Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN TECNOLOXÍAS DA INFORMACIÓN

Software de gestión de contratación con arquitectura Java EE multicapa para empresas proveedoras de servicios

Estudante: Catarina García Cal

Dirección: Emilio José Padrón González

A Coruña, septiembre de 2022.

A mis padres.

Agradecimientos

Mi gratitud incondicional a todas y cada una de las personas que han colaborado en la realización de este trabajo.

A mi tutor de proyecto, Emilio José Padrón González por aceptar mi propuesta de TFG y conseguir que la llevara a cabo con su continua comprensión, apoyo y motivación para finalizar este trabajo.

A mis amigos que siempre tuvieron palabras de ánimo en los momentos en los que compaginar trabajo y estudios se tornaba un imposible.

A mi familia, por su apoyo y confianza incondicional en todo lo que emprendo.

A todos, gracias.

Resumen

El presente trabajo presenta una aplicación de gestión de contratación basada en código abierto con arquitectura Jakarta Entreprise Edition multicapa, que pueda ser usada por cualquier proveedor de servicios con una cartera de clientes y un catálogo de productos y servicios facturables. Dichos clientes podrán contratar los diferentes productos y servicios del catálogo, a los que se les podrá aplicar descuentos sobre los correspondiente elementos facturables asociados, registrándose dicha relación contractual en la aplicación. Además de la implementación de las distintas contrataciones la aplicación permitirá definir el catálogo de productos y servicios junto con las parametrizaciones pertinentes, como pueden ser los tipos de tasas aplicables sobre las distintas entidades facturables, o los ciclos de facturación asociados a los clientes.

Abstract

This paper presents an open-source-based contract management application with multilayer Jakarta Entreprise Edition architecture, which can be used by any service provider with a client portfolio and a catalog of billable products and services. These customers may contract the different products and services of the catalog, being able to apply discounts on the corresponding associated billable elements, registering the contractual relationship in the application. In addition to the implementation of the different contracts, the application will allow to define the catalog of products and services together with the relevant parameterizations, such as the tax types applicable to the different billable entities, or the billing cycles associated with the clients.

Palabras clave:

Aplicación Web

Jakarta8 EE

Primefaces

PostgreSql

JooQ

Software libre

Keywords:

Web Application

Jakarta8 EE

Primefaces

PostgreSql

JooQ Open Source

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introdución

En economía se entiende por venta «la entrega de un determinado bien o servicio bajo un precio estipulado o convenido y a cambio de una contraprestación económica en forma de dinero por parte de un vendedor o proveedor» [**?**]. Por lo tanto la realización de ventas supone el núcleo de la actividad económica de un gran margen del espectro económico, donde los actores económicos obtienen ganancias dinerarias tras la entrega de un producto o servicio en el que se especializan. El proceso de venta culmina con la concreción de transacción de venta efectiva [**?**], por lo que una vez realizada dicha transacción es necesario disponer de una herramienta que permita registrar dicha venta, esto es, la relación entre lo que se ha vendido, a quién, cuando y por qué importe.

Con el objetivo de diseñar y desarrollar una herramienta que le permita a una empresa gestionar su catálogo de productos y servicios, así como la cartera de clientes y los produtos y servicios que han contratado dichos clientes a lo largo de su vinculación la empresa nace este trabajo.

Hemos optado por referirnos genéricamente a esta herramienta como un software de gestión de contratación, ya que su principal función es la de gestionar la contratación de los productos ofertados por los distintos clientes de la empresa. Evitamos así el uso del del término más familiar de Gestión de relaciones con el cliente o *Customer Relationship Management* (CRM) ya que éste engloba muchas más funcionalidades que las que realiza la herramienta desarrollada: desde un punto de vista informático, un CRM engloba todos los sistemas informáticos de apoyo a la gestión de las relaciones con los clientes, a la venta y al marketing de una empresa, los cuales se integran en los llamados Sistemas de gestión empresarial (SGE).

1.1 Motivación

El proyecto ”Software de gestión de contratación con arquitectura Java EE multicapa para empresas proveedoras de servicios” surge de aunar el deseo de desarrollar una aplicación web

como parte del TFG de la titulación y el conocimiento adquirido durante la labor de soporte realizada en un sistema de facturación de una empresa proveedora de servicios.

Visto a muy alto nivel, los procesos de negocio se pueden definir en tres fases tal y como se muestra en la figura 1.1 (página 2)

Figura 1.1: Fases del proceso de negocio

- Venta: conjunto de actividades orientadas a realizar y formalizar la venta de un servicio a través de la contratación del mismo.
- Provisión: conjunto de actividades cuya finalidad es la de proveer el servicio previamente contratado al cliente.
- Facturación: comprende todas las actividades destinadas a facturar la prestación de los servicios contratados y provisionados y realizar la posterior puesta a cobro de los mismos.

Como ya hemos indicado, la herramienta desarrollada para este Trabajo fin de grado se centrará en la fase de venta, concretamente la correspondiente a registrar la venta y gestionar los productos y servicios contratados así como el catálogo de productos y servicios de la empresa.

1.2 Definición de objetivos

El objetivo primordial del presente proyecto es diseñar y desarrollar una aplicación que permita gestionar el catálogo de productos y servicios de una empresa así como la cartera de clientes y sus contrataciones, para lo que se tendrán en cuenta una serie de elementos parametrizables (esto es, modificables según las necesidades del sistema) que conformarán la definición de las entidades a manejar por el sistema.

Además de realizar las diferentes altas, bajas y modificciones de los elementos que comportan el catálogo de productos y servicios y la cartera de clientes, la aplicación desarrollada deberá gestionar el histórico de datos para los elementos susceptibles de cambio y así poder reflejar los continuos cambios derivados que puedan sufrir (cambio de precio en entidades facturables, modificación del servicio contratado, cambios en las tasas impositivas, etc). También permitirá a los usuarios de la misma la gestión de su cuenta, permitiéndoles cambiar tanto la

contraseña como los datos de contacto. La herramienta a desarrollar será una aplicación web con el objetivo de disminuir el riesgo de incompatibilidades y dependencias de software así como de centralizar su distribución y posibles actualizaciones en un único punto.

Durante el desarrollo de este TFG se ha tenido en mente el sector TIC de las operadoras de telefonía, ya que, como se ha indicado anteriormente, se ha querido aprovechar el conocimiento adquirido a través de la experiencia profesional, aunque podría adaptarse a cualquier otra empresa con necesidades similares.

Capítulo 2

Proceso de negocio. Fundamentos teóricos

A día de hoy es innegable el papel crucial que juegan las tecnologías de la información en general y los sistemas de información en particular dentro de la gestión de procesos de negocio de cara a la consecución de objetivos de forma eficaz y eficiente.

El mercado está en continua evolución. Los sistemas de información nos proporcionan la tecnología básica necesaria tanto para crear nuevas funcionalidades como para adaptar las existentes para poder satisfacer las nuevas necesidades que van surgiendo.

Este capítulo pretende ser una aproximación a los conceptos teóricos en los que se enmarcan los procesos empresariales dentro del sector Tecnologías de la Información y la Comunicación (TIC).

2.1 Proceso de negocio

Hammer & Champy (1993) [**?**] definieron el proceso de negocio como:

> Una colección de actividades que toman uno o más tipos de entradas y crean una salida que tiene valor para el cliente.

Así pues, podemos entender un proceso de negocio como un conjunto de funciones dentro de una secuencia específica que proporciona valor a un cliente interno o externo y cada función dentro de un proceso puede ser a su vez interpretado como un proceso en si mismo llamado sub-proceso. El desencadenante de este subproceso es el proceso previo (o por el evento inicial que da comienzo al proceso en general) y devuelve un resultado de valor para el siguiente proceso o para el cliente final y sus procesos si este es el último suproceso de un proceso de extremo a extremo [**?**].

Esta descomposición de procesos en subprocesos puede extenderse tanto como las funciones resultados sigan teniendo sentido desde el punto de vista de negocio: por ejemplo el subproceso ”gestión de ventas” puede describirse en detalle usando funciones como ”registro de orden de venta” o ”asignación de productos a la orden de venta”, sin embargo descomponer la función ”registro de orden de venta” en actividades como ”introducir nombre de cliente”, ”introducir dirección de cliente”, etc. no aportan relevancia alguna desde el punto de vista de negocio.

Con el objetivo de poder definir de forma sencilla los procesos más complejos sin dejar de lado ningún aspecto importante de los mismos surge la Arquitectura de sistema de información integrada o *Architecture of Integrated Information Systems* (ARIS), una aproximación al proceso de arquitectura y modelado empresarial desarrollado por August-Wilhem Scheers y que permite describir un proceso de negocio desde cinco puntos de vista diferentes, dando respuesta a todas las cuestiones relevantes relativas al proceso:

- Vista organizacional: ¿quién (gente, departamentos, empresas, etc.) está involucrado en el proceso?
- Vista funcional: ¿qué funciones se llevan a cabo dentro del proceso?
- Vista de datos: ¿qué datos o información es necesaria/producida para/en el proceso?
- Vista de entrega: ¿qué son los entregables del proceso y por qué los necesito?
- Vista de control: el cómo se ve en conjunto refleja el quién está haciendo qué, qué datos producen entregables y cual es la secuencia lógica que hace que las funciones lleven a cabo su función.

La figura 2.1 (página 5) muestra la arquitectura ARIS:

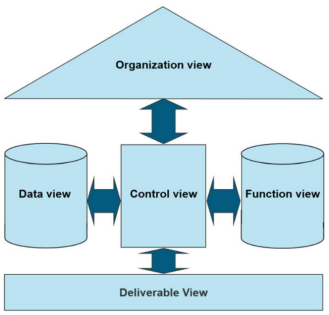


Figura 2.1: Arquitectura de sistema de información integrada (ARIS)

El elemento más importante de la arquitectura ARIS es la vista de control. Esta muestra como dos o más aspectos de un proceso *encajan*, por ejemplo quién es responsable de una función determinada o qué función usa determinados datos. La vista resultado de la integración

de varios aspectos del proceso de negocio es la llave para la gestión exitosa de dichos procesos. ARIS ayuda a convertir los procesos en algo *tangible* al definir cómo describirlos asentando la base para convertir la Gestión de procesos de negocio o *Business Process Management* en una disciplina de gestión real.

2.1.1 Gestión de proceso de negocio

Un Gestión de procesos de negocio o *Business Process Management* (BPM) incluye conceptos, métodos y tecnicas para soportar el diseño, la administración, la configuración, la representación y el análisis de los procesos de negocio [**?**].

La base del BPM es la representación explícita de los procesos de negocio con sus actividades y las restricciones de ejecución de los mismos.

El ciclo de vida del BPM consta de cinco fases [**?**] tal y como se muestra en la figura 2.2 (página 6):

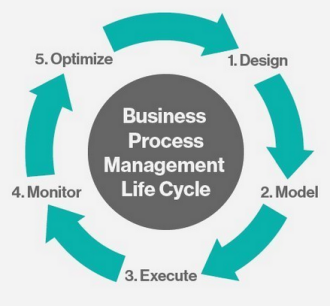


Figura 2.2: Ciclo de vida del BPM

1. Diseño: el diseño de procesos abarca tanto la identificación de procesos existentes como el diseño de procesos ”futuros”. El objetivo de este paso es garantizar un diseño correcto y eficiente. Debe haber una sincronización entre los procesos existentes y el diseño de un nuevo proceso para evitar posibles interrupciones.

2. Modelado: se crea una representación visual del modelo de proceso. Esto debe incluir detalles específicos, como líneas de tiempo, descripciones de tareas y cualquier flujo de datos en el proceso.

3. Ejecución: consiste en una prueba de concepto, probando el nuevo sistema BPM con un grupo limitado. Después de incorporar cualquier comentario, el equipo puede comenzar a implementar el proceso a un público más amplio.

4. Supervisión: durante esta etapa, se supervisa el proceso, se miden las mejoras en la eficiencia y se identifica cualquier cuello de botella adicional.

5. Optimización: se realizan los ajustes finales al proceso para mejorar la actividad empresarial.

Entre las ventajas que proporcionan los BPM se encuentran las siguientes:

- Mayor eficiencia y ahorro de costes: gracias a la optimización de los procesos existentes y a la incorporación de más estructura en el desarrollo de nuevos procesos, eliminando redundancias y cuellos de botella.
- Mejor experiencia de empleados y clientes: ya que permite eliminar el trabajo repetitivo y aumentar la accesibilidad de la información mejorando la productividad y la implicación con el cliente.
- Procesos más escalables: a través de la mejora en la ejecución de procesos y la automatización de flujos de trabajo se facilita el escalado de procesos a otras zonas geográficas de todo el mundo.
- Mayor transparencia: puesto que la automatización de procesos de negocio define claramente a los propietarios para las tareas a lo largo del proceso, se proporciona más transparencia y responsabilidad a lo largo de un proceso determinad, favoreciendo la comunicación entre equipos.

2.2 Sistemas de soporte operacional y Sistemas de soporte de negocio

Centrándonos en el área de las telecomunicaciones, nos encontramos con dos nuevos conceptos:

Sistemas de soporte operacional u *Operational Support Systems* (OSS). Se trata de un término usado para describir los sistemas de procesado de información utilizados por las operadoras para gestionar sus redes de comunicación. Originalmente conocidas como herramientas de Gestión de redes de telecomunicaciones o Telecommunication Network Management tools actualmente estas soluciones son mucho más sofisticadas, permitiendo coordinar clientes, servicios, recursos, procesos y actividades. Ayudan a las operadoras a diseñar, construir, operar y mantener las redes de comunicaciones.

Sistemas de soporte de negocio o *Business Support Systems* (BSS) es el término tradicionalmente utilizado para describir las funcionalidades de negocio y/o orientada al cliente. Estas herramientas permiten que una organización contacte con sus clientes (por ejemplo a través del CRM), crear ofertas para ellos, emitir facturas así como transacciones entre operadores de comunicaciones (liquidaciones, punto de interconexión).

De forma conjunta OSS y BSS permiten a los operadores de red ofrecer servicios de manera eficiente y confiable a un enorme número de suscriptores en algunas de las máquinas más complejas del mundo, las redes globales de telecomunicaciones.

Existen distintos estándares para aportar consistencia entre las distintas aplicaciones existentes. Comentaremos brevemente algunas de ellas.

2.2.1 TMN de ITU-T

Las Comisiones de Estudio del Sector de normalización de las telecomunicaciones de la UIT o *ITU Telecommunication Standardization Sector* (ITU-T) reúnen a expertos de todo el mundo para elaborar documentos de especificación de telecomunicaciones y protocolos informáticos, conocidos como Recomendaciones ITU-T, que actúan como elementos definitorios de la infraestructura mundial de las Tecnologías de la Información y la Comunicación (TIC) [**?**].

Los esfuerzos de normalización de la ITU-T comenzaron en 1865 con la formación de la International Telegraph Union (en español Unión Telegráfica Internacional), que más tarde se convirtió en la Unión Internacional de Telecomunicaciones o *International Telecommunication Union* (ITU). La ITU se convirtió en un organismo especializado de las Naciones Unidas en 1947. El Comité Consultivo Internacional de Telégrafos o *Comité Consultatif International Téléphonique et Télégraphique* (CCITT) fue creado en 1956, y pasó a llamarse ITU-T en 1993.

El modelo tecnológico TMN se presenta en la recomendación M.3010 y suele referenciarse como la pirámide TMN (figura 2.3, página 9). Aborda la complejidad de la gestión de las telecomunicaciones a través de una arquitectura lógica por capas, la cual organiza las funciones en grupos denominados capas lógicas y describe las relaciones entre las mismas. Una capa lógica refleja aspectos particulares de la gestión e implica el agrupamiento de la información de gestión relativa a ese aspecto. Estas capas son las siguientes:

- Capa de gestión de elementos. Proporciona definición y coordinación de una colección de dispositivos de red, aunque sea un subconjunto de toda la red. Esta capa normalmente incluiría la consolidación de la administración de alarmas, la copia de seguridad, el registro y el mantenimiento de los sistemas que admiten los dispositivos de red.

- Capa de gestión de red. Proporciona la vista de administración general de la red como una suma de partes componentes. Es responsable de la supervisión, configuración y control de extremo a extremo de la red.

- Capa de gestión de servicios. Responsable de definir los servicios ofrecidos por las operadoras de comunicaciones. Esto proporciona la interfaz entre los servicios de un cliente y la red, incluyendo la definición, la administración y la carga.

- Capa de gestión empresarial. Representa la funcionalidad relacionada con la planificación estratégica del negocio, como las tendencias, la calidad, etc., y proporciona la base para la facturación, el presupuesto y el establecimiento de objetivos.

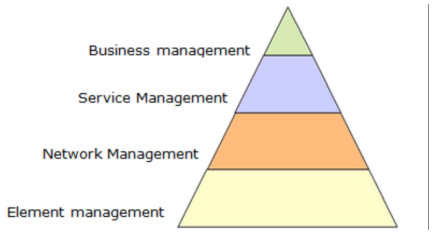


Figura 2.3: Modelo tecnológico TMN

Las áreas funcionales de gestión presentadas en la recomendación M.3400 incluyen entre otras las siguientes áreas:

1. Administración de clientes.
2. Gestión de aprovisionamiento de red.
3. Administración de tarifas, cobros y contabilidad.
4. Administración de Medición y Análisis de Tráfico
5. Gestión del tráfico.
6. Administración de Enrutamiento y Análisis de Dígitos
7. Administración de Seguridad.

2.2.2 TMF

Foro de Gestión de Telecomunicaciones o *Telecommunication Management Forum* es una alianza compuesta por más de 850 compañías globales que trabajan conjuntamente proporcionando un entorno abierto y colaborativo y un soporte práctico que permite a las compañías proveedoras de servicios y sus distribuidores transformar rápidamente sus operaciones comerciales, sistemas de TI y ecosistemas para capitalizar las oportunidades presentadas en un mundo digital en rápida evolución [**?**].

TM Forum fue fundada como OSI/Network Management Forum (en español Foro de Gestión de Redes/OSI) en 1988 por ocho empresas para resolver de forma conjunta los problemas de gestión operativa y de sistemas con los protocolos OSI. En 1998 el nombre fue cambiado a TeleManagement Forum. En 2008, la organización cambió su nombre a TM Forum.

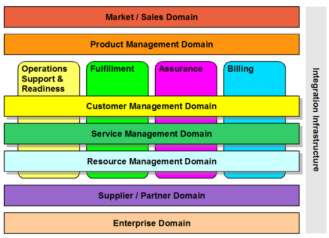
A través de estas colaboraciones los modelos de TMN han ganado un uso generalizado en el campo de las implementaciones de OSS. Dentro de los trabajos desarrollados por el TM Forum encuentra el Frameworx. Se trata un marco de arquitectura empresarial orientado a los proveedores de servicios de comunicaciones que consta de 4 marcos de trabajo o frameworks:

1. Un modelo de aplicación (Mapa de aplicaciones de telecomunicaciones o *Telecom Applications Map* (TAM)). Proporciona un marco modular de bloques funcionales de gestión. Esto ayuda a proporcionar una mayor consistencia (y compatibilidad) entre los conjuntos de productos de diferentes proveedores (figura 2.4a, página 10).

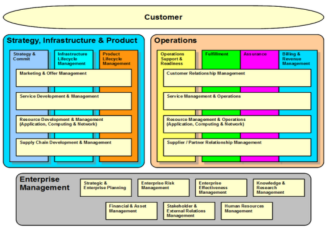
2. Un modelo de proceso (Mapa de Operación de Telecomunicaciones mejorado o *enhanced Telecom Operation Map* (eToM)). Tiene como objetivo proporcionar un lenguaje común y un catálogo de procesos de negocio utilizados en entornos de telecomunicaciones. Este nivel de normalización tiene por objeto simplificar las líneas de comunicación entre los proveedores de servicios y los integradores de sistemas asociados (figura 2.4b, página 10).

3. Un modelo de información (Modelo de información/datos compartidos o *Shared Information/Data model* (SID)). Su objetivo es definir las entidades esenciales, las relaciones y los atributos de los objetos de datos que prevalecen en las aplicaciones/bases de datos de telecomunicaciones. También proporciona un lenguaje común para su uso por los desarrolladores/integradores de OSS. (TM Forum) también ha tratado de desarrollar API estandarizadas basadas en SID (interfaces de programación de aplicaciones) como OSS/J para acelerar la integración de sistemas dispares.

4. Un marco de integración de sistemas (Arquitectura tecnológica neutral o *Technology Neutral Architecture* (TNA)). Tiene como objetivo proporcionar estandarización arquitectónica, sin dejar de ser tecnológicamente neutral, incluyendo interfaces, mecanismos y políticas comunes. La integración también se conoce comúnmente como el (Programa de integración del foro TM o *TM Forum Integration Project* (TIP))



(a) Modelo de proceso TAM



(b) Modelo de proceso eToM

Figura 2.4: Modelos de referencia del marco de trabajo de TMN

Además de estos marcos de trabajo (TM Forum), el (Foro de Gestión de Telecomunicaciones o *Telecommunication Management Forum*) cuenta con la suite TM Forum Open API, que tiene más de 50 API basadas en REST (y muchas más en desarrollo) y con un el (TM Forum Open Digital Architecture (ODA)), un estándar en evolución que tiene la intención de establecer una nueva visión para OSS/BSS que parecen estar ganando uso generalizado en el mundo de las telecomunicaciones.

2.2.3 ITIL e ITSM

Information Technology Infrastructure Library o *Information Technology Infrastructure Library* (ITIL) describe procesos, procedimientos, tareas y listas de verificación que no son ni específicos de la organización ni de la tecnología, pero que pueden ser aplicados por una organización hacia la estrategia, la entrega de valor y el mantenimiento de un nivel mínimo de competencia. Se creó originalmente en la década de 1980 como una colección de libros, cada uno de los cuales cubría una práctica específica de gestión de servicios de TI. Fue desarrollado por el Gobierno del Reino Unido en un intento de estandarizar los procesos de TI, en particular el traspaso de la implementación a los entornos de soporte de TI en curso. Entre 1986 y 1996, esta colección aumentó a más de 30 volúmenes. Durante los años 2000 y 2001 estos volúmenes se consolidaron en 9 conjuntos que agrupaban temas relacionados [**? ?**].

Gestión de servicios de tecnología de la información o *Information Technology Service Management* (ITSM) es un conjunto de políticas, procesos y procedimientos para gestionar la implementación, mejora y soporte de servicios de TI orientados al cliente. A diferencia de otras prácticas de gestión de TI que se centran en hardware, redes o sistemas, ITSM es un enfoque más centrado en los procesos y las personas, teniendo como objetivo la mejora constante el servicio al cliente de TI de forma alineada a los objetivos comerciales. Las empresas que utilizan ITSM consideran la TI como un servicio, con un enfoque en la prestación de servicios valiosos a los clientes, en lugar de un departamento que administra la tecnología.

Los principales elementos de ITSM son:

- Gestión de incidencias.
- Gestión de problemas.
- Gestión del cambio.
- Gestión de proyectos.
- Gestión de activos.
- Conocimiento, política y procedimiento.

- Catálogo de servicios.
- *Service Desk.*

Aunque a veces se usan indistintamente, ITSM e ITIL no son lo mismo: ITIL es uno de los marcos más populares dentro de la disciplina ITSM, y ha ayudado a informar e inspirar otros marcos ITSM. Por lo tanto ITIL es la base en la que se apoya ITSM proporcionando a las empresas la metodología necesaria para adoptar e implementar ITSM (figura 2.5, página 12)

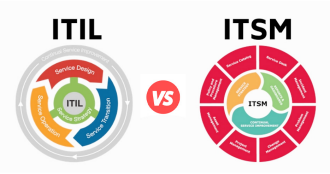


Figura 2.5: ITIL y ITSM

Por otro lado, de la misma forma que existe una una superposición cada vez mayor entre las TI y las tecnologías de redes de telecomunicaciones, también hay una mayor prevalencia de marcos de TI (como ITIL) que se utilizan junto con los marcos de telecomunicaciones (como eToM). Los proveedores de servicios de telecomunicaciones dependen de una infraestructura de TI altamente confiable (por ejemplo, servidores, etc.) e ITIL se considera actualmente como la mejor práctica para la gestión de este tipo de activos. Del mismo modo, los clientes están subcontratando la gestión de la infraestructura de TI y telecomunicaciones de extremo a extremo, exigiendo a sus proveedores de servicios que suministren una gestión de servicios de alta calidad de la red y los activos de TI cada vez más críticos del cliente.

eToM comparte algunos puntos en común y varias diferencias con ITIL. TM Forum ha publicado una nota de aplicación eTOM – ITIL (GB921L) titulada ”Using eTOM to model the ITIL Processes”. Esta nota proporciona orientación sobre el modelado de la gestión de servicios de TI utilizando los elementos de proceso estándar dentro de eToM así como una superposición detallada de procesos ITIL con procesos eToM (figura 2.6, página 12)

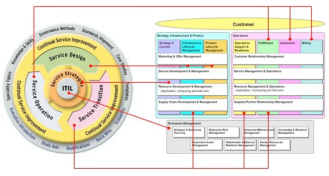


Figura 2.6: Aproximación al mapeo entre eToM e ITIL [**?**]

2.3 Gestión de relaciones con los clientes

La Gestión de relaciones con el cliente o *Customer Relationship Management* (CRM) es una estrategia comercial que optimiza los ingresos y la rentabilidad al tiempo que promueve la satisfacción y la lealtad del cliente. Las tecnologías CRM permiten la estrategia, e identifican y gestionan las relaciones con los clientes [**?**]. El compendio de conceptos, procedimientos y reglas que una corporación sigue al comunicarse con sus consumidores se conocen como CRM [**?**] y por definición cubre todas las formas en las que se administran las relaciones con los clientes en los distintos ámbitos empresariales: ventas, marketing, servicio al cliente y comercio electrónico [**?**] .

Por lo general, cuando hablamos de CRM estamos refiriéndonos a los sistemas de software que nos ayudan a construir estrategias de negocio para retener y captar clientes. El software Gestión de relaciones con el cliente o *Customer Relationship Management* (CRM) permite automatizar e integrar estas actividades orientadas al cliente e incluso pueden ofrecer herramientas para el análisis de clientes, la personalización, las redes sociales, la colaboración y mucho más. La funcionalidad que el software CRM proporciona a las empresas se divide en cuatro segmentos: ventas, marketing, servicio al clientes y comercio digital [**?**].

2.3.1 Breve historia del CRM

La necesidad de mantener registros relativos a la información comercial no es algo nuevo. De hecho podríamos decir que existe desde que existe el comercio: con el tiempo el saber quien poseía qué o quien debía a quién requiería alguna forma de notación y registro permanente, así como alguna forma de contabilidad para la gestión de los bienes. El tratamiento de esta información fue evolucionando adaptándose a las necesidades que iban surgiendo en cada momento como pudiera ser la segmentación de clientes atendiendo a sus capacidad de pago o a su nivel económico, permitiendo aplicar distintas estrategias comerciales en función de la información almacenada.

La aparición del Rodolex (palabra compuesta a partir de las palabras *rolling* e *index*) en la década de los 50 (figura 2.7 , página 14) supuso un gran avance en el tratamiento de los clientes. Este dispositivo giratorio que permitía almacenar tarjetas indexadas de quita y pon que contenían la información relevante de distintas personas y empresas, resultó ser una forma mucho más rápida, cómoda y eficaz de acceder y/o modificar dicha información que los métodos existentes hasta el momento (principalmente la búsqueda de información en libros de contabilidad y archivos personales del cliente). Este término se ha vuelto algo genérico para nombrar cualquier organizador personal que realice esta función, o como una metonimia para el total de los contactos comerciales acumulados de un individuo.

Figura 2.7: Rodolex

En la década de los 60 comienza la introducción de las computadoras en las empresas. Los sistemas de mainframe independientes se hicieron ampliamente disponibles para las empresas lo que les permitiría manejar bases de datos de los clientes y aplicar un nivel (básico) de automatización, ayudando a mantener registros contables. Inicialmente la gestión de clientes pertenecía al departamento de cuentas, pero esta automatización pronto se extendió a otros departamentos, incluidos ventas y marketing. En la década de los 70 el precio de los ordenadores disminuyó drásticamente, lo que permitió que hasta las pequeñas empresas se sumaran al carro de la revolución informática y a partir de ahí vino la siguiente evolución del CRM.

Los inicios del CRM tal como lo conocemos comenzaron en la década de 1980. En esta época se produzco el cámbio del márketing directo al márketing de base de datos. Esto marca el comienzo de la combinación de datos de clientes con estrategia de ventas, una parte central de CRM tal como lo conocemos hoy [**?**]:

- Marketing directo: Cualquier práctica de marketing que implique comunicación directa o interacción con clientes individuales. Por ejemplo, a través de la comunicación postal directa como catálogos y cartas.
- Marketing de base de datos: Una forma de marketing directo que implica recopilar datos de clientes, analizar estos datos y usarlos para crear mensajes de marketing personalizados y predecir las acciones y deseos de nuevos clientes potenciales. Se trata de utilizar datos para optimizar los esfuerzos de marketing.

Robert y Kate Kestnbaum fueron pioneros del marketing de bases de datos. Se trataba de una forma de marketing directo que analizaba estadísticamente la base de datos de clientes para identificar qué clientes tendrían más probabilidades de reaccionar a una campaña de marketing. El concepto despegó y Kestnbaum, junto con Robert Shaw, nos trajo nuevos conceptos y metodologías, que van desde el valor de por vida del cliente hasta la gestión del canal.

En 1986, Pat Sullivan y Mike Muhney lanzaron un sistema de evaluación de clientes llamado ACT! (acrónimo de "Automated Contact Tracking" - Seguimiento automatizado de contactos en español) basado en el principio de Rolodex digital, que ofrecía por primera vez un servicio de gestión de contactos que facilitaba el almacenamiento de información de contactos (nombres, direcciones, números de teléfono, etc.), lo que podría considerarse como el primer CRM automatizado [**?**].

A medida que las oficinas se digitalizaron a lo largo de la década de 1990 surgieron una gran cantidad de nuevos productos de gestión de datos de clientes. Estos productos se definían como Automatización de la fuerza de ventas o *Sales Force Automation* (SFA) y eran una amalgama de márketing de base de datos y gestión de contactos. En 1993, Tom Siebel diseñó el primer producto de CRM, Siebel Customer Relationship Management. Con el fin de competir con estas novedosas soluciones de CRM, las empresas de Software de planificación de recursos empresariales o *Enterprise resource planning software* (ERP)[1] también vieron una oportunidad y el mercado se volvió muy competitivo. A mediados de los años 90, este mercado se disparó en ofertas de productos de todas las formas y tamaños, ahora conocidas como sistemas CRM. La gestión de relaciones con los clientes se popularizó en 1997, debido al trabajo de Siebel, Gartner e IBM. Entre 1997 y 2000, los principales productos de CRM. se enriquecieron con capacidades de envío y marketing. Siebel introdujo la primera aplicación de CRM. móvil llamada Siebel Sales Handheld en 1999. La idea de una base de clientes independiente y alojada en la nube pronto fue adoptada por otros proveedores líderes en ese momento, incluidos PeopleSoft (adquirido por Oracle), Oracle, SAP y Salesforce.com.

Y así llegamos a nuestros días, en los que el mercado de nuevos productos CRM no parece haber alcanzado su punto de saturación. Las nuevas empresas continúan llegando al mercado con productos en la nube, mientras que los proveedores existentes han cambiado sus modelos de licencia para ofrecer alternativas en la nube a las licencias de sitios tradicionales. El último cambio es el aumento de los datos sociales y la necesidad de interactuar con los clientes en las diversas plataformas sociales. El móvil se ha vuelto aún más imperativo como una oferta con la llegada del teléfono inteligente. El ritmo del cambio es tan rápido que muchos proveedores están luchando para mantenerse al tanto de los últimos desarrollos, que van desde chatbots hasta big data e IA. Si bien se podría esperar que los productos de CRM hayan madurado, los clientes aún experimentan dificultades para lograr implementaciones exitosas. Esto se debe a que ellos también están luchando por mantener sus modelos de negocio relevantes y sostenibles en esta era disruptiva.

[1] ERP: sofware que se ocupa de la gestión de los principales procesos comerciales, incluidas las áreas relacionadas con las relaciones con los clientes

Capítulo 3

Fundamentos Técnicos

En este capítulo realizaremos una aproximación a las diferentes tecnologías usadas en el desarrollo del presente TFG.

3.1 Jakarta EE

Jakarta Entreprise Edition es un conjunto de especificaciones que amplía Java Standard Edition con especificaciones para características empresariales como pueden ser la informática distribuida y los servicios web.Las aplicaciones de Jakarta EE manejan transacciones, seguridad, escalabilidad, concurrencia y administración de los componentes que está implementando. Entre las áreas en las que se utiliza Jakarta EE se encuentran el comercio electrónico, la contabilidad o los sistemas de información bancaria [**?**].

Jakarta EE ofrece a los desarrolladores un conjunto completo de especificaciones abiertas y neutrales para proveedores que se utilizan para desarrollar aplicaciones Java modernas y nativas de la nube desde cero. Ofrece una combinación de ventajas que no se pueden encontrar en otros marcos Java: Madurez, estabilidad y retrocompatibilidad. Su flexibilidad arquitectónica permite arquitecturas de microservicios basadas en la nube, así como arquitecturas tradicionales y monolíticas. Se rata de una plataforma con todas las funciones que se puede configurar en solo unas pocas docenas de líneas de código y con capacidad de cambiar fácilmente las tecnologías subyacentes para cumplir con los nuevos requisitos y aprovechar las implementaciones más rápidas y ligeras. Todo esto brinda a los desarrolladores y empresas de Java todo lo que necesitan para construir y evolucionar aplicaciones Java nativas de la nube hoy y en el futuro [**?**].

3.1.1 Historia

El lenguaje de programación Java nació en 1991 de la mano de James Gosling, que en aquellos momentos trabajaba en Sun Microsystems, y su equipo. Los objetivos de Gosling eran

implementar una máquina virtual y un lenguaje con una estructura y sintaxis similar a C++. La promesa inicial de Gosling era «*Write Once, Run Anywhere*» (Escríbelo una vez, ejecútalo en cualquier lugar), proporcionando un lenguaje independiente de la plataforma y un entorno de ejecución ligero y gratuito para las plataformas más populares (la Máquina virtual de java o *Java Virtual Machine* (JVM)), de forma que los binarios (bytecode) de las aplicaciones Java pudiesen ejecutarse en cualquier plataforma. En 1995 fue lanzado comercialmente por Sun Microsystems como un componente fundamental de la plataforma Java [**?**].

La plataforma Java es un conjunto de programas que facilitan el desarrollo y la ejecución de programas escritos en el lenguaje de programación Java [**?**]. Una plataforma Java incluye un motor de ejecución (la JVM), un compilador y un conjunto de bibliotecas. También puede haber servidores adicionales y bibliotecas alternativas que dependen de los requisitos. Las plataformas Java se han implementado para una amplia variedad de hardware y sistemas operativos con el fin de permitir que los programas Java se ejecuten de manera idéntica en todos ellos. Existen distintas plataformas orientadas adistintas clases de dispositivos y aplicaciones, a saber:

- Java Card: Una tecnología que permite que las pequeñas aplicaciones basadas en Java (applets) se ejecuten de forma segura en tarjetas inteligentes y dispositivos similares de pequeña memoria.
- Java ME (Micro Edition): especifica varios conjuntos diferentes de bibliotecas (conocidas como perfiles) para dispositivos con capacidades limitadas de almacenamiento, visualización y energía. A menudo se utiliza para desarrollar aplicaciones para dispositivos móviles, PDA, decodificadores de TV e impresoras.
- Java SE (Standard Edition): Para uso general en PC de escritorio, servidores y dispositivos similares.
- Jakarta EE (Enterprise Edition): conjunto de Java SE junto a varias API que son útiles para aplicaciones empresariales cliente-servidor de varios niveles.

Puesto que el desarrollo de este trabajo ha sido a través de Jakarta EE, que nos centraremos en dicha plataforma. Java Enterprise Edition fue un proyecto iniciado por Sun Microsystems en el año 1999 bajo el nombre de J2EE y adquirida por Oracle diez años más tarde, dando continuidad al proyecto hasta 2017, momento en el que el proyecto pasó a manos de la Eclipse Foundation, quien continúa con su desarrollo bajo la modalidad de código abierto en la actualidad [**?**].

Esta transferencia de proyecto de Oracle a Eclipse Foundation no incluyó el uso de la marca Java, por lo que Eclipse Foundation renombró el proyecto pasando a denominarse Jakarta EE. La primera versión lanzada bajo el parguas de Eclipse Foundation fue la 8 (versión con la

que se ha desarrollado la aplicación de este TFG) y actualmente está disponible la versión 9.1 (página 18).

Versión	Año	Soporte Java SE	Empresa
hline J2EE 1.2	1999	J2SE 1.2	Sun Microsystems
J2EE 1.3	2001	J2SE 1.3	Sun Microsystems
J2EE 1.4	2003	J2SE 1.4	Sun Microsystems
Java EE 5	2006	Java SE 5	Sun Microsystems
Java EE 6	2009	Java SE 6	Oracle
Java EE 7	2013	Java SE 7	Oracle
Java EE 8	2017	Java SE 8	Oracle
Jakarta EE 8	2019	Java SE 8	Eclipse Foundation
Jakarta EE 9	2020	Java SE 8	Eclipse Foundation
Jakarta EE 9.1	2021	Java SE 11//Java SE 8	Eclipse Foundation

Tabla 3.1: Evolución de Java EE

3.1.2 Arquitectura Jakarta EE

Jakarta EE se define por su especificación. La especificación define las Interfaz de programación de aplicaciones o *Application Programming Interface* (API) y sus interacciones. Jakarta EE incluye varias especificaciones que sirven para diferentes propósitos, como pueden ser generar páginas web, leer y escribir desde una base de datos de manera transaccional, administrar colas distribuidas. Las API de Jakarta EE incluyen varias tecnologías que amplían la funcionalidad de las API de Java SE base, como Jakarta Enterprise Beans, conectores, servlets, Jakarta Server Pages y varias tecnologías de servicios web [?]. En la figura 3.1 se muestra la arquitectura Jakarta EE (página 18)[?].

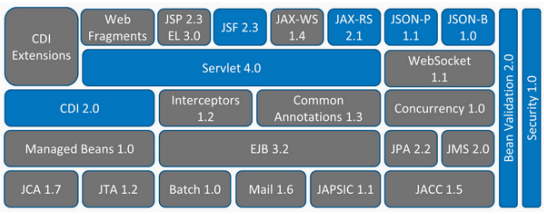


Figura 3.1: Arquitectura Jakarta EE

Entre las distintas especificaciones de Jakarta EE cabe destacar las siguientes:

- Especificaciones web
  - Jakarta Servlet: define cómo gestionar las solicitudes Protocolo de transferencia de hipertexto o *Hypertext Transfer Protocol* (HTTP), de forma síncrona o asíncrona. Es de bajo nivel y otras especificaciones de Jakarta EE dependen de él.
  - Jakarta WebSocket: define un conjunto de API para dar servicio a las conexiones de WebSocket
  - Jakarta Server Faces: una tecnología para construir interfaces de usuario a partir de componentes.
  - Jakarta Lenguaje de expresiones o *Expression Languaje* es un lenguaje simple diseñado originalmente para satisfacer las necesidades específicas de los desarrolladores de aplicaciones web.
- Especificaciones de servicios web
  - Jakarta RESTful Web Services: proporciona soporte en la creación de servicios web de acuerdo con el patrón arquitectónico de Transferencia de estado representativo o *Representation State Transfer* (REST).
  - Jakarta JSON Processing: conjunto de especificaciones para gestionar la información codificada en formato JavaScript Object Notation o *JavaScript Object Notation* (JSON).
  - Jakarta JSON Binding: proporciona especificaciones para convertir información JSON. en o desde clases Java
  - Jakarta XML Binding: permite asignar Lenguaje de marcado extensible o *Extensible Markup Language* (XML) a objetos Java; Los servicios Web XML de Jakarta se pueden utilizar para crear servicios web Simple Object Access Protocol o *Simple Object Access Protocol* (SOAP).
- Especificaciones empresariales
  - Jakarta Activation (JAF): especifica una arquitectura para extender los beans de componentes proporcionando tipos de datos y enlaces de dichos tipos.
  - Jakarta ontextos e inyección de dependencias o *Contexts and Dependency Injection* (CDI): es una especificación para proporcionar un contenedor de inyección de dependencias
  - Jakarta Enterprise Bean (EJB) define un conjunto de API ligeras que un contenedor de objetos (el contenedor EJB) admitirá para proporcionar transacciones, llamadas a procedimientos remotos, control de simultaneidad, inyección de dependencias

y control de acceso para objetos de negocio. Este paquete contiene las clases e interfaces de EJB que definen los contratos entre el bean empresarial y sus clientes y entre el bean empresarial y el contenedor ejb.

  - Jakarta Persistence API o *Jakarta Persistence API* (JPA): especificaciones sobre el mapeo objeto-relacional entre tablas de bases de datos de relaciones y clases Java.
  - API de transacción Jakarta o *Jakarta Transaction API* (JTA): contiene las interfaces y anotaciones para interactuar con el soporte de transacciones ofrecido por Jakarta EE.
  - Jakarta Messaging (JMS) proporciona una forma común para que los programas Java creen, envíen, reciban y lean los mensajes de un sistema de mensajería empresarial.

- Otras especificaciones

  - Validación: contiene las anotaciones e interfaces para el soporte de validación declarativa ofrecido por la API de validación de Bean.
  - Jakarta Batch: proporciona los medios para que el procesamiento por lotes en aplicaciones ejecute tareas en segundo plano de larga duración que posiblemente impliquen un gran volumen de datos y que puedan necesitar ser ejecutadas periódicamente.
  - Jakarta Connectors: herramienta basada en Java para conectar servidores de aplicaciones y Sistemas de información empresarial o *Enterprise Information Systems* (EIS) como parte de la integración de EAI. Esta es una API de bajo nivel dirigida a proveedores con los que el desarrollador de aplicaciones promedio generalmente no entra en contacto.

3.1.3 Componentes Jakarta EE

Jakarta EE define cuatro tipos de componentes de aplicación [**?**]:

- Aplicaciones cliente: suelen ser programas Interfaz gráfica de usuario o *Graphic User Interface* (GUI) que se ejecutan en un equipo de escritorio.
- Applets: son componentes de GUI que normalmente se ejecutan en un navegador web, pero pueden ejecutarse en una variedad de otras aplicaciones o dispositivos que admiten el modelo de programación de applets.
- Aplicaciones web como servlets, páginas Jakarta Server Pages (JSP) y JSF que se ejecutan en un contenedor web y responden a las peticiones HTTP del cliente.

- Aplicaciones empresariales como los componentes EJB, JTA o JMS, que se ejecutan en un contenedor EJB .

3.1.4 Contenedores

Jakarta EE se divide en dominios lógicos llamados contenedores. En una aplicacion Jakarta EE los componentes nunca interactúan con otros componentes de la aplicación si no que utilizan los protocolos y métodos del contenedor para interactuar entre sí y con los servicios de la plataforma [**?**]. Cada contenedor tiene una función específica, soporta un conjunto de API y ofrece servicios a los componentes tales como seguridad, acceso a base de datos, gestión de transacciones, nombres de directorios e inyección de recursos. Los contenedores ocultan la complejidad técnica y mejoran la portabilidad lo que permite al desarrollador centrarse en la lógica de aplicación en lugar de resolver problemas técnicos. Por ejemplo el contenedor EJB es responsable de administrar la ejecución de los beans que contienen la lógica de negocio.

Servidores

Detrás de un contenedor Jakarta EE se encuentra el servidor del que forma parte [**?**]. Un proveedor de productos de Jakarta EE por lo general implementa la funcionalidad del lado del servidor de Jakarta EE usando una infraestructura de transacciones existente en combinación con la tecnología Java SE de la plataforma Java. La funcionalidad del cliente Jakarta EE se basa por lo general en la tecnología Java SE .

3.2 Librerías y componentes Jakarta EE usados

A continuación ofrecemos una relación de las principales librerías y componentes Jakarta EE usados en el desarrollo de este TFG.

3.2.1 Jakarta Enterprise Bean

Un Jakarta Enterprise Bean (EJB) es un componente del lado del servidor que encapsula la lógica empresarial de una aplicación. La lógica de negocio es el código que cumple el propósito de la aplicación [**?**].

Estos componentes se ejecutan en el contenedor EJB. Aunque es transparente para el desarrollador de aplicaciones, el contenedor EJB proporciona servicios a nivel de sistema tales como transacciones y seguridad a sus enterprise beans. Estos servicios le permiten crear e implementar rápidamente EJB, que son los que forman el núcleo de las aplicaciones transaccionales de Jakarta EE.

Entre las ventajas que ofrecen los *enterprise beans* se encuentran los siguientes:

- Simplifican el desarrollo de la aplicación ya que es el contenedor EJB el que se encarga de gestionar los servicios a nivel de sistema, como la gestión de transacciones y la autorización de seguridad.
- La lógica del negocio reside en los *enterprise beans* y no en el lado del cliente, permitiendo que el desarrollo del lado del cliente este desacoplado de la lógica del negocio.
- Los *enterprise beans* son componentes portables, reutilizables y pueden ser desplegados en servidores que usen los estándares del Jakarta EE Jakarta EE.
- Pueden residir en diferentes servidores pudiendo ser invocados por un cliente remoto.

El uso *enterprise beans* está recomendado en aplicaciones escalables, en aquellas en las que necesitemos asegurar la integridad de los datos y/o en aplicaciones con variedad de clientes.

3.2.2 Java Server Faces

Jakarta Server Faces (JSF) es una tecnología para el desarrollo de GUI en aplicaciones web dentro de la plataforma Jakarta EE [**?**].

La especificación está definida por Java Community Process (JCP), lo que lo convierte en un estándar. Se basa en la utilización del patrón Modelo vista controlador o *Model View Controller - MVC* (MVC) que tiene como objetivo separar los datos (modelo) de su procesamiento (controlador) y la forma en la que estos son presentados al usuario en una aplicación (vista).JSF oculta al programador los detalles de las peticiones y respuestas HTTP, simplificando la programación de aplicaciones web y acercándolas a un estilo de desarrollo similar al de las aplicaciones de escritorio.

En JSF se define un servlet llamado Faces Servlet dentro de un archivo de configuración llamado *web.xml*, que tiene como objetivo recibir las distintas solicitudes HTTP de los cliente y procesarlas.

Ciclo de vida de una aplicación JSF

El ciclo de vida de una aplicación JSF consta de las siguientes fases [**?**]:

1. Restaurar la vista: en esta etapa JSF comprueba si la solicitud HTTP recibida es nueva o no con el fin de crear o restaurar el árbol de componentes (objetos Java necesarios).
2. Aplicar los valores de la solicitud: una vez creado el arbol cada componente actualiza sus valores con los parámetros presentes en la solicitud.

3. Procesar validaciones: Valida y convierte los datos que el usuario ingresó al tipo que corresponda. En caso de que se produzca algún error, se saltará a la etapa de renderización, en la que se le informará al usuario de los errores producidos. En caso contrario se pasará a la siguiente fase (actualizar valores).

4. Actualizar los valores del modelo: se refresca el árbol de componentes actualizando los valores del modelo.

5. Invocar la lógica de la aplicación: se invocan los métodos solicitados por el usuario.

6. Renderizar la respuesta: se envía al cliente la respuesta en HTML.

3.2.3 Primefaces

Primefaces es una librería de componentes visuales open source que cuenta con un amplio conjunto de componentes enriquecidos para diseñar interfaces de usuario creada por PrimeTek Informatics. El desarrollo inicial de PrimeFaces se inició a finales de 2008[**?**].

Las principales características que presenta son las siguientes [**?**]:

- Amplio conjunto de componentes (HtmlEditor, Dialog, Autocompletar, Gráficos y muchos más).
- Soporte nativo AJAX basado en API JSF AJAX estándar.
- Ligero, sin necesidad de dependencias ni configuraciones adicionales.
- Capacidad de respuesta y accesibilidad incorporadas.
- Amplia documentación y comunidad de usuarios.

3.2.4 Omnifaces

OmniFaces es una biblioteca de utilidades open source para JSF desarrollada por dos miembros de la JSF Expert Group (JSF EG), Bauke Scholtz (alias BalusC) y Arjan Tijms[**?**].

A diferencia de otras bibliotecas de componentes JSF existentes en el mercado (como PrimeFaces, BootsFaces o ButterFaces), OmniFaces no contiene ningún componente orientado a enbellecer la visualización, si no que está más orientado a las ”utilidades” que resuelven problemas prácticos cotidianos y soluciones para (pequeñas) deficiencias en la API JSF. Esta librería se centra en utilidades que facilitan las tareas cotidianas con la API JSF estándar. Estamos ante una respuesta a los problemas recurrentes con los que los autores se han encontrado a lo largo de su desarrollo profesional con JSF y de las preguntas que se publican en Stack Overflow.

3.2.5 Java Object Oriented Querying

Java Object Oriented Query (JOOQ) es una biblioteca de software de asignación de bases de datos ligera en Java que implementa el patrón de registro activo desarrollada por Lukas Eder. Su propósito es ser tanto relacional como orientado a objetos al proporcionar por un lado un lenguaje específico del dominio para construir consultas SQL seguras y por otro generar clases a partir de un esquema de base de datos a través de su API fluida sobre las que construir dichas consultas. [**? ?**].

El enfoque de JOOQ pone el foco en el lado de Lenguaje de consulta estructurada o *Structured Query Language* (SQL) en lugar de JPA como hacen los Mapeo objeto-relacional o *Object-Relational Mapping* (ORM) como por ejemplo Hibernate.

Alguna de las características que presenta JOOQ son las siguientes[**?**]:

- Proporciona un generador de código que permite, mediante ingeniería inversa, converir el esquema de base de datos en conjunto de tablas de modelado de clases Java, registros, secuencias, POJO, DAO, procedimientos almacenados, tipos definidos por el usuario y muchos más listas para usarse.
- Ofrece la posibilidad de utilizar toda la funcionalidad de la base de datos seleccionada en lugar de una abstracción o un subconjunto limitado gracias a su Lenguaje de dominio específico o *Domain Specific Languaje* (DSL) fluido que se asigna uno a uno con SQL
- El Lenguaje de dominio específico o *Domain Specific Languaje* (DSL) que genera permite que su IDE se complete automáticamente mientras escribe las consultas.
- Utiliza el sistema de tipos de Java para garantizar que el SQL que genera es válido.
- La API de JOOQ hace todo lo posible para asegurarse de que está utilizando el tipo de datos Java correcto para cada columna de su base de datos.
- Presenta una extensión de conversor de tipos fácil de usar para transformar tipos de datos comunes, como Timestamp o SQL Date/Time a java.time.ZonedDateTime.
- Existe una funcionalidad de mapeo que ayuda a asignar conjuntos de resultados a Plain Old Java Object (POJO) y proporciona una implementación del patrón de registro activo para trabajar de una manera similar a ORM
- Se pueden usar los streams existentes a partir de Java 8 para transformar conjuntos de resultados de forma fácil y sucinta.
- Si se producen cambios en el esquema de la base de datos se obtienen errores de compilación en lugar de los de tiempo de ejecución.

- Funciona con las principales bases de datos relacionales: MS Access, MS Access 2013, CUBRID, IBM DB2, Apache Derby, Firebird, H2, Hypersonic, Informix, Ingres, MariaDB, MySQL, Oracle, Oracle, PostgreSQL, SQLite, SQL Server, SQL Server y Sybase.
- Tiene doble licencia, siendo de uso gratuito para todas las bases de datos de código abierto.

3.2.6 Log4j

Es una biblioteca código abierto perteneciente a los Java Logging Frameworks desarrollada en Java por la Apache Software Foundation que permite generar mensajes de logging de una forma limpia, sencilla, permitiendo filtrarlos por importancia y pudiendo configurar su salida por vía consola, fichero u otras distintas.

Existen diversos niveles de prioridad. A continuación se muestran los más utlizados

- DEBUG: usado para escribir mensajes de depuración.
- INFO: mensajes puramente informativos.
- WARN: alertas. Generealmente utilizados para alertar de eventos de los que se quiere dejar constancia pero que no afectan al correcto funcionamiento de la aplicación.
- ERROR: usado para mensajes de eventos que afectan al programa pero lo dejan seguir funcionando (por ejemplo un parámetro figura como vacío por lo que se se carga el parámetro por defecto).
- FATAL: usado para errores críticos.

3.3 Formatos de almacenamiento. PostgreSQL

Los datos se han almacenado en una base de datos relacional. Para ello se ha optado por el gestor de base de datos de código abierto PostgreSQL, ya que es una solución robusta, con amplias funcionalidades, con un uso cada vez más extendido y es 100% open source [**?**].

3.3.1 Historia

El sistema de gestión de bases de datos objeto-relacionales ahora conocido como PostgreSQL se deriva del paquete POSTGRES escrito en la Universidad de California en Berkeley. Con más de dos décadas de desarrollo a sus espaldas, PostgreSQL es ahora la base de datos de código abierto más avanzada disponible en cualquier lugar.

Inicios. POSTGRES

La implementación de POSTGRES comenzó en 1986. Desde entonces ha tenido varios lanzamientos importantes. El primer sistema "demoware[1] " entró en funcionamiento en 1987 y se mostró en la Conferencia ACM-SIGMOD de 1988. La versión 1 fue lanzada a algunos usuarios externos en junio de 1989. La Versión 2 fue lanzada en junio de 1990 con un nuevo sistema de reglas. La versión 3 apareció en 1991 y agregó soporte para múltiples administradores de almacenamiento, un ejecutor de consultas mejorado y un sistema de reglas reescrito. En su mayor parte, las versiones posteriores hasta Postgres95 se centraron en la portabilidad y la confiabilidad.

POSTGRES se ha utilizado para implementar muchas aplicaciones diferentes de investigación y producción. También se ha utilizado como una herramienta educativa en varias universidades. A finales de 1992, POSTGRES se convirtió en el principal gestor de datos del proyecto de computación científica Sequoia 2000.

Postgres95

En 1994, Andrew Yu y Jolly Chen agregaron un intérprete de lenguaje SQL a POSTGRES. Bajo un nuevo nombre, Postgres95 fue posteriormente lanzado a la web para encontrar su propio camino en el mundo como un descendiente de código abierto del código original de POSTGRES Berkeley. Estaba codificado completamente en ANSI C y redujo su tamaño un 25% respecto al código original. También se mejoró el rendimiento y la capacidad de mantenimiento.

PostgreSQL

En 1996 se eligió un nuevo nombre, PostgreSQL, para reflejar la relación entre el POSTGRES original y las versiones más recientes con capacidad SQL. Al mismo tiempo se configuró la numeración de la versión para que comenzara en 6.0, poniendo los números de nuevo en la secuencia originalmente iniciada por el proyecto POSTGRES de Berkeley.

Durante el desarrollo de PostgreSQL el foco se trasladó hacia el aumento de las características y capacidades

3.3.2 Características

PostgreSQL es compatible con una gran parte del estándar SQL y entre sus caraterísticas figuran las siguientes:

[1] Tipo de software que permite su uso sin ninguna restricción por un período limitado de tiempo, por un número de usos o por progresión hasta un determinado punto. Generalmente pasado el período de prueba, se deshabilitan todas las funciones o parte de ellas.

- Es *open source*
- Emplea un lenguaje SQL estandar.lo que permite la importación de otras bases de datos.
- Cumple con el modelo ACID (atomicidad, consistencia, aislamiento y durabilidad de los datos almacenados).
- Capacidad de realizar consultas complejas.
- Claves foráneas.
- Triggers(disparadores).
- Vistas actualizables.
- Integridad transaccional.
- Control de versiones simultánea.

También permite ampliaciones por parte del usuario de múliples formas, por ejemplo mediante la agregación de:

- Nuevos tipos de datos
- Funciones
- Operadores
- Funciones agregadas
- Lenguajes procedurales
- Extensiones

Capítulo 4

Análisis y diseño

En este capítulo nos centraremos en el análisis y posterior diseño realizado en este TFG.

4.1 Análisis de las necesidades

Se pretende desarrollar una aplicación web que permita registrar las distintas contrataciones realizadas por los clientes de una empresa proveedora de servicios. Puesto que las contrataciones se realizan sobre el catálogo de servicios de la empresa, también proveerá de las herramientas necesarias para mantener dicho catálogo. Asimismo deberá permitir definir una serie de elementos parmetrizables necesarios para definir las distintas entidades que alberga dicha aplicación. Por lo tanto las entidades a tratar en el sistema se englobarán en tres grandes grupos:

- Parametrización: esta categoría engloba todos los elementos de parametrización necesarios para la definición de las distintas entidades del sistema (niveles de aplicación, tipos de descuento, tipos impositivos, etc.).
- Catálogo: esta categoría engloba todos los elementos de que definen el catálogo de servicios la empresa (tipos de productos, servicios, cuotas, etc. y sus relaciones), así como los tipos de entidades que definen el catálogo de clientes y contrataciones (tipos de clientes, tipos de ciclos de facturación, etc.).
- Instancia: esta categoría engloba todos los elementos que conforman las contrataciones: clientes, cuentas, elementos contratados, etc.

Dicha aplicación podrá ser utilizada por el personal del departamento de contratación y ventas de una empresa, que podrá llevar a cabo altas, bajas y modificaciones, así como personal de otros departamentos que puedan necesitar acceder a la información de la herramienta desarrollada con el fin de poder llevar a cabo sus tareas. Estos usuarios pertenecientes a otros

departamentes han de tener limitado el acceso a las funcionalidades definidas en la aplicación. Además existirá una figura de administrador, que tendrá acceso a funcionalidades avanzadas.

Es por ello que se definen tres perfiles de acceso a la herramienta en función de las funcionalidades a las que se le quiera dar acceso al usuario. Dichos perfile serán los siguientes:

- READ: perfil que da sólo acceso a la consulta de los datos almacenados en el sistema, esto es, los usuarios que tengan este perfil asociado solo podrán listar la información, pero no podrán realizar ninguna modificación sobre los mismos.
- WRITE: perfil que además de permitir la consulta de los datos almacenados en el sistema permite realizar altas, bajas y modificaciones sobre todos los elementos que conforman el sistema.
- ADMIN: perfil que amplía el acceso a las funcionalidades definidas para el perfil WRITE, otorgándole acceso a la gestión de usuarios: podrá dar de alta nuevos usuarios, modificar usuarios existentes (incluido el cambio de contraseña) o darlos de baja.

Con el fin de poder hacer un seguimiento de los distintos cambios producidos en el sistema se deberá llevar registro de la creación de los elementos del catálogo y de las instancias, así como de la última modificación realizada.

También con el objetivo de poder llevar un registro sobre los cambios producidos sobre determinadas entidades que pueden sufrir cambios durante su ciclo de vida se establecen registros de histórico para dichas entidades. El objetivo de estos históricos es poder tener la *foto* del estado de una determinada entidad en un momento determinado, por ejemplo, el precio que tenía la definición de una cuota determinada o los distintos servicios activos de un cliente en una fecha determinada.

Esta aplicación tendrá el objetivo de formar parte de un ecosistema mayor, integrándose en una segunda fase con el resto de herramientas de la empresa para conformar el área comercial de la misma (figura 4.1, 29)

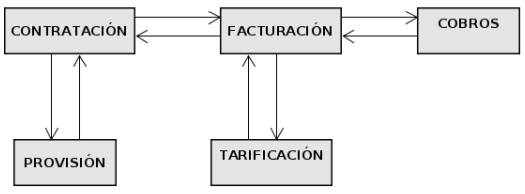


Figura 4.1: Entorno comercial

4.1.1 Catálogo de servicios

ITIL define el catálogo de servicios como [**?**]: «una base de datos centralizada que contiene información precisa sobre las ofertas de servicios de TI activas y un subconjunto de la cartera de servicios del proveedor de servicios de TI». De forma simple vendría siendo la *mercancía* que ofrece una determinada empresa. Pero va un paso más allá del listado de elementos ofertados en este momento, si no que lleva registro del ciclo de vida de todos los productos y servicios gestionados por la empresa, tanto los servicios y productos retirados como los que se ofrecen actualmente e incluso los que están en desarrollo.

La idea es la siguiente: la empresa define su catálogo de servicios(tecnología a ofertar), los elementos facturables a aplicar por la prestación de servicios y los descuentos existentes sobre dichos elementos facturables. Se distinguen dos tipos de elementos facturables: cuotas (cargos periódicos asociados a la prestación de los distintos servicios) y consumos (cargos puntuales derivados del consumo de algún elemento definido como consumible) El catálogo de servicio de nuestra empresa estará formado por cinco entidades principales y las relaciones que guardan entre sí (figura 4.2, página 31):

Tipo de producto es el elemento raiz del catálogo del sistema sobre el que se irán definiendo el resto de elementos que se pueden ofrecer al cliente. Podemos verlo como el *paquete* que contendrá todo lo que se le ofrece al cliente.

Tipo de servicio es el elemento que realmente aporta valor al cliente, ya que es el que define el elemento que va a aportar el valor de la contratación, por ejemplo la línea de telefono.

Tipo de promoción es el elemento que definen los descuentos que define la empresa. Su ámbito de aplicación se circunscribe a productos y servicios y forma parte de su definición. El descuento se aplicará sobre los distintos elementos facturables existente en el sistema.

Tipo de cuota es el elemento que define la contraprestación económica que va a recibir la empresa de forma periódica por la prestación de servicios que ofrece al cliente y su ámbito de aplicación se circunscribe a productos y servicios y forma parte de la definición del tipo de producto o servicio.

Tipo de consumos es el elemento que define el consumo puntual de un determinado elemento por parte del cliente, como puede ser una llamada de teléfono. Sólo tiene interés desde el punto de vista de la definición de la aplicación de los distintos descuentos, ya que a diferencia de las cuotas, para las que podemos definir un precio cerrado que puede conocerse de antemano, el importe de los consumos vendrá definido por la naturaleza propia del evento

del consumo (por ejemplo, el coste de una llamada de teléfono viene dada, además de por el tipo de llamada que lleva asociado un precio por minuto, por la duración de la misma o la franja horaria en la que se realiza) por lo que en la herramienta simplemente se definirán los consumos susceptibles de ser descontados. Se considera un único ámbito de aplicación: el de servicio.

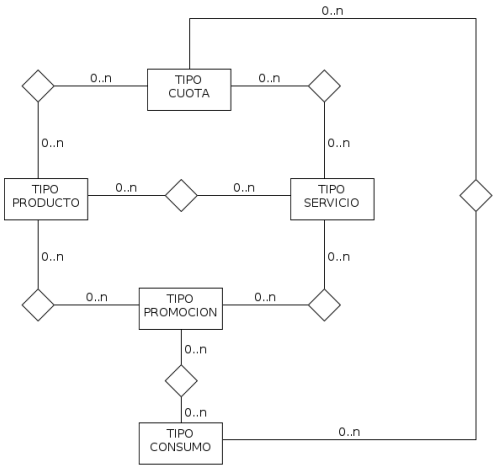


Figura 4.2: Catálogo de servicios

Puesto que el catálogo de servicios contiene la definición de los distintos elementos que podrán contratarse, hemos incluido dentro del concepto de catálogo otros elementos que permiten definir la plantilla con la que crear la posterior *instanciación* de las mismas, a saber:

Tipo de ciclo de facturación la instanciación de los distintos ciclos de facturación no forma parte del ámbito de la herramienta de contratación propiamente (entraría dentro del ámbito de facturación) pero puesto que forma parte de la definición de los distintos tipos de cuentas existentes, incluimos la definición del tipo de ciclo de facturación en la herramienta a desarrollar.

Tipo de cliente esta entidad permitirá la creación de clientes atendiendo a una clasificación definida por la empresa, por ejemplo para distinguir entre clientes particulares de clientes de empresa.

Tipo de cuenta la cuenta es la entidad sobre la que se realizarán las contrataciones, que a su vez dependerá de un cliente particular. La cuenta estará definida por una serie de valores especificados en la tipología de la misma, como el tipo de pago.

4.1.2 Contratación

La contratación contendrá la implementación (instancia) real de los distintos elementos definidos en el catálogo, esto es el conjunto de clientes y sus relaciones con los productos y servicios contratados (figura 4.3, 32):

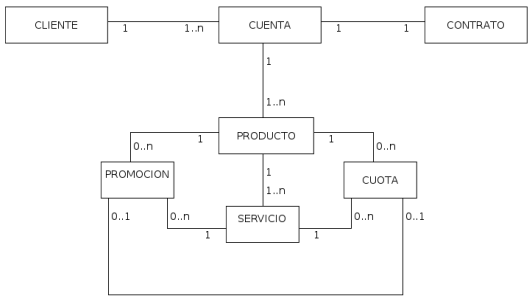


Figura 4.3: Contratación

4.1.3 Elementos de parametrización

Los elementos de parametrización son elementos que nos permiten modelar entidades superiores para que se comporten de una forma determinada. No es lo mismo aplicar un descuento en forma de porcentaje sobre el total que en número de unidades contratadas. Para el desarrollo de la herramienta se han considerado los siguientes elementos de parametrización.

Entidades define los distintos tipos de entidades definidas en el sistema.

Clases de consumo define las distintas clases de consumo definidas en el sistema (llamadas a fijo, a móviles, sms,...).

Estado para reflejar los distintos estados que tienen las entidades del sistema a lo largo de su ciclo de vida.

Nivel de aplicación para indicar sobre qué entidad aplican las distintas cuotas y promociones: si sobre un producto o sobre un servicio.

Tipos de descuento reflejan los distintos tipos de descuento que pueden tenerse en cuenta (porcentaje sobre el total, unidades de consumo,...).

Tipos de pago reflejan los distintos métodos de pago que contempla la empresa (domiciliación bancaria, transferencia,...).

Período de facturación define el número de días que conforman los períodos de facturación.

Tipo impositivo define los distintos tipos impositivos que contempla la aplicación (iva, igic,...) y sus valores porcentuales de aplicación.

Tipo de documento de identificación define los distintos tipos de documento de identificación contemplados por el sistema.

4.2 Casos de uso

Se definen tres tipos de usuarios de la aplicación en función de los funcionalidades que puedan realizar en la misma en función del perfil que tengan asignado. Los perfiles defindos para la aplicación son los siguientes:

- READ: perfil de sólo lectura. Sólamente permite visualizar la información almacenada en la aplicación, pero no puede realizar ninguna modificación, salvo la relativa a su información de contacto y su contraseña.
- WRITE: perfil de lectura y modificación. Además de las funcionalidades descritas para el perfil READ, permite realizar modificaciones sobre las distintas entidades del sistema (altas/bajas/modificaciones).
- ADMIN: perfil de administrador. Además de las funcionalidades descritas para el perfil WRITE, permite la gestión de usuarios (altas/bajas/modificaciones).

4.2.1 Actores

Figura 4.4: Actores del sistema

El actor ADMIN representa a los usuarios de la aplicación que tienen acceso completo a todas las funciones, incluyendo la gestión de usuarios.

Los actores READ y WRITE representa a cualquier usuario con un perfil READ o WRITE que haya sido dado de alta previamente por un actor ADMIN, que representa a cualquier usuario con perfil ADMIN, y disponga de un nombre de usuario y una contraseña.

4.2.2 Casos de uso

Todos los usuarios comparten los casos de uso de acceso y los relativos a la consulta de las entidades del sistema. El actor WRITE amplía esos casos de uso con funcionalidades de creación, modificación y borrado de entidades y por último el actor ADMIN amplía esos casos de uso con el caso de uso de administración de usuarios. Las siguientes figuras (figura B.2, figura B.3, figura B.4 de las páginas B.2,B.3,B.4) muestran una visión de alto nivel de los distintos casos de uso definidos en el sistema. Dichos casos de uso se analizan con más detalle en los siguientes apartados.

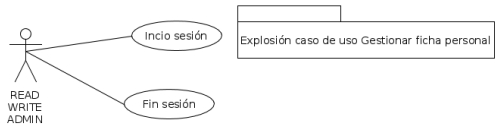


Figura 4.5: Caso de uso del acceso al sistema - Actores READ, WRITE y ADMIN

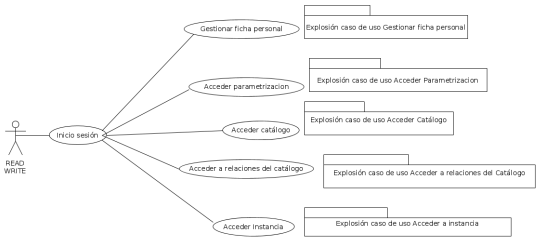


Figura 4.6: Casos de uso de los actores READ, WRITE

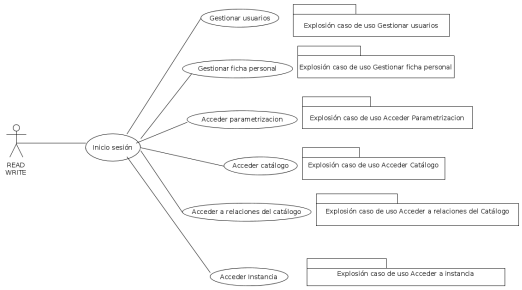


Figura 4.7: Casos de uso del actor ADMIN

A continuación se muestra un caso de uso destacable que representa aspecto centrales de la aplicación.La definición completa de casos de uso se encuentra en el apéndice B.

Se trata de los casos de uso CU-05 (tabla B.5, página 60), CU-06 (tabla B.6, página 61), CU-07 (tabla B.7, página 62) y CU-09 (tabla B.9 página 64, que hacen referencia a la modificación de un registro de histórico de una entidad con histórico de registros (como pudiera ser un tipo de cuota). A estos casos de uso pueden acceder los actores WRITE y ADMIN a través del correspondiente caso de uso de listado de la entidad a tratar y seleccionando el correspondiente registro de histórico a modificar.

Para este tipo de entidades con registro de histórico es fundamental que exista una consecución temporal de los mismos. No puede haber registros de histórico que se solapen para una misma instacia de una entidad, ni tampoco *huecos* en la línea temporal del ciclo de vida de la misma. Por lo tanto se debe revisar que los cambios producidos en las fechas de una entidad con registro de histórico sean coherentes con lo almacenado en el sistema, actualizando fechas de registros anteriores y/o posteriores en caso de ser necesario, para *acortarlos* o *alargarlos* según sea el caso.

CU-05	**Editar registro de histórico del elemento seleccionado**
Descripción	Editar registro de histórico del elemento seleccionado.
Actores	WRITE y ADMIN.
Precondiciones	El usuario con perfil WRITE o ADMIN ha pulsado el botón editar para el registro de histórico para el elemento seleccionado.
Postcondiciones	Se modifica el registro de histórico elemento del listado de la entidad y, si es necesario, se modifican las fechas de los regisros registros anterior y posterior para que sean consecutivos.
Flujo básico	1. El sistema muestra los campos editables del registro seleccionado y dos botones: guardar y cancelar. 2. El usuario modifica los campos deseados sobre el registro. Si el estado del registro cambia al estado cancelado se dispara el caso de uso B.6 (página 61). (a) El usuario pulsa el botón de guardar. El sistema comprueba que los campos obligatorios han sido completados y que los datos tienen el formato correcto. Realiza las modificaciones pertinentes en los registros de histórico anterior y posterior del elemento seleccionado, de forma que todos los registros de histórico sean correlativos. Se guardan los datos de fecha de creación y nombre de usuario tanto para el nuevo registro de histórico como para los registros adyacentes modificados. Se termina la edición del registro. Se informa al usuario que el registro ha sido modificado y se actualiza el listado de la entidad con las modificaciones realizadas. i. ***Flujo alternativo:*** *Si no se cubre algún campo obligatorio o se produce algún error de validación el sistema informa del error.* (b) El usuario pulsa el botón cancelar. Se termina la edición del registro y se descartan los posibles cambios. 3. Finaliza el caso de uso.

Tabla 4.1: CU-05 Editar histórico de elemento seleccionado

CU-06	**Cancelar el registro de histórico del elemento seleccionado**
Descripción	Cancelar el registro de histórico del elemento seleccionado.
Actores	WRITE y ADMIN.
Precondiciones	El usuario con perfil WRITE o ADMIN ha seleccionado el estado cancelado para el registro de histórico que está modificando.
Postcondiciones	Se propaga el estado de cancelado para todos los registros de histórico posteriores al registro seleccionado.
Flujo básico	1. Se muestra un cuadro de diálogo solicitando propagar el estado cancelado para los registros posteriores al registro de histórico seleccionado. 2. (a) El usuario pulsa el botón de aceptar. El sistema propaga el nuevo estado al resto de registros de históricos posteriores al registro de histórico seleccionado. Se cierra el cuadro de diálogo. (b) El usuario pulsa el botón cancelar. Se cierra el cuadro de diálogo y se descarta el cambio de estado. 3. Finaliza el caso de uso.

Tabla 4.2: CU-06 Cancelar el registro de histórico del elemento seleccionado

CU-07	**Añadir registro de histórico del elemento seleccionado**
Descripción	Añadir registro de histórico del elemento seleccionado.
Actores	WRITE y ADMIN.
Precondiciones	El usuario con perfil WRITE o ADMIN ha pulsado el botón añadir registro de histórico para el elemento seleccionado.
Postcondiciones	Se añade un nuevo registro de histórico comprendido entre el elemento seleccionado del listado de la entidad y el siguiente elemento de la lista, o a continuación del elemento seleccionado si no hay más elementos en la lista, modificando las fechas de dichos regisros para que sean consecutivos.
Flujo básico	1. El sistema muestra un formulario con una copia del registro de histórico del elemento seleccionado con los campos susceptibles de modificar habilitados. El usuario modificará las fechas de inicio y/o fin del registro, así como el resto de campos que considere oportuno. (a) El usuario pulsa el botón de guardar. El sistema comprueba que los campos obligatorios han sido completados y que los datos tienen el formato correcto. Evalúa las fechas de inicio y fin y realiza las modificaciones pertinentes sobre los histórico anterior y posterior del elemento seleccionado o sólo del anterior en caso de que no hubiera más registros, de forma que todos los registros de histórico sean correlativos. Se guardan los datos de fecha de creación y nombre de usuario para el nuevo registro de histórico, así como los de fecha de modificación y usuario para los registros de histórico adyacentes modificados. Se termina la edición del registro. Se informa al usuario que el elemento ha sido añadido. Se añade el registro al listado de la entidad y se actualiza con las modificaciones realizadas. i. ***Flujo alternativo:*** *Si no se cubre algún campo obligatorio o se produce algún error de validación el sistema informa del error.* (b) El usuario pulsa el botón cancelar. Se vuelve al caso de uso inicial B.22 (74). 2. Finaliza el caso de uso.

Tabla 4.3: CU-07 Añadir histórico de elemento seleccionado

CU-09	**Borrar histórico del elemento seleccionado**
Descripción	Borrar histórico elemento seleccionado.
Actores	WRITE y ADMIN.
Precondiciones	El usuario con perfil WRITE o ADMIN ha pulsado el botón de borrar para el registro de histórico del elemento selecconado.
Postcondiciones	Se elimina el registro de histórico del elemento seleccionado, modificando las fechas de los regisros anterior y/o posterior para que sean consecutivos.
Flujo básico	1. El sistema muestra un cuadro de diálogo solicitando confirmación de borrado del registro de histórico del elemento seleccionado con dos botones: aceptar y cancelar. 2. (a) El usuario pulsa el botón de aceptar. El sistema elimina del el registro de histórico del elemento seleccionado y realiza las modificaciones pertinentes en los registros de histórico anterior y posterior del elemento seleccionado, de forma que todos los registros de histórico sean correlativos. Se informa al usuario que el elemento ha sido borrado. Se elimina el elemento del listado de la entidad y se actualiza con las modificaciones realizadas. Se cierra el cuadro de diálogo. i. ***Flujo alternativo:*** *Si se produce algún error el sistema informa del error.* (b) El usuario pulsa el botón cancelar. Se vuelve al caso de uso inicial B.22 (74). 3. Finaliza el caso de uso.

Tabla 4.4: CU-09 Borrar histórico del elemento seleccionado

4.3 Diseño

En esta sección describiremos los distintos elementos usados para el diseño de la aplicación.

4.3.1 Arquitectura en tres niveles

Para el desarrollo de este proyecto se ha optado por el patrón de arquitectura Modelo vista controlador o *Model View Controller - MVC* que presenta una arquitectura que separa

las aplicaciones en tres niveles [1] de informática lógica y física: la interfaz de usuario, el nivel donde se procesan los datos y el nivel donde se almacenan y gestionan esos datos [?] (figura 4.8, página 40):

Nivel de presentación es la interfaz de usuario, donde el usuario final interactúa con la aplicación. Este primer nivel puede ejecutarse en una navegador web como una aplicación de escritorio o una GUI.

Nivel de aplicación también conocido como el nivel lógico o medio, es el núcleo de la aplicación. Se procesa la información recogida en el nivel de presentación. En ocasiones este procesado de información se realiza junto con otra información en el nivel de datos a través de lo que se conoce como lógica empresarial: un conjunto específico de reglas empresariales. Este nivel también puede añadir, suprimir o modificar datos en el nivel de datos. Se comunica con el nivel de datos mediante llamadas a las distintas API.

Nivel de datos también denominado nivel de base de datos, de acceso a datos o backend, es donde se almacena y gestiona la información procesada por la aplicación. Puede ser un sistema de gestión de base de datos relacional (PostgreSQL, MySQL, Oracle,...) o en un servidor de bases de datos NoSQL (MongoDB, Cassandra,...).

En una aplicación de tres niveles, toda la comunicación pasa por el nivel de aplicación. Los niveles de presentación y de datos no pueden comunicarse directamente entre sí.

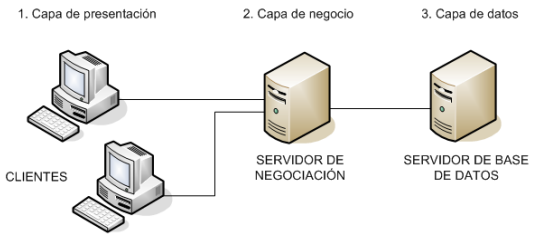


Figura 4.8: Arquitectura en tres niveles

La arquitectura de tres niveles aporta múltiples ventajas [?]:

- Desarrollo más rápido: debido a que cada nivel puede ser desarrollado simultáneamente por diferentes equipos, una empresa puede llevar su aplicación al mercado más rápido y

[1] Aunque está comunmente aceptado capa como sinónimo de nivel, no significan lo mismo. Una ***capa*** se refiere a una *división funcional del software*, pero un ***nivel*** se refiere a una *división funcional del software que se ejecuta en una infraestructura separada* de las otras divisiones, por lo que las capas no pueden ofrecer los mismos beneficios que los niveles

los programadores pueden utilizar los mejores y más recientes lenguajes y herramientas para cada nivel.

- Escalabilidad mejorada: cualquier nivel se puede escalar independientemente de los demás según sea necesario.
- Confiabilidad mejorada: es menos probable que una interrupción en un nivel afecte la disponibilidad o el rendimiento de los otros niveles.
- Seguridad mejorada: debido a que los niveles de presentación y de datos no se pueden comunicar directamente entre sí, un nivel de aplicación bien diseñado puede funcionar como una especie de firewall interno, lo que impide ataques de inyecciones SQL y otras vulnerabilidades maliciosas.

4.3.2 Patrones utilizados

A continuación se listan algunos de los patrones utilizados en el diseño de la herramienta presentada.

Patrón modelo vista controlador El framework JSF se basa en el patrón Modelo vista controlador o *Model View Controller - MVC*.

Aunque el patrón MVC comparte con la arquitectura de tres niveles la visión de la apliación en 3 áreas separadas, difiere en la forma de entender la comunicación entre ellas: en el patrón de modelo vista controlador las distintas capas se comunican de forma triangular, dos a dos, mientras que en la arquitectura de tres niveles toda comunicación pasa por el nivel de aplicación.

- Capa de persistencia: Recibe solicitudes de almacenamiento o recuperación de información por parte de la capa de negocio, es decir, es la encargada de cargar y modificar la información guardada en el servidor de base de datos. Se encarga de convertir los datos de los registros de la base de datos a objetos manejables por Jakarta EE, para lo que cuenta con un mecanismo de conversión de datos relacionales a objetos.
- Capa de negocio: Se encarga de las reglas del negocio que resuelve o satisface las necesidades de un dominio en un negocio particular. En este nivel se encuentra en el servidor de aplicaciones. Los componentes de este nivel interactúan con el nivel de persistencia y suelen implementarse como componentes EJB.
- Nivel de presentación: Se encarga de interactuar con el usuario final. Contiene la lógica de presentación que se emplea para generar una respuesta al cliente y sus operaciones son soportadas por el contenedor web que es el encargado de dar un acceso visual al

sistema a través de una interfaz de usuario como la web a través de un navegador. Esta capa utiliza la capa de negocio para realizar ciertas operaciones.

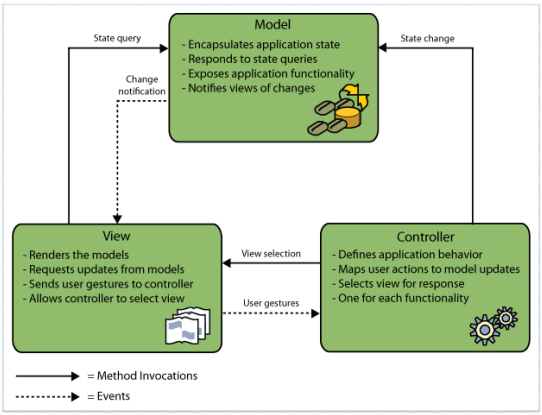


Figura 4.9: Patron MVC

Composite View El sistema de plantillas de JSF se basa en el patrón de diseño Composite View, lo que significa que las secciones de una página o incluso las propias páginas pueden ser reutilizadas por otras páginas. Se han utilizado plantillas en el diseño de la ventana principal de la aplicación, la barra de herramientas de la aplicación que contiene el acceso al menú y a la información del usuario, así como la información de la vista actual.

DTO - Data Transfer Object Se utiliza este patrón para reducir el número de llamadas a métodos mediante el uso de objetos que transportan datos entre procesos. También permite la encapsulación de la lógica de la serialización (el mecanismo que traduce la estructura del objeto y los datos a un formato específico que se puede almacenar y transferir) proporcionando un único punto de cambio en los matices de serialización.

Los objetos de este patrón generalmente se crean como Plain Old Java Object. Se trata de estructuras de datos planas que no contienen lógica de negocio, sólo almacenamiento, accesorios y, finalmente, métodos relacionados con la serialización o el análisis.

DAO - Data Access Object Se usa este patrón para abstraer y encapsular todos los accesos al almacén de datos persistente, gestionando la conexión con el origen de datos para obtener y almacenar los datos. Este patrón ha sido utilizado en la herramienta para insertar, modificar y borrar los distintos datos de la base de datos.

Capítulo 5

Implementación

En este capítulo se detallan los detalles técnicos necesarios para llevar a cabo la implementación de la aplicación.

5.1 Hardware Utilizado

El proyecto fue desarrollado sobre un equipo con las siguientes características:

- Procesador: Intel® Core™ i7-1195G7, 2.90GHz × 8
- Memoria: 16,0 GiB
- Disco Duro: 1,0 TB

5.2 Software Utilizado

UBUNTU 22.04.1 LTS y 22.04.1 LTS
Sistema operativo sobre el que se desarrolló la aplicación.

WILDFLY 20.0.1
Servidor de aplicaciones Jakarta EE de código abierto, utilizado para la implantación de la aplicación web.

MOZILLA FIREFOX 104.0
Navegador web usado para la ejecución y pruebas de la aplicación web.

ECLIPSE 4.16 2020-06
Entorno de desarrollo integrado o *Integrated Development Environment* (IDE) código abierto

utilizado para la generación del código de la aplicación .

JAKARTA EE 8.0 (y JAVA SE 1.8.0)
Lenguaje de programación utilizado para desarrollar la aplicación.

MAVEN 3.6.3
Herramienta de software de código abierto para la gestión y construcción de proyectos Java.

POSTGRESQL 14.5
SGBD de código abierto utilizado como soporte de almacenamiento.

DBeaver 22.1.4
Herramienta gráfica código abierto de diseño y gestión de bases de datos, utilizada para gestionar la base de datos creada para la aplicación.

SCHEMASPY 5.0.0
Herramienta basada en Java que genera representaciones visuales de un esquema de base de datos en un formato visible por el navegador.

UMLET 15.0
Herramienta código abierto para la modelización de diagramas UML.

UMBRELLO 2.35.0
Programa para desarrllollar diagramas en Lenguale de modelado unificado o *Unified Modeling Language* (UML) basado en tecnología KDE.

LaTeX
Sistema de composición de textos utilizado para generar el presente documento.

TEXMAKER 5.0.3
Editor LaTeX para redactar el presente documento.

Capítulo 6

Pruebas de software

El enfoque que se ha seguido para la validación del software desarrollado es la utilización de los casos de uso del actor READ y los específicos de los actores WRITE y ADMIN para cada una de las entidades existentes en el sistema. A partir de los mismos se han probado las distintas acciones que se pueden realizar en el sistema y su correcto funcionamiento.

Aunque muchas pruebas se han ido realizando de forma paralela a la codificación, se distinguen dos etapas donde se han intensificado, coincidiendo una con la obtención de un primer prototipo, que constituye el grueso de la aplicación, y otra con el final de la codificación.

Para probar el funcionamiento global se han introducido un conjunto de datos con las casuísticas suficientes como para permitir verificar todas las funcionalidades y tener una visión global del comportamiento de la aplicación una vez puesta en producción.

Capítulo 7

Planificación temporal y estimación de costes

Tan importante como la estimación económica del proyecto es la planificación temporal del mismo. La duración de cada actividad del proyecto vendrá dada por múltiples factores entre ellos la complejidad, el esfuerzo requerido y el tiempo disponible.

Del equilibrio entre ambos factores depende en buena parte la viabilidad del proyecto. Por ello, es necesario realizar un estudio del tiempo que se va a dedicar a cada fase y un análisis económico. Una buena gestión maximiza la probabilidad de consecución de resultados a tiempo, dentro de presupuesto y con la calidad esperada.

A continuación se presenta la planificación temporal y la estimación de costes de nuestro proyecto.

7.1 Fases del proyecto

La planificación se ha realizado desglosando el proyecto en seis fases o tareas: análisis, diseño, codificación, pruebas, documentación y memoria.

La duración de cada actividad del proyecto viene dada por múltiples factores, entre ellos la curva de aprendizaje de las tecnologías utilizadas para cada actividad, la complejidad de los problemas a resolver y los contratiempos que han surgido en cada etapa.

Para la realización del proyecto, por circunstancias personales se ha establecido un horario de 28 horas semanales, salvo para el último mes que han sido de 40 horas semanales. En aquellos casos en que las actividades se solapan, para facilitar los cálculos, se ha supuesto que el tiempo dedicado a cada una de ellas ha sido equivalente. Cada cuadro sombreado de la gráfica corresponde aproximadamente a una semana de trabajo (figura 7.1, página 47)

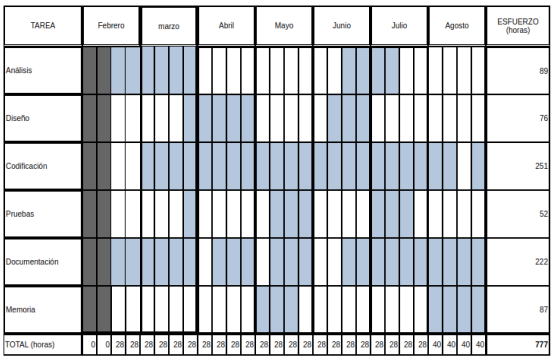


Figura 7.1: Tabla de tiempos de planificación del proyecto

7.2 Costes estimados

Este trabajo ha requerido aproximadamente 777 horas de trabajo personal que podríamos valorar en 13.986 € a razón de 18 €/hora. A este coste habría que añadir el coste del esfuerzo dedicado por el director del proyecto, software, equipo, y otros gastos generales como electricidad, conexión a Internet, etc.

Si el proyecto fuese llevado a cabo por un equipo multidisciplinar, el coste laboral total dependería de la categoría de cada integrante atendiendo a su cualificación profesional. Podría estar formado por:

- Consultor, que se encarga de las líneas de investigación del departamento.
- Ingeniero Senior, con funciones de dirección de proyecto.
- Ingeniero Junior, con una experiencia menor de cinco años.
- Técnicos Informáticos.
- Personal Auxiliar.

A los costes laborales habría que añadir los derivados de:

- Material y equipo (hardware, licencias, consumibles, conexión a Internet, etc.).
- Costes generales (luz, teléfono, impuestos).
- Otros gastos no reflejados en los apartados anteriores.

Capítulo 8

Conclusións

En este TFG se ha desarrollado una herramienta para la gestión del catálogo de servicios y contratación de una cartera de clientes para una empresa proveedora de servicios, alcanzándose los principales objetivos que se pretendían al comienzo de su realización:

- Se ha implementado una aplicación con acceso vía web que abarca los principales componentes que conforman el sistema: catálogo de servicios, contrataciones y la parametrización necesaria para la configuración de estos elementos.
- La herramienta permite una gestión de usuarios para implementar el control de acceso o modificación de la información.
- Se definen distintos niveles de acceso a las funcionalidades de la aplicación en función de distintos perfiles de usuario, lo que evita accesos inadecuados a determinadas funcionalidades del sistema.
- A la hora de manejar entidade con histórico se realiza una correcta gestión de las fechas de inicio y fin de los mismos de forma que queda garantizada la no existencia de *huecos* entre registros históricos para una misma entidad o instancia (es decir, todos los registros comprendidos entre la fecha mínima de inicio y la máxima de fin son consecutivos).
- Permite gestionar las diferentes relaciones de dependencia de entidades del catálogo de servicios con un sólo click.
- Permite búsquedas sencillas para establecer las relaciones de dependencia de las distintas contrataciones a realizar.

A esto hay que sumar el cumplimiento de los objetivos personales de ampliar mis conocimientos en distintas áreas técnicas como la de iniciar una aplicación web desde cero con Jakarta EE o gestionar una base de datos completa.

8.1 Líneas futuras

La herramienta aquí presentada es un prototipo cuyo principal objetivo es la puesta en escena de las principales funcionalidades descritas para la herramienta de contratación. Con vistas a la difusión de una herramienta como la propuesta sería interesante mejorar, o desarrollar en algunos casos, los siguientes aspectos:

- Mejorar el entorno gráfico para que resulte más agradable y atractivo.
- Mejorar la experiencia de los usuarios añadiendo opciones de personalización de la interfaz.
- Incluir la funcionalidad de extracción de informes útiles para departamentos como márketing como pudiera ser volumetrías de clientes dados de alta en un período determinado o productos más contratados.
- Incluir un canal de comunicación (por ejemplo procesos batch que generan o cargan ficheros de datos) con los distintos sistemas con los que se comunica dentro del área comercial de la empresa, como pudieran ser los sistemas de provisión y facturación, de forma que exista un flujo de información que garantice una correcta alineación en los sistemas del estado de los datos implicados (contrataciones pendientes de provisión, finalización de provisión, suspensión por impago,...).
- Incluir la posibilidad de presentar la aplicación en distintos idiomas mediante la internalización de java.

Apéndices

Apéndice A

Material adicional

Exemplo de capítulo con formato de apéndice, onde se pode incluír material adicional que non teña cabida no corpo principal do documento, suxeito á limitación de 80 páxinas establecida no regulamento de TFGs.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque.

Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Apéndice B

Referencia técnica

En este apéndice se reflejan las distintas referencias técnicas del TFG:

- Casos de uso
- Diagrama de clases
- Diagrama entidad-relación
- Estructura del código

B.1 Casos de uso

Se definen tres tipos de usuarios de la aplicación en función de los funcionalidades que puedan realizar en la misma en función del perfil que tengan asignado. Los perfiles defindos para la aplicación son los siguientes:

- READ: perfil de sólo lectura. Sólamente permite visualizar la información almacenada en la aplicación, pero no puede realizar ninguna modificación, salvo la relativa a su información de contacto y su contraseña.
- WRITE: perfil de lectura y modificación. Además de las funcionalidades descritas para el perfil READ, permite realizar modificaciones sobre las distintas entidades del sistema (altas/bajas/modificaciones).
- ADMIN: perfil de administrador. Además de las funcionalidades descritas para el perfil WRITE, permite la gestión de usuarios (altas/bajas/modificaciones).

B.1.1 Actores

El actor ADMIN representa a los usuarios de la aplicación que tienen acceso completo a todas las funciones, incluyendo la gestión de usuarios.

Los actores READ y WRITE representa a cualquier usuario con un perfil READ o WRITE que haya sido dado de alta previamente por un actor ADMIN, que representa a cualquier usuario con perfil ADMIN, y disponga de un nombre de usuario y una contraseña.

B.1.2 Casos de uso

Todos los usuarios comparten los casos de uso de acceso y los relativos a la consulta de las entidades del sistema. El actor WRITE amplía esos casos de uso con funcionalidades de creación, modificación y borrado de entidades y por último el actor ADMIN amplía esos casos de uso con el caso de uso de administración de usuarios. Las siguientes figuras (B.2, B.3, B.4 de las páginas B.2,B.3,B.4) muestran una visión de alto nivel de los distintos casos de uso definidos en el sistema. Dichos casos de uso se analizan con más detalle en los siguientes apartados.

Caso de uso de acceso al sistema

Para poder accede al sistema es necesario que el usuario esté dado de alta y disponga de un usuario y contraseña válidos. Desde el punto de vista del acceso existen dos casos de uso: inicio de sesión (B.1, página 56) y término de sesión(B.2, página 57).

Figura B.1: Actores del sistema

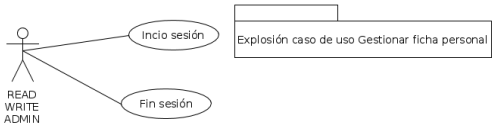


Figura B.2: Caso de uso del acceso al sistema - Actores READ, WRITE y ADMIN

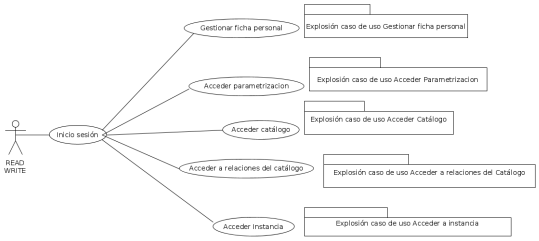


Figura B.3: Casos de uso de los actores READ, WRITE

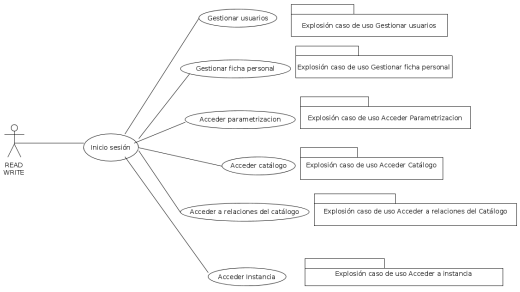


Figura B.4: Casos de uso del actor ADMIN

CU-01	**Inicio de sesión**
Descripción	Permite el acceso del usuario a la aplicación.
Actores	READ, WRITE y ADMIN.
Precondiciones	El usuario no tiene sesión iniciada.
Postcondiciones	Se crea una nueva sesión en el sistema para el usuario.
Flujo básico	1. El sistema muestra un formulario con campos de nombre de usuario y contraseña y un botón de envío de los datos del formulario. 2. El usuario cubre el formulario con los datos correspondientes y envía la información del mismo a través del botón de envío. 3. El sistema valida los datos de conexión del usuario, le crea una nueva sesión y lo redirige a la página principal. (a) ***Flujo alternativo:*** *Si el usuario no existe, o la contraseña no es correcta, el sistema informa del error y se reinicia el caso de uso.* 4. Finaliza el caso de uso.

Tabla B.1: CU-01 Inicio de sesión

CU-02	**Término de sesión**
Descripción	Permite al usuario terminar la sesión previamente iniciada y desconectarse del sistema.
Actores	READ, WRITE y ADMIN.
Precondiciones	El usuario se encuentra conectado actualmente.
Postcondiciones	Se finaliza la sesión de usuario y se eliminan los recursos asociados.
Flujo básico	1. El usuario da orden de finalizar la sesión mediante un botón de fin de sesión. 2. El sistema finaliza la sesión del usuario y descarta todos los datos asociados. 3. El sistema redirige al usuario hacia la pantalla inicial de acceso a la aplicación. 4. Finaliza el caso de uso.

Tabla B.2: CU-02 Término de sesión

Casos de uso genéricos (búsqueda, creación, edición, borrado, etc.

En este apartado se muestran los casos de uso genéricos para la búsqueda, selección, creación, edición y borrado de los distintos elementos del sistema.

CU-03	**Crear nuevo elemento**
Descripción	Crear nuevo elemento.
Actores	WRITE y ADMIN.
Precondiciones	El usuario con perfil WRITE o ADMIN ha pulsado el botón de crear nueva entidad.
Postcondiciones	Se añade un nuevo registro en el listado de la entidad.
Flujo básico	1. El sistema muestra un cuadro de dialogo con los campos que describen el elemento. 2. El usuario rellena los campos deseados. Pulsa el botón de guardar. 3. El sistema comprueba que los campos obligatorios han sido completados y que los datos tienen el formato correcto. Se guardan los datos de fecha de creación y nombre de usuario, se añade al listado de la entidad, se informa al usuario que el ítem ha sido añadido y se cierra el cuadro de diálogo. (a) ***Flujo alternativo:*** *Si no se cubre algún campo obligatorio o se produce algún error de validación el sistema informa del error.* 4. Finaliza el caso de uso.

Tabla B.3: CU-03 Crear nueva entidad de parametrización

CU-04	**Editar elemento seleccionado**
Descripción	Editar elemento seleccionado.
Actores	WRITE y ADMIN.
Precondiciones	El usuario con perfil WRITE o ADMIN ha pulsado el botón editar para un elemento de la entidad.
Postcondiciones	Se modifica el elemento del listado de la entidad.
Flujo básico	1. El sistema muestra los campos editables del registro seleccionado y dos botones: guardar y cancelar. 2. El usuario modifica los campos deseados sobre el registro. (a) El usuario pulsa el botón de guardar. El sistema comprueba que los campos obligatorios han sido completados y que los datos tienen el formato correcto. Se guardan los datos de fecha de creación y nombre de usuario. Se termina la edición del registro. Se informa al usuario que el ítem ha sido modificado. Se actualiza el listado de la entidad con las modificaciones realizadas. i. ***Flujo alternativo:*** *Si no se cubre algún campo obligatorio o se produce algún error de validación el sistema informa del error.* (b) El usuario pulsa el botón cancelar. Se termina la edición del registro y se descartan los posibles cambios. 3. Finaliza el caso de uso.

Tabla B.4: CU-04 Editar elemento seleccionado

CU-05	**Editar registro de histórico del elemento seleccionado**
Descripción	Editar registro de histórico del elemento seleccionado.
Actores	WRITE y ADMIN.
Precondiciones	El usuario con perfil WRITE o ADMIN ha pulsado el botón editar para el registro de histórico para el elemento seleccionado.
Postcondiciones	Se modifica el registro de histórico elemento del listado de la entidad y, si es necesario, se modifican las fechas de los regisros registros anterior y posterior para que sean consecutivos.
Flujo básico	1. El sistema muestra los campos editables del registro seleccionado y dos botones: guardar y cancelar. 2. El usuario modifica los campos deseados sobre el registro. Si el estado del registro cambia al estado cancelado se dispara el caso de uso B.6 (página 61). (a) El usuario pulsa el botón de guardar. El sistema comprueba que los campos obligatorios han sido completados y que los datos tienen el formato correcto. Realiza las modificaciones pertinentes en los registros de histórico anterior y posterior del elemento seleccionado, de forma que todos los registros de histórico sean correlativos. Se guardan los datos de fecha de creación y nombre de usuario tanto para el nuevo registro de histórico como para los registros adyacentes modificados. Se termina la edición del registro. Se informa al usuario que el registro ha sido modificado y se actualiza el listado de la entidad con las modificaciones realizadas. i. ***Flujo alternativo:*** *Si no se cubre algún campo obligatorio o se produce algún error de validación el sistema informa del error.* (b) El usuario pulsa el botón cancelar. Se termina la edición del registro y se descartan los posibles cambios. 3. Finaliza el caso de uso.

Tabla B.5: CU-05 Editar histórico de elemento seleccionado

CU-06	**Cancelar el registro de histórico del elemento seleccionado**
Descripción	Cancelar el registro de histórico del elemento seleccionado.
Actores	WRITE y ADMIN.
Precondiciones	El usuario con perfil WRITE o ADMIN ha seleccionado el estado cancelado para el registro de histórico que está modificando.
Postcondiciones	Se propaga el estado de cancelado para todos los registros de histórico posteriores al registro seleccionado.
Flujo básico	1. Se muestra un cuadro de diálogo solicitando propagar el estado cancelado para los registros posteriores al registro de histórico seleccionado. 2. (a) El usuario pulsa el botón de aceptar. El sistema propaga el nuevo estado al resto de registros de históricos posteriores al registro de histórico seleccionado. Se cierra el cuadro de diálogo. (b) El usuario pulsa el botón cancelar. Se cierra el cuadro de diálogo y se descarta el cambio de estado. 3. Finaliza el caso de uso.

Tabla B.6: CU-06 Cancelar el registro de histórico del elemento seleccionado

CU-07	**Añadir registro de histórico del elemento seleccionado**
Descripción	Añadir registro de histórico del elemento seleccionado.
Actores	WRITE y ADMIN.
Precondiciones	El usuario con perfil WRITE o ADMIN ha pulsado el botón añadir registro de histórico para el elemento seleccionado.
Postcondiciones	Se añade un nuevo registro de histórico comprendido entre el elemento seleccionado del listado de la entidad y el siguiente elemento de la lista, o a continuación del elemento seleccionado si no hay más elementos en la lista, modificando fechas de dichos regisros para que sean consecutivos.
Flujo básico	1. El sistema muestra un formulario con una copia del registro de histórico del elemento seleccionado con los campos susceptibles de modificar habilitados. El usuario modificará las fechas de inicio y/o fin del registro, así como el resto de campos que considere oportuno. (a) El usuario pulsa el botón de guardar. El sistema comprueba que los campos obligatorios han sido completados y que los datos tienen el formato correcto. Evalúa las fechas de inicio y fin y realiza las modificaciones pertinentes sobre los histórico anterior y posterior del elemento seleccionado o sólo del anterior en caso de que no hubiera más registros, de forma que todos los registros de histórico sean correlativos. Se guardan los datos de fecha de creación y nombre de usuario para el nuevo registro de histórico, así como los de fecha de modificación y usuario para los registros de histórico adyacentes modificados. Se termina la edición del registro. Se informa al usuario que el elemento ha sido añadido. Se añade el registro al listado de la entidad y se actualiza con las modificaciones realizadas. i. ***Flujo alternativo:*** *Si no se cubre algún campo obligatorio o se produce algún error de validación el sistema informa del error.* (b) El usuario pulsa el botón cancelar. Se vuelve al caso de uso inicial B.22 (74). 2. Finaliza el caso de uso.

Tabla B.7: CU-07 Añadir histórico de elemento seleccionado

CU-08	**Borrar elemento seleccionado**
Descripción	Borrar elemento seleccionado.
Actores	WRITE y ADMIN.
Precondiciones	El usuario con perfil WRITE o ADMIN ha pulsado el botón de borrar para elemento de la entidad.
Postcondiciones	Se elimina el elemento del listado de la entidad.
Flujo básico	1. El sistema muestra un cuadro de diálogo solicitando confirmación de borrado de la entidad con dos botones: aceptar y cancelar. 2. (a) Si se pulsa aceptar se elimina del sistema el elemento seleccionado. Se informa al usuario que el elemento ha sido borrado. Se elimina el elemento del listado de la entidad. Se cierra el cuadro de diálogo. (b) Si se pulsa cancelar se cierra el cuadro de diálogo, 3. Finaliza el caso de uso.

Tabla B.8: CU-08 Borrar elemento seleccionado

CU-09	**Borrar histórico del elemento seleccionado**
Descripción	Borrar histórico elemento seleccionado.
Actores	WRITE y ADMIN.
Precondiciones	El usuario con perfil WRITE o ADMIN ha pulsado el botón de borrar para el registro de histórico del elemento selecconado.
Postcondiciones	Se elimina el registro de histórico del elemento seleccionado, modificando las fechas de los regisros anterior y/o posterior para que sean consecutivos.
Flujo básico	1. El sistema muestra un cuadro de diálogo solicitando confirmación de borrado del registro de histórico del elemento seleccionado con dos botones: aceptar y cancelar. 2. (a) El usuario pulsa el botón de aceptar. El sistema elimina del el registro de histórico del elemento seleccionado y realiza las modificaciones pertinentes en los registros de histórico anterior y posterior del elemento seleccionado, de forma que todos los registros de histórico sean correlativos. Se informa al usuario que el elemento ha sido borrado. Se elimina el elemento del listado de la entidad y se actualiza con las modificaciones realizadas. Se cierra el cuadro de diálogo. i. ***Flujo alternativo:*** *Si se produce algún error el sistema informa del error.* (b) El usuario pulsa el botón cancelar. Se vuelve al caso de uso inicial B.22 (74). 3. Finaliza el caso de uso.

Tabla B.9: CU-09 Borrar histórico del elemento seleccionado

CU-10	**Buscar elementos con histórico**
Descripción	Obtiene un listado con los elementos de histórico del tipo seleccionado.
Actores	READ, WRITE y ADMIN.
Precondiciones	El usuario ha pulsado el botón de buscar elementos con históricos.
Postcondiciones	Se obtiene un listado con los elementos de histórico que cumplen con los criterios de búsqueda.
Flujo básico	1. El sistema muestra un panel emergente conteniendo el listado de datos de la entidad seleccionada atendiendo a los criterios de búsqueda indicados: (a) Si el criterio de búsqueda es el total de histórico se mostrará el listado de todos los elementos de ese tipo en el sistema. (b) Si el criterio de búsqueda es por fecha se mostrará el listado de todos los elementos para los que la fecha de búsqueda especificada se encuentra entre la fecha de inicio y fin del registro de histórico. Las cabeceras de los campos disponen de filtros para facilitar la búsqueda de una determinada entidad. 2. Si el usuario establece filtros, se muestran los datos que cumplen las condiciones seleccionadas en los mismos. 3. El listado mostrará un botón para seleccionarla entidad del registro indicado a mostrar así como un aspa en la parte superior para cerrar el listado. (a) Si se pulsa el botón de seleccionar el registro se dará paso al caso de uso B.13 (página 68). (b) Si se pulsa el aspa de cierre se cierra el listado. 4. Finaliza el caso de uso.

Tabla B.10: CU-10 Buscar elementos con histórico

CU-11	**Buscar relación para elementos simples**
Descripción	Obtiene listado de los elementos padre para los que se desea mostrar la relación seleccionada .
Actores	READ, WRITE y ADMIN.
Precondiciones	El usuario ha pulsado el botón de buscar elementos simples para los que mostrar la relación seleccionada.
Postcondiciones	Se muestra un listado con los elementos simples del padre para la relación seleccionada.
Flujo básico	1. El sistema muestra un panel emergente con el listado de todos los elementos del sistema para la entidad seleccionada. Las cabeceras de los campos disponen de filtros para facilitar la búsqueda de una determinada entidad. 2. El listado mostrará un botón para seleccionar la entidad del registro indicado a mostrar así como un aspa en la parte superior para cerrar el listado. (a) Si se pulsa el botón de seleccionar el registro se dará paso a los casos de uso B.14 (página 68 B.15 (página 69 y B.16 (página 70 (b) Si se pulsa el aspa de cierre se cierra el listado. 3. Finaliza el caso de uso.

Tabla B.11: CU-11 Buscar relación para elementos simples

CU-12	**Buscar relación para elementos con histórico**
Descripción	Obtiene listado de los elementos padre para los que se desea mostrar la relación seleccionada.
Actores	READ, WRITE y ADMIN.
Precondiciones	El usuario ha pulsado el botón de buscar elementos con histórico para los que mostrar la relación seleccionada.
Postcondiciones	Se muestra un listado con los elementos con histórico del padre para la relación seleccionada.
Flujo básico	1. El sistema muestra un panel emergente conteniendo el listado de datos de la entidad seleccionada atendiendo a los criterios de búsqueda indicados: (a) Si el criterio de búsqueda es el total de histórico se mostrará el listado de todos los elementos de ese tipo en el sistema. (b) Si el criterio de búsqueda es por fecha se mostrará el listado de todos los elementos para los que la fecha de búsqueda especificada se encuentra entre la fecha de inicio y fin del registro de histórico. Las cabeceras de los campos disponen de filtros para facilitar la búsqueda de una determinada entidad. 2. El listado mostrará un botón para seleccionar la entidad del registro indicado a mostrar así como un aspa en la parte superior para cerrar el listado. (a) Si se pulsa el botón de seleccionar el registro se dará paso a los casos de uso B.15 (página 69 y B.16 (página 70 (b) Si se pulsa el aspa de cierre se cierra el listado. 3. Finaliza el caso de uso.

Tabla B.12: CU-12 Buscar relación para elementos simples

CU-13	**Obtener información de los registros de histórico del elemento**
Descripción	Obtiene listado de los registros de histórico del elemento seleccionado.
Actores	READ, WRITE y ADMIN.
Precondiciones	El usuario ha pulsado el botón de mostrar históricos del elemento seleccionado.
Postcondiciones	Se obtiene el listado con la información de los registros de histórico elemento seleccionado.
Flujo básico	1. El sistema obtienen los registros de histórico del elemento seleccionado. 2. Finaliza el caso de uso.

Tabla B.13: CU-13 Obtener información de los registros de histórico del elemento

CU-14	**Obtener información del padre de la relación**
Descripción	Obtiene la información relevante de la entidad padre de la relación seleccionada.
Actores	READ, WRITE y ADMIN.
Precondiciones	El usuario ha pulsado el botón de seleccionar entidad padre de la relación.
Postcondiciones	Se obtiene la información relevante del padre de la relación seleccionado.
Flujo básico	1. El sistema obtiene la información relevante del padre de la relación. 2. Finaliza el caso de uso.

Tabla B.14: CU-14 Obtener información del padre de la relación

CU-15	**Obtener elementos relacionados**
Descripción	Obtiene el listado de elementos relacionados para la entidad padre.
Actores	READ, WRITE y ADMIN.
Precondiciones	El usuario ha pulsado el botón de seleccionar entidad padre de la relación.
Postcondiciones	Se obtiene un listado con los elementos relacionados con el elemento padre seleccionado.
Flujo básico	1. El sistema obtiene el listado de todos los elementos del sistema relacionados con la entidad padre seleccionada. En caso de que los elementos relacionados sean entidades con histórico el listado obtenido atenderá a un criterio de búsqueda determinado: (a) Si el criterio de búsqueda es el total de histórico se mostrará el listado de todos los elementos de ese tipo en el sistema. (b) Si el criterio de búsqueda es por fecha se mostrará el listado de todos los elementos para los que la fecha de búsqueda especificada se encuentra entre la fecha de inicio y fin del registro de histórico. 2. Finaliza el caso de uso.

Tabla B.15: CU-15 Obtener elementos relacionados

CU-16	**Obtener elementos candidatos**
Descripción	Obtiene el listado de elementos candidatos para la entidad padre.
Actores	READ, WRITE y ADMIN.
Precondiciones	El usuario ha pulsado el botón de seleccionar entidad padre de la relación.
Postcondiciones	Se obtiene un listado con los elementos candidatos a establecer relación con el elemento padre seleccionado.
Flujo básico	1. El sistema obtiene el listado de todos los elementos del sistema que pueden estar relacionados con la entidad padre seleccionada pero para los que todavía no se ha establecido relación. En caso de que los elementos relacionados sean entidades con histórico el listado obtenido contendrá los datos de los registros con la menor fecha de inicio. 2. Finaliza el caso de uso.

Tabla B.16: CU-16 Obtener elementos candidatos

CU-17	**Añadir relación**
Descripción	Establece relación entre entre la entidad candidata y la entidad padre.
Actores	WRITE y ADMIN.
Precondiciones	El usuario ha pulsado el botón de añadir entidad a la relación.
Postcondiciones	Se añade el elemento a la relación de entidades.
Flujo básico	1. El sistema establece la relación entre la entidad padre y la entidad candidata seleccionada. Se añade la entidad candidata al listado de relaciones y se elimina del listado de candidatos. 2. Finaliza el caso de uso.

Tabla B.17: CU-17 Añadir relación

CU-18	**Eliminar relación**
Descripción	Elimina la relación existente entre la entidad seleccionada y la entidad padre.
Actores	WRITE y ADMIN.
Precondiciones	El usuario ha pulsado el botón de eliminar entidad de la relación.
Postcondiciones	Se elimina el elemento de la relación de entidades.
Flujo básico	1. El sistema elimina la relación entre la entidad padre y la entidad candidata seleccionada. Se elimina la entidad candidata al listado de relaciones y se añade al listado de candidatos. 2. Finaliza el caso de uso.

Tabla B.18: CU-18 Eliminar relación

Casos de uso Gestionar ficha personal Este caso de uso define las funcionalidades de gestión de la ficha personal del usuario: cambiar su contraseña o sus datos de contacto.

CU-19	**Mostrar ficha personal**
Descripción	Muestra la ficha personal del usuario.
Actores	READ, WRITE y ADMIN.
Precondiciones	El usuario se encuentra conectado actualmente.
Postcondiciones	Se muestra la ficha con los datos del usuario.
Flujo básico	1. El sistema muestra la información de la ficha del usuario. Se muestra un botón para modificar la contraseña y otro botón para editar los datos de contacto del usuario. 2. Si el usuario selecciona la opción de cambiar contraseña se da acceso al caso de uso B.20 (página 72) y si selecciona la opción de editar se da acceso al caso de uso B.21 (página 73) . 3. Finaliza el caso de uso.

Tabla B.19: CU-19 Mostrar ficha personal

CU-20	**Cambiar contraseña**
Descripción	Cambia la contraseña.
Actores	READ, WRITE y ADMIN.
Precondiciones	El usuario ha pulsado el botón cambiar contraseña de la página de ficha personal.
Postcondiciones	Se realiza el cambio de contraseña del usuario.
Flujo básico	1. El sistema muestra dos campos: uno para introducir la contraseña y otro para confirmar la contraseña introducida. Se muestran dos botones, uno para guardar los cambios y otro para cancelar. 2. El usuario introduce los datos relativos a la nueva contraseña. 3. (a) Si el usuario selecciona la opción de guardar la nueva contraseña, el sistema valida que coincidan los datos de ambos campos y se guarda la nueva contraseña. i. ***Flujo alternativo:*** *Si no se cubre algún campo obligatorio o se produce algún error de validación el sistema informa del error.* (b) Si el usuario selecciona la opción de cancelar se da al caso de uso anterior B.19 (71). 4. Finaliza el caso de uso.

Tabla B.20: CU-20 Cambiar contraseña

CU-21	**Editar ficha personal**
Descripción	Editar ficha personal.
Actores	READ, WRITE y ADMIN.
Precondiciones	El usuario ha pulsado el botón editar ficha personal de la página de ficha personal.
Postcondiciones	Se realizan los cambios realizados por el usuario en la ficha personal.
Flujo básico	1. El sistema