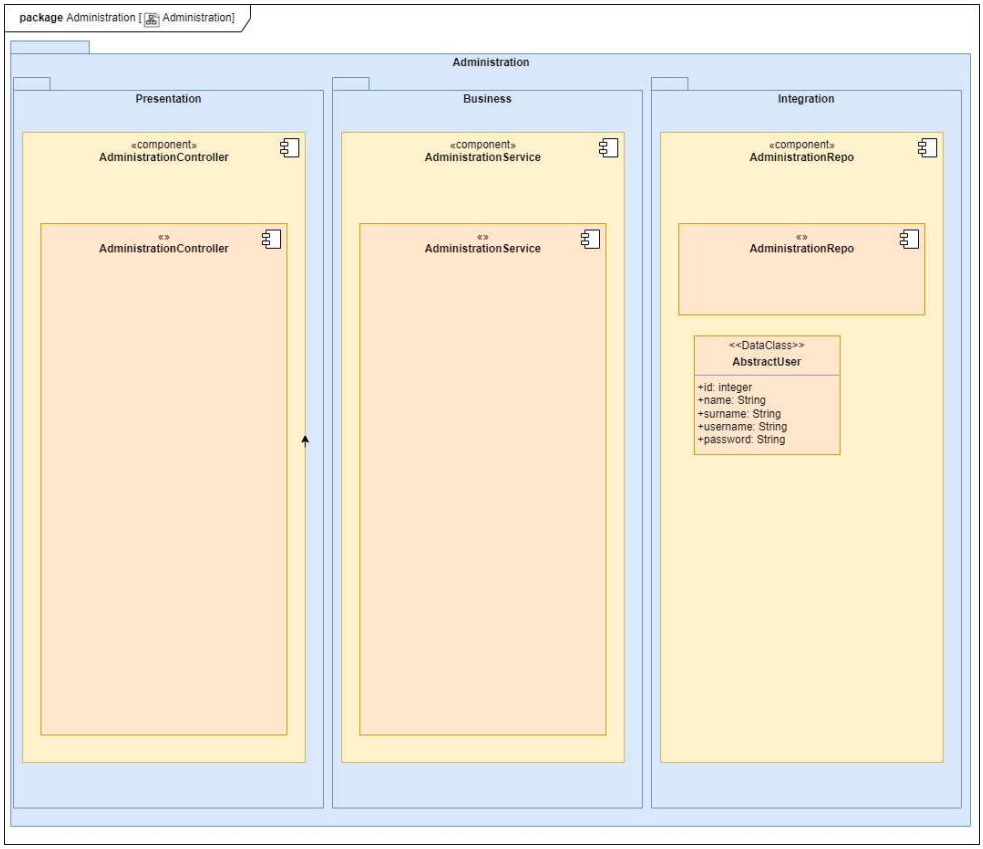


Ilustración 32. Capa de integración módulo Administration.

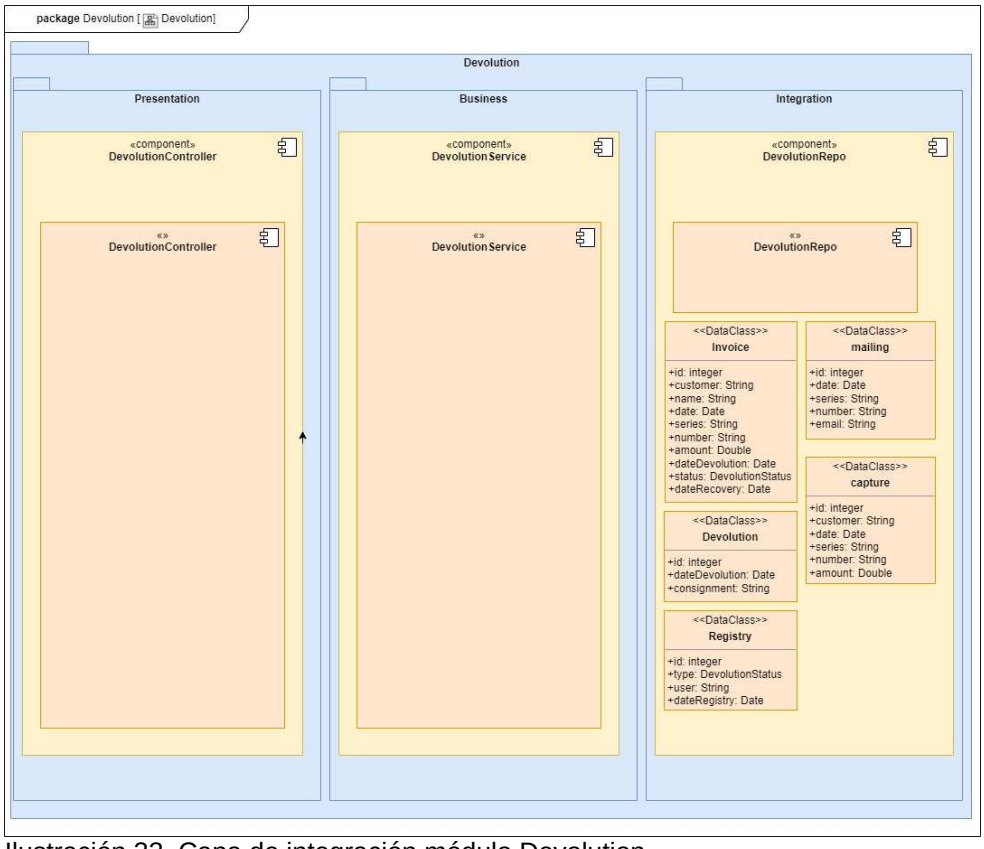


Ilustración 33. Capa de integración módulo Devolution.

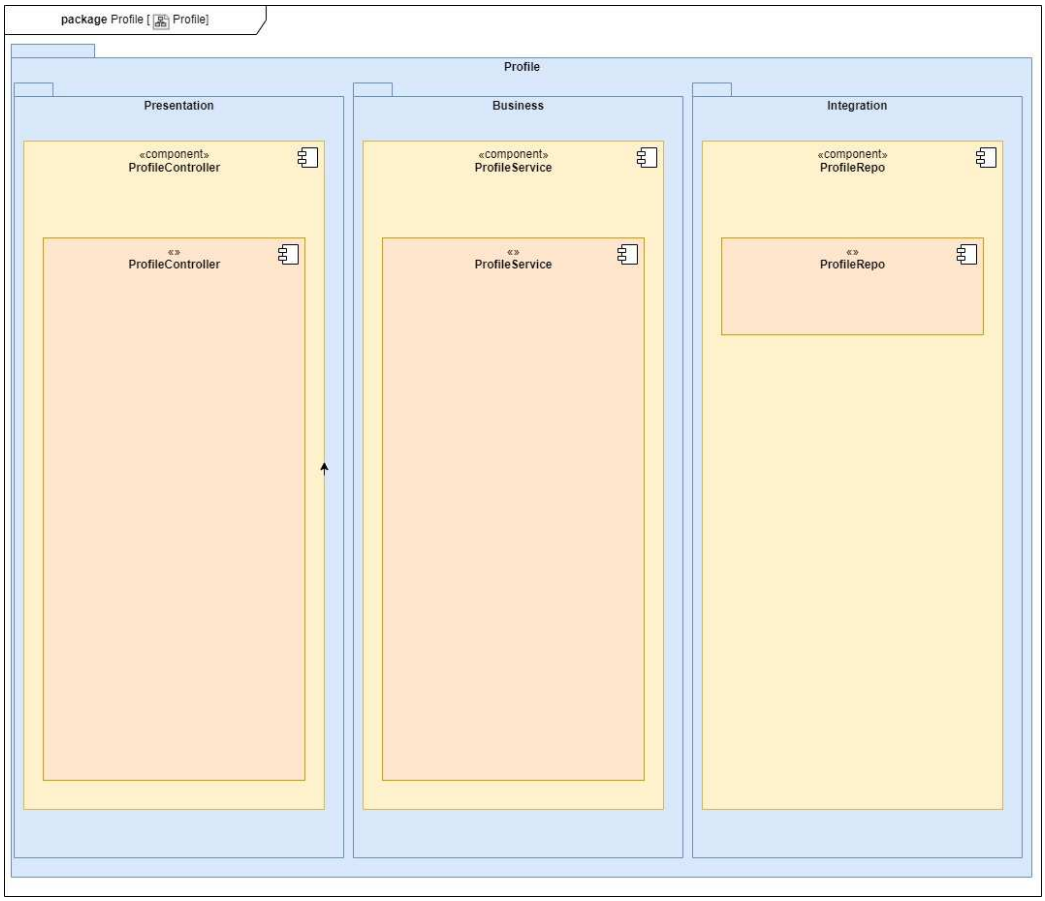


Ilustración 34. Capa de integración módulo Profile.

Inicialmente, la tecnología para la aplicación podría ser JAVA EE[18] como cualquiera similar, pero se ha elegido .NET[19]. La aplicación se deberá implementar bajo el paradigma de la programación orientada a objetos (POO). La principal idea es que la aplicación pueda hacer uso de los servicios, contenedores y los componentes, definiéndolos como sigue:

- Servicios: Se tratan de servicios que permiten que el desarrollador se concentre en la lógica de negocios, mientras que estos proporcionan funcionalidades sin la necesidad de tener que desarrollarlos.
- Contenedores: Son entornos en tiempos de ejecución que cargan los diferentes servicios.
- Componentes: Son objetos que contienen la lógica de negocio de la aplicación y usan los servicios proporcionados por el contenedor.

Para el desarrollo de aplicaciones Windows utilizaríamos .NET Framework 4.0 y mediante **Windows Forms**. Los *formularios* (Forms) funcionarán como superficies visuales donde se mostrará información al usuario. Por otro lado, tanto para mostrar como aceptar la entrada de datos se utilizarán *controles.* Y a partir de las acciones del usuario se generan eventos que son los que activan las diferentes acciones y funcionalidades. Aunque también podría usarse WPF (Windows Presentation Foundation) que también permite la creación de aplicaciones cliente de escritorio.

3.7 Diagrama de secuencia críticos

Un diagrama de secuencias perteneciendo al tipo de diagrama de interacción nos permite expresar y describir cómo un grupo de objetos funcionan en conjunto y en qué orden. De esta manera, nos permite tanto comprender los requisitos de nuestra aplicación o documentar un proceso.

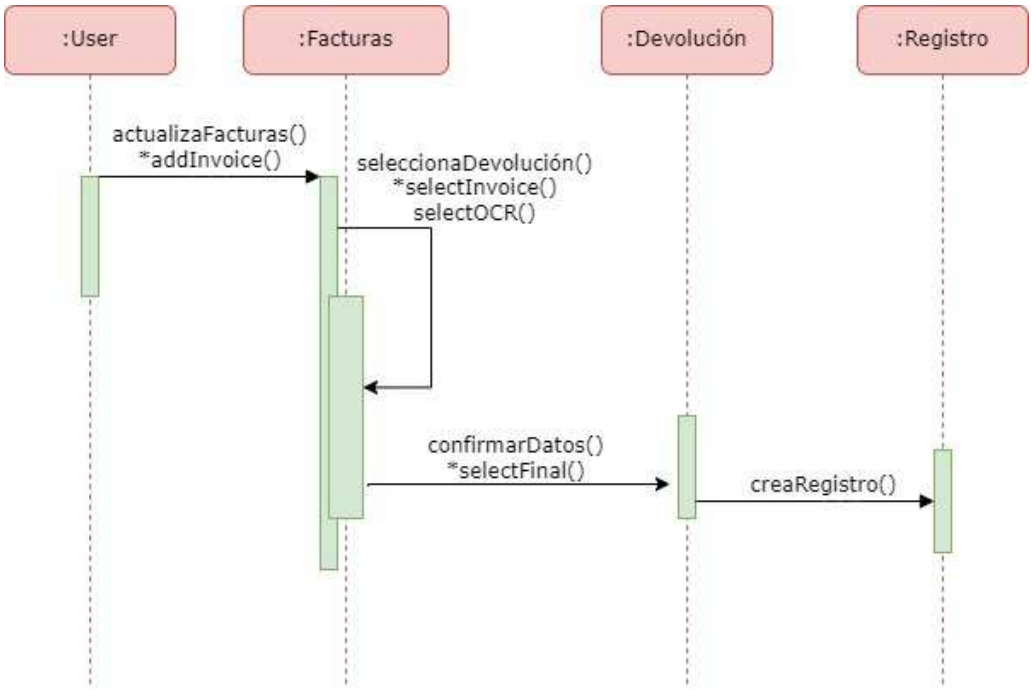


Ilustración 35. Diagrama secuencia general.

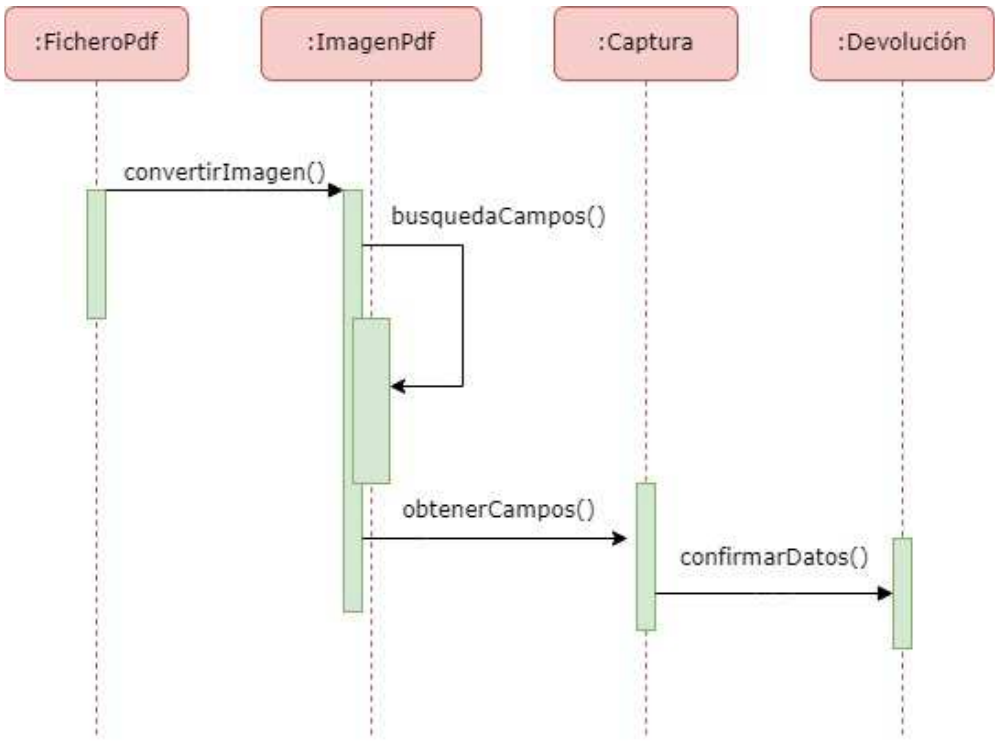


Ilustración 36. Diagrama de secuencia Captura OCR.

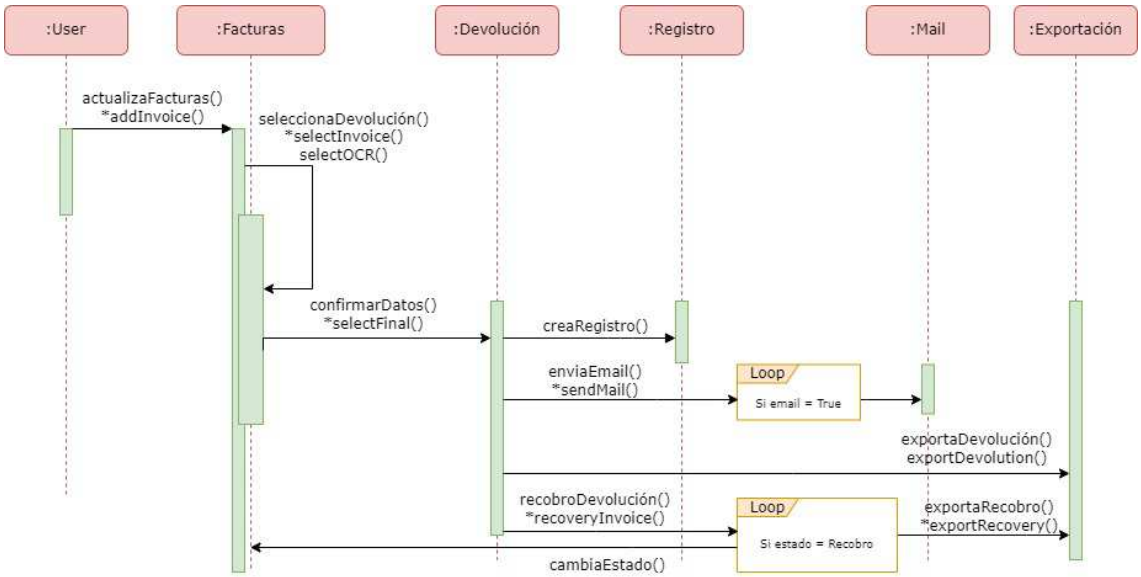


Ilustración 37. Diagrama de secuencia general con envío de email y exportación.

3.8 Diseño de la interfaz

La interfaz gráfica resulta fundamental para facilitar el correcto uso del sistema. Se han diseñado las diferentes pantallas bajo un entorno visual sencillo que permita al usuario una experiencia agradable y fácil de utilizar. De hecho, teniendo en cuenta que el ahorro de costes es una premisa fundamental por parte de los stakeholders, el entorno visual tiene que cumplir que los requisitos de facilidad de uso, evitando así elevados costes de aprendizaje.

En los diseños se ha buscado un diseño regular en todas las pantallas, evitando cambios de colores o formatos. Se ha buscado limitar la cantidad de botones y que el uso sea realmente intuitivo con pocos clicks. Se presentan las diferentes vistas de un posible diseño de la interfaz gráfica y que permite el uso de las funciones diseñadas.

Pantalla principal

Una posible página principal donde se presenta una lista con las facturas importadas para la posterior selección de devoluciones podría ser como sigue:

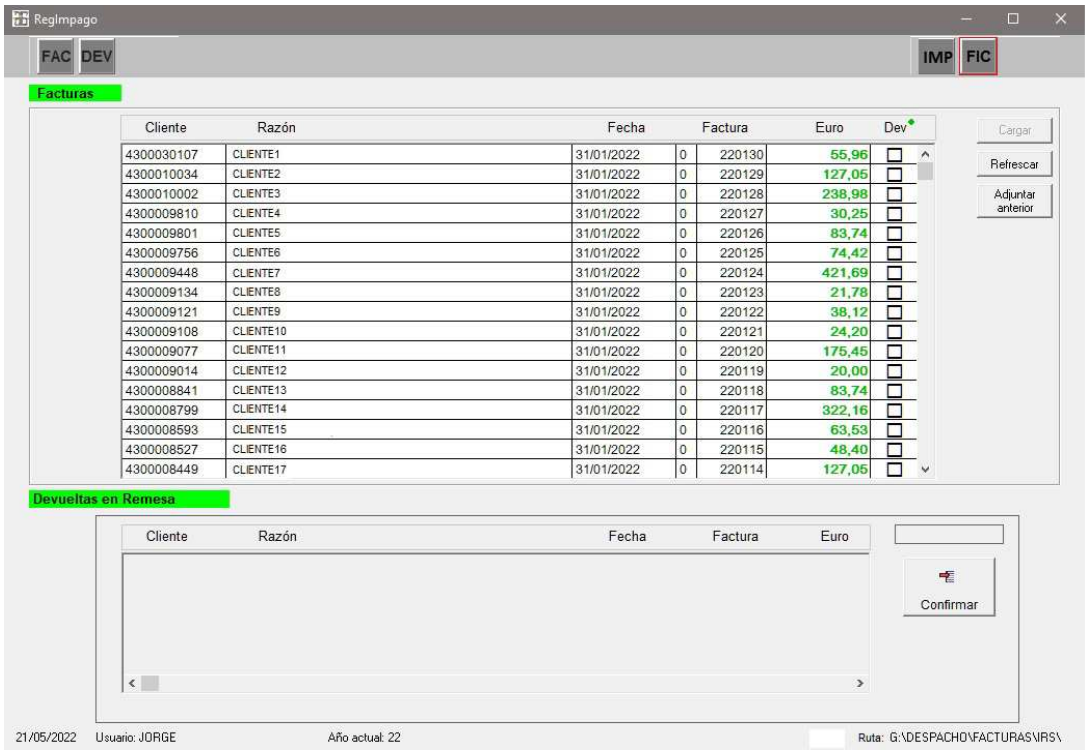


Ilustración 38. Interfaz pantalla principal.

Tras la selección de una factura devuelta, la interfaz solicita resto de la información necesaria para su gestión posterior (fecha de devolución, cuenta contable del banco, …) y se presenta en una lista temporal en la parte inferior que permite su retroceso en el caso de que no se haya confirmado mediante el botón que genera la devolución definitiva (Confirmar):

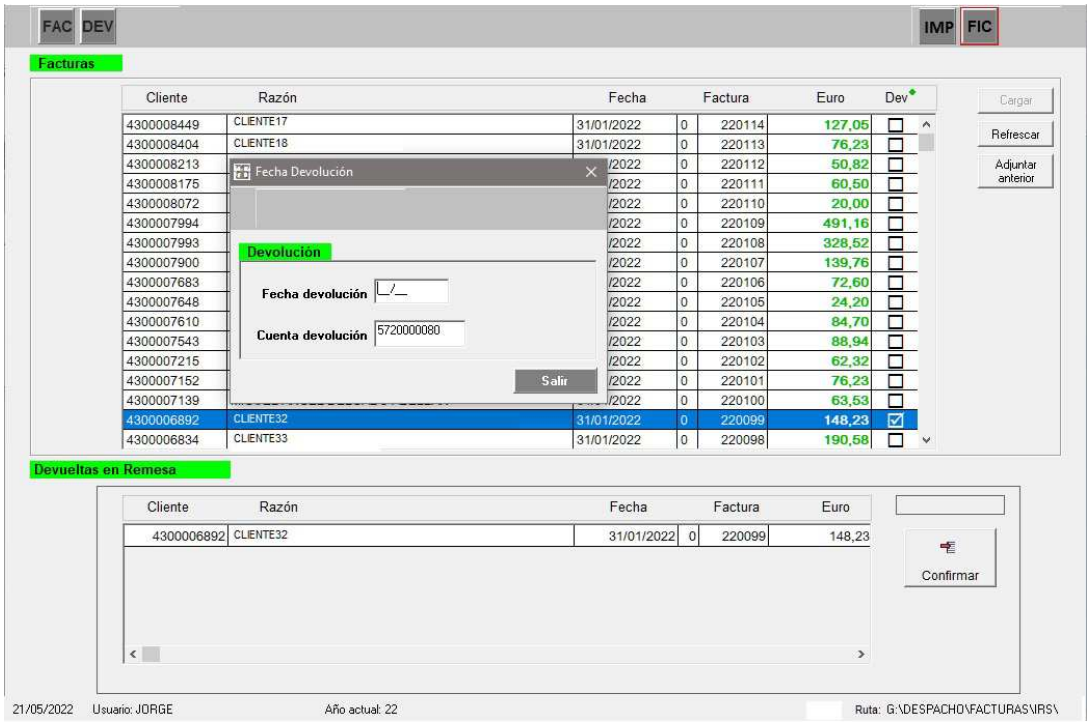


Ilustración 39. Interfaz pantalla principal registrando una devolución.

Pantalla función OCR

Puesto que según el diseño se ha presentado la posibilidad de obtener los datos de devoluciones a partir de los ficheros que la propia entidad envía y que estos no se encuentran en un formato en el que se pueden extraer de forma directa, una posible interfaz de la pantalla que capturaría los datos mediante un sistema OCR podría ser la siguiente:

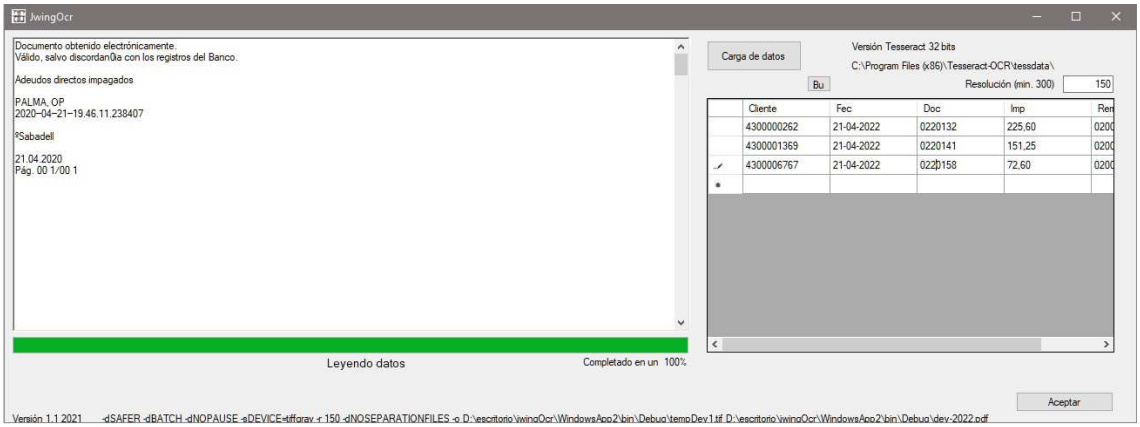


Ilustración 40. Interfaz captura OCR.

Pantalla secundaria de devoluciones

Seguidamente, se adjunta una posible pantalla secundaria donde se encontrarían aquellas facturas devueltas que han quedado registradas de forma definitiva y desde la cual se pueden seleccionar para su registro contable en el programa de destino, previo enlace o generación de fichero de exportación, así como generar un email de información al cliente tras su selección y tras pulsar el botón de “Enviar”:

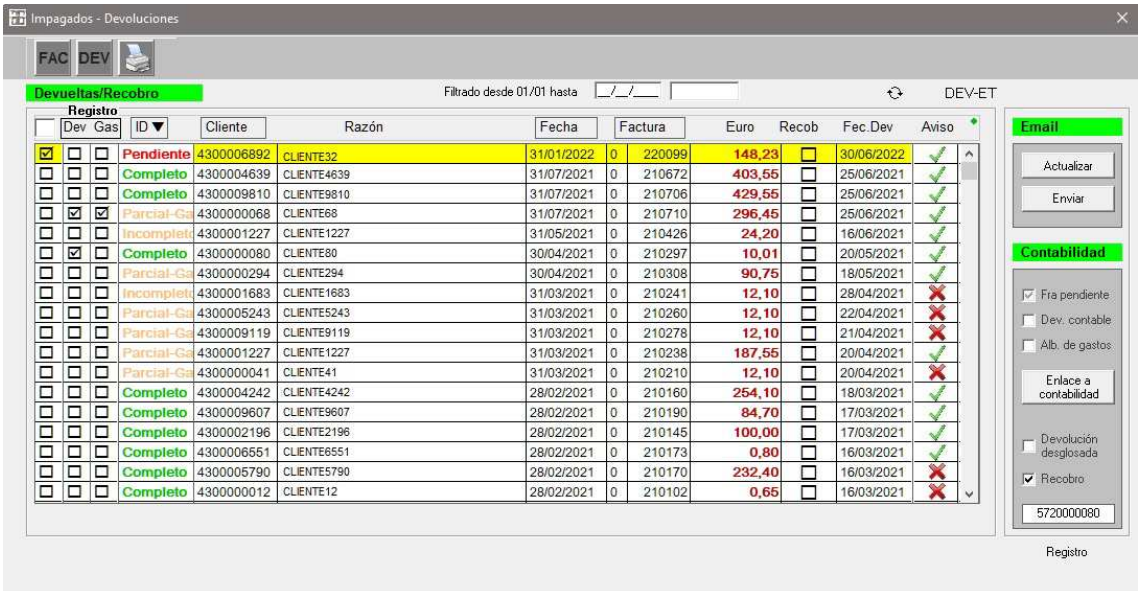
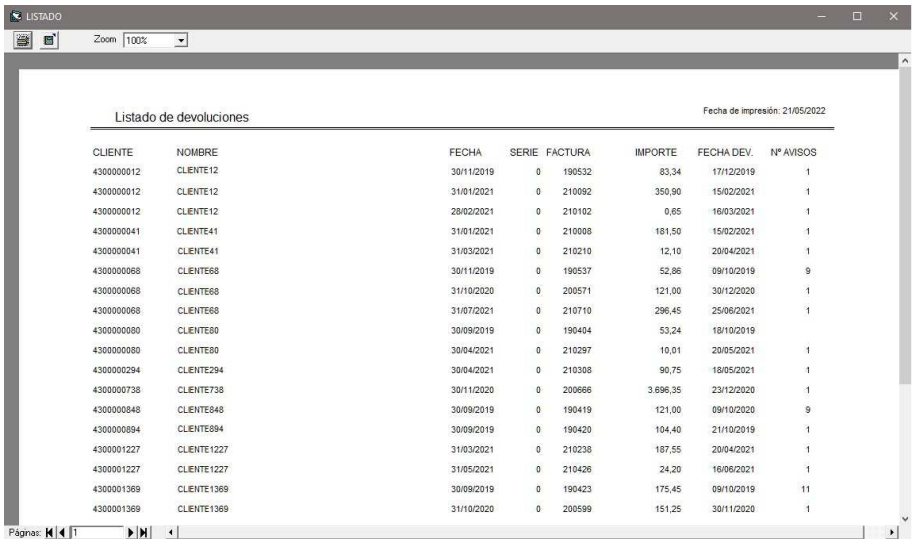


Ilustración 41. Interfaz secundaria de listado de devoluciones.

Pantalla de listados

Desde la aplicación, se ha diseñado la posibilidad de poder obtener listados de las diferentes devoluciones, así como permitir listar diferentes datos relevantes. Una posible pantalla de la interfaz podría ser la que sigue:



CLIENTE	NOMBRE	FECHA	SERIE	FACTURA	IMPORTE	FECHA DEV.	Nº AVISOS
4300000012	CLIENTE12	30/11/2019	0	190532	83,34	17/12/2019	1
4300000012	CLIENTE12	31/01/2021	0	210092	350,90	15/02/2021	1
4300000012	CLIENTE12	28/02/2021	0	210102	0,65	16/03/2021	1
4300000041	CLIENTE41	31/01/2021	0	210008	181,50	15/02/2021	1
4300000041	CLIENTE41	31/03/2021	0	210210	12,10	20/04/2021	1
4300000068	CLIENTE68	30/11/2019	0	190537	52,86	09/10/2019	9
4300000068	CLIENTE68	31/10/2020	0	200571	121,00	30/12/2020	1
4300000068	CLIENTE68	31/07/2021	0	210710	296,45	25/08/2021	1
4300000080	CLIENTE80	30/09/2019	0	190404	53,24	18/10/2019	
4300000080	CLIENTE80	30/04/2021	0	210297	10,01	20/05/2021	1
4300000294	CLIENTE294	30/04/2021	0	210308	90,75	18/05/2021	1
4300000738	CLIENTE738	30/11/2020	0	200666	3.696,35	23/12/2020	1
4300000848	CLIENTE848	30/09/2019	0	190419	121,00	09/10/2020	9
4300000894	CLIENTE894	30/09/2019	0	190420	104,40	21/10/2019	1
4300001227	CLIENTE1227	31/03/2021	0	210238	187,55	20/04/2021	1
4300001227	CLIENTE1227	31/05/2021	0	210426	24,20	16/06/2021	1
4300001369	CLIENTE1369	30/09/2019	0	190423	175,45	09/10/2019	11
4300001369	CLIENTE1369	31/10/2020	0	200599	151,25	30/11/2020	1

Ilustración 42. Interfaz de pantalla de listados.

4. Conclusiones

El diseño de una aplicación desde cero y con el rigor que se necesita exige mucha dedicación. Si bien es cierto que cualquier trabajo conlleva esfuerzo, el hecho de que las tecnologías de la información estén en constante cambio, hace que lo que servía hace poco tiempo, haya quedado desfasado y sea necesario renovarse, lo que supone un extra de esfuerzo. Además, el desarrollo de una aplicación no se trata únicamente de realizar la parte de la implementación, la creación de una interfaz y unas funciones que hagan funcionar la misma, sino que empieza antes, en el diseño.

Este trabajo me ha permitido reafirmar la realidad, que para lograr una aplicación operativa y capaz de mejorarla con el tiempo, es necesario el diseño desde abajo. Tal como una casa se lleva a cabo desde los cimientos, es necesaria la planificación y su replanteo en más de una ocasión, de la misma forma, la creación de una aplicación es similar. Si tal como se dice, “no empieces la casa por el tejado”, este trabajo me ha ayudado a conocer si cabe que es fundamental en primer lugar, empezar con las primeras fases de desarrollo y será “la formación del tejado” o la implementación del software, la fase final.

Las primeras fases para la creación de una aplicación son las relacionadas con su diseño, tanto de la parte arquitectónica, de base de datos y la funcionalidad esperada. Y estas, son en realidad, más importantes para alcanzar los objetivos marcados y no quedarse por el camino. Sin desmerecer ninguna de las fases, en ocasiones, pudiera parecer que con un pequeño boceto es más que suficiente, pero sin duda, he podido aprender que cuanto más detalle se haya logrado y se hayan tenido en cuenta todos los detalles, la improvisación es más probable que no exista, pues al final, es lo que hace que las aplicaciones pierdan eficiencia y robustez si no se hace de la forma correcta.

La idea clara que deseo transmitir y que he aprendido es que no se trata únicamente de hacer que algo funciona y ya está, sino que además exista un orden y coordinación en la parte interior y que no se ve. De hecho, a lo largo de la carrera, a través de las diferentes asignaturas se ensalza esta forma de trabajar. Puede parecer innecesaria, tediosa… pero claramente está justificada y es imprescindible para lograr aspectos como la escalabilidad y el funcionamiento eficiente.

En cuanto al alcance de los diferentes objetivos marcados diría honestamente que se han logrado, aunque se podrían haber desarrollado más si cabe. Se han llevado a cabo todos los objetivos de la parte de diseño, aunque es posible que en alguno se hayan podido mayor justificación de los motivos de decisión o una mayor explicación. Se han respetado las fechas establecidas para la presentación de las diferentes PAC, aunque en especial algunos aspectos marcados para la PAC2, se han realizado finalmente en la PAC3. Por otro lado, pienso que es algo recurrente en los estudiantes pensar que se podría haber logrado un mayor desarrollo y un mayor detalle en alguna fase. El tiempo es algo que escasea y más teniendo que compaginar con muchas otras actividades, como las seglares, familiares y otras de menor importancia, pero que lleva consigo la vida.

Como indicaba en el anterior párrafo, en rasgos generales sí he seguido la planificación y metodología a lo largo del trabajo. Se han alcanzado a tiempo las presentaciones de las PAC y en aquellos aspectos en los que el consultar ha indicado alguna mejoría, finalmente en esta entrega se han tenido en cuenta en la medida de lo posible. Sin duda, la metodología prevista en más que necesaria para evitar dejar toda la carga de trabajo para el final. El hecho de ir redactando y generando con cada entrega parte de la memoria, me ha ayudado a que el trabajo final tanto para la entrega como la presentación no sea algo inalcanzable.

Un cambio para garantizar el éxito, tal vez podría hablar de que a pesar de seguir la planificación y haber ido trabajando a lo largo de estos meses, es que finalmente, es necesario dedicar algo más de tiempo en las últimas semanas para lograr hacer algo que no estamos del todo habituados y es hacer una presentación mediante un vídeo (aunque sea únicamente de diapositivas y audio). A pesar de todo, en la propia planificación, desde la presentación de la PAC3, se había tenido en cuenta la realización de los últimos retoques para lograr la presentación completa (memoria, presentación y formulario de evaluación). Con todo, la elección, búsqueda, instalación y prueba del software necesario para montar la presentación, sí que se ha dejado un poco para última hora. Aunque finalmente, con esfuerzo, se ha conseguido. Sin duda, la presentación pudiera ser mucho más espectacular, cuidada y de mayor impacto, pero pienso que es más importante la manera de presentarla, sin que tampoco sea un conferenciante sumamente elocuente. Pero claramente, no ha sido necesario cambiar la planificación puesto que se han seguido las recomendaciones del consultor y esto ha llevado a obtener los mejores resultados.

Como futuras líneas de trabajo puesto que no ha sido posible asumir en este trabajo sería principalmente la implementación de la aplicación. Aunque el diseño conlleva un esfuerzo elevado, está claro que la implementación también supone mucho trabajo, pero permite apreciar un producto más acabado, completo. La parte de desarrollo es la que llevaría a cabo para quedarme con un mejor “sabor de boca”. Y entre posibles mejoras, sería el desarrollo en un mayor número de plataformas para poder hacer más usable para un mayor número de usuario que trabajan con otros sistemas. Por otro lado, también para ofrecer más facilidades al usuario, también pienso que la importación de los datos externos podría realizarse desde un mayor número de formatos de ficheros. Además, a pesar de que se ha logrado diseñar la funcionalidad pensada en un inicio, una posible mejora y línea de trabajo sería el desarrollo de un bot o robot que permitiera una automatización en los avisos a los clientes hasta su recobro para seguir recordando el impago de la factura.

5. Glosario

[1] **OCR:** El reconocimiento óptico de caracteres (ROC), generalmente conocido como reconocimiento de caracteres y expresado con frecuencia con la sigla OCR (del inglés *Optical Character Recognition*), es un proceso dirigido a la digitalización de textos, los cuales identifican automáticamente a partir de una imagen símbolos o caracteres que pertenecen a un determinado alfabeto, para luego almacenarlos en forma de datos. Así podremos interactuar con estos mediante un programa de edición de texto o similar. [a]

[2] **PYME:** Es el acrónimo utilizado a la hora de hablar de pequeñas y medianas empresas. Estas, generalmente suelen contar con un bajo número de trabajadores y de un volumen de negocio e ingresos moderados en comparación con grandes corporaciones industriales o mercantiles. [b]

[3] **PDF:** (Portable Document Format, 'formato de documento portátil') es un formato de almacenamiento para documentos digitales independientes de plataformas de software o hardware. Este formato es de tipo compuesto (imagen vectorial, mapa de bits y texto). [c]

[4] **API:** La interfaz de programación de aplicaciones, conocida también por la sigla API (application programming interface), es un conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizada por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas de programación. [d]

[5] **Tesseract:** Tesseract es un motor de reconocimiento óptico de caracteres para varios sistemas operativos. Es software libre, liberado bajo la licencia Apache, Versión 2.02 y su desarrollo es financiado por Google desde el 2006. Tesseract fue considerado en 2006 como uno de los motores de OCR de código abierto más precisos disponibles. [e]

[6] **Licencia Apache:** La licencia Apache (Apache License o Apache Software License para versiones anteriores a 2.0) es una licencia de software libre permisiva creada por la Apache Software Foundation (ASF).8 La licencia Apache (con versiones 1.0, 1.1 y 2.0) requiere la conservación del aviso de derecho de autor y el descargo de responsabilidad, pero no es una licencia copyleft, ya que no requiere la redistribución del código fuente cuando se distribuyen versiones modificadas.
Al igual que otras licencias de software libre, todo el software producido por la ASF o cualquiera de sus proyectos está desarrollado bajo los términos de esta licencia, es decir, la licencia otorga al usuario la libertad de usar el software para cualquier propósito, re distribuirlo, modificarlo y distribuir versiones modificadas del software, bajo los términos de la licencia, sin preocuparse de las regalías. [f]

[7] **Metodología ágil:** La metodología ágil constituye un enfoque iterativo de la gestión de proyectos y el desarrollo de software que ayuda a los equipos a proporcionar valor a sus clientes más rápido y con menos dolores de cabeza. En lugar de centrarse en un lanzamiento de gran envergadura, un equipo ágil entrega el trabajo en incrementos pequeños, pero que se pueden consumir. Los requisitos,

los planes y los resultados se evalúan de forma continua, de modo que los equipos disponen de un mecanismo natural para responder con rapidez ante los cambios. [g]

[8] **SCRUM:** Scrum es un marco que permite el trabajo colaborativo entre equipos. Al igual que un equipo de rugby (de donde proviene su nombre) cuando entrena para un gran partido, scrum anima a los equipos a aprender a través de las experiencias, a autoorganizarse mientras aborda un problema y a reflexionar sobre sus victorias y derrotas para mejorar continuamente. [h]

[9] **UML:** Unified Modeling Language o lenguaje unificado de modelado. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados. [i]

[10] **Stakeholders:** Los stakeholders son aquellas personas o entidades que tienen algún impacto o interés en este sistema. A pesar de que todos los usuarios de un sistema son stakeholders, puesto que todos lo usan y, por lo tanto, tienen intereses en él, no todos los stakeholders son usuarios, puesto que puede haber muchas personas o entidades que, a pesar de que no utilicen el sistema, se ven afectadas por su implantación y que, por lo tanto, también tienen intereses en él. [j]

[11] **POO:** La Programación Orientada a Objetos es un paradigma de programación que parte del concepto de "objetos" como base, los cuales contienen información en forma de campos (a veces también referidos como atributos o propiedades) y código en forma de métodos. [k]

[12] **PostgreSQL:** PostgreSQL, también llamado Postgres, es un sistema de gestión de bases de datos relacional orientado a objetos y de código abierto, publicado bajo la licencia PostgreSQL, similar a la BSD o la MIT. [l]

[13] **LOPD-GDD:** La Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales (LOPD-GDD) es una ley orgánica aprobada por las Cortes Generales de España que tiene por objeto adaptar el Derecho interno español al Reglamento General de Protección de Datos. Esta ley orgánica deroga a la anterior Ley Orgánica 15/1999 de Protección de Datos de Carácter Personal3 (aunque se mantiene vigente para la regulación de ciertas actividades). [m]

[14] **Arquitectura 3 capas:** La programación por capas es un modelo de desarrollo software en el que el objetivo primordial es la separación (desacoplamiento) de las partes que componen un sistema software o también una arquitectura cliente-servidor: lógica de negocios, capa de presentación y capa de datos. De esta forma, por ejemplo, es sencillo y mantenible crear diferentes interfaces sobre un mismo sistema sin requerirse cambio alguno en la capa de datos o lógica. [n]

[15] **Escalabilidad:** La escalabilidad es un concepto que hace referencia a la capacidad de una empresa, un proyecto, o incluso un sistema informático, de

alcanzar un crecimiento exponencial. Esto es, una expansión cada vez más acelerada. [ñ]

[16] **MVC:** MVC (Modelo-Vista-Controlador) es un patrón en el diseño de software comúnmente utilizado para implementar interfaces de usuario, datos y lógica de control. Enfatiza una separación entre la lógica de negocios y su visualización. Esta "separación de preocupaciones" proporciona una mejor división del trabajo y una mejora de mantenimiento. Algunos otros patrones de diseño se basan en MVC, como MVVM (Modelo-Vista-modelo de vista), MVP (Modelo-Vista-Presentador) y MVW (Modelo-Vista-Whatever). [o]

[17] **JAVA EE:** Oracle Platform, Enterprise Edition o Java EE (anteriormente conocido como Java 2 Platform, Enterprise Edition o J2EE hasta la versión 1.4; traducido informalmente como Java Empresarial) es una plataforma de programación —parte de la Plataforma Java— para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java. Permite utilizar arquitecturas de N capas distribuidas y se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. La plataforma Java EE está definida por una especificación. Similar a otras especificaciones del Java Community Process (JCP), Java EE es también considerado informalmente como un estándar debido a que los proveedores deben de cumplir ciertos requisitos de conformidad para declarar que sus productos son conformes a Java EE; estandarizado por JCP. [p]

[18] **.NET:** Es una plataforma de desarrollo código abierto que es simplemente una manera de decir que es una colección de lenguajes y bibliotecas que pueden trabajar conjuntamente para crear todo tipo de aplicaciones. [q]

6. Bibliografía

Libros

Jordi Pradel Miquel y José Raya Martos (PID_00230163) Introducción a la ingeniería del software. FUOC

Jordi Pradel Miquel y José Raya Martos (PID_00232012) Requisitos. FUOC

IEEE Std (1993). IEEE Software Engineering Standard: glossary of Software Engineering Terminology. IEEE Computer Society Press.

[j]Jordi Pradel Miquel y José Raya Martos (PID_00230179) Introducción a la ingeniería de requisitos

Antonio Vallecillo Moreno, José Raúl Romero Salguero, Nathalie Moreno Vergara y Francisco Javier Durán Muñoz (PID_00201503) Diseño de aplicaciones distribuidas

Jordi Pradel Miquel y José Raya Martos (PID_00171155) Análisis UML. FUOC

Antonio Vallecillo Moreno, José Raúl Romero Salguero, Nathalie Moreno Vergara y Francisco Javier Durán Muñoz (PID_00201504) Arquitectura del software. FUOC

Jordi Pradel Miquel y José Raya Martos (PID_00165658) Catálogo de patrones. FUOC

Vicenç Font i Sagristà (PID_00187402) Caso práctico de estudio. Diseño. FUOC

Ken Schwaber & Jeff Sutherland, Noviembre 2020 La Guía Scrum (La Guía Definitiva de Scrum: Las Reglas del Juego)

Webs

Wikipedia [en línea] [consulta: 24 de febrero de 2022]. Disponible en: https://es.m.wikipedia.org/wiki/Tesseract_OCR

GFT [en línea] [consulta: 30 de mayo de 2022]. Disponible en: https://blog.gft.com/es/2018/11/06/como-usar-redes-neuronales-lstm-en-la-prediccion-de-averias-en-las-maquinas/

Wikipedia [en línea] [consulta: 26 de marzo de 2022]. Disponible en: https://es.wikipedia.org/wiki/PostgreSQL

Caisistemas [en línea] [consulta: 09 de mayo de 2022]. Disponible en: https://www.caisistemas.es/actualidad/diferencias-software-local-cloud#:~:text=Se%20entiende%20por%20Aplicaciones%20en,Premise%20o%20Aplicaciones%20de%20Escritorio

Postgresql [en línea] [consulta: 27 de mayo de 2022]. Disponible en: https://www.postgresql.org/

Wikipedia [en línea] [consulta: 19 de mayo de 2022]. Disponible en: https://es.wikipedia.org/wiki/Java_EE

Nearsoftjobs [en línea] [consulta: 15 de mayo de 2022]. Disponible en: https://blog.nearsoftjobs.com/patr%C3%B3n-de-dise%C3%B1o-mvc-2366948b5fc7

Lucidchart [en línea] [consulta: 20 de mayo de 2022]. https://www.lucidchart.com/pages/es/diagrama-de-interaccion-uml

Lucidchart [en línea] [consulta: 20 de mayo de 2022]. https://www.lucidchart.com/pages/es/diagrama-de-secuencia

*Disrupciontecnologica (*Diagrama de despliegue*)* [en línea] [consulta: 19 de mayo de 2022]. https://www.disrupciontecnologica.com/capas-y-niveles-diseno-y-confusion/?reload=566246

*Creately (*Diagrama de despliegue*)* [en línea] [consulta: 19 de mayo de 2022]. https://creately.com/blog/es/diagramas/tutorial-de-diagrama-de-despliegue/

Microsoft – Windows Forms [en línea] [consulta: 03 de junio de 2022] https://docs.microsoft.com/es-es/dotnet/desktop/winforms/overview/?view=netdesktop-6.0

[a]*Wikipedia [en línea] [consulta: 01 de junio de 2022] Disponible en:* https://es.wikipedia.org/wiki/Reconocimiento_%C3%B3ptico_de_caracteres

[b]*Economipedia [en línea] [consulta: 01 de junio de 2022] Disponible en:*

https://economipedia.com/definiciones/pyme.html#:~:text=Pyme%20es%20el%20acr%C3%B3nimo%20utilizado,grandes%20corporaciones%20industriales%20o%20mercantiles.

[c]*Wikipedia [en línea] [consulta: 01 de junio de 2022] Disponible en:* https://es.wikipedia.org/wiki/PDF

[d]*Wikipedia [en línea] [consulta: 01 de junio de 2022] Disponible en:* https://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones

[f]*Wikipedia [en línea] [consulta: 01 de junio de 2022] Disponible en:* https://es.wikipedia.org/wiki/Apache_License

[g]*Atlassian [en línea] [consulta: 01 de junio de 2022] Disponible en:* https://www.atlassian.com/es/agile

[h]*Atlassian [en línea] [consulta: 01 de junio de 2022] Disponible en:* https://www.atlassian.com/es/agile/scrum

UML [en línea] [consulta: 01 de junio de 2022] Disponible en: https://www.uml.org/what-is-uml.htm

[i]*Wikipedia [en línea] [consulta: 01 de junio de 2022] Disponible en:* https://es.wikipedia.org/wiki/Lenguaje_unificado_de_modelado

[k]*Wikipedia [en línea] [consulta: 01 de junio de 2022] Disponible en:* https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos

[l]*Wikipedia [en línea] [consulta: 01 de junio de 2022] Disponible en:* https://es.wikipedia.org/wiki/PostgreSQL

[m]*Wikipedia [en línea] [consulta: 01 de junio de 2022] Disponible en:* https://es.wikipedia.org/wiki/Ley_Org%C3%A1nica_de_Protecci%C3%B3n_de_Datos_Personales_y_garant%C3%ADa_de_los_derechos_digitales

[n]*Wikipedia [en línea] [consulta: 01 de junio de 2022] Disponible en:* https://es.wikipedia.org/wiki/Programaci%C3%B3n_por_capas#:~:text=Se%20dice%20que%20la%20arquitectura,tres%20capas%20y%20dos%20niveles.

[ñ]*Economipedia [en línea] [consulta: 01 de junio de 2022] Disponible en:* https://economipedia.com/definiciones/escalabilidad.html

[o]*Mdn [en línea] [consulta: 01 de junio de 2022] Disponible en:* https://developer.mozilla.org/es/docs/Glossary/MVC

[p]*Wikipedia [en línea] [consulta: 01 de junio de 2022] Disponible en:* https://es.wikipedia.org/wiki/Java_EE

[q]*Microsoft [en línea] [consulta: 01 de junio de 2022] Disponible en:* https://docs.microsoft.com/es-ES/shows/net-core-101/what-is-net

7. Anexos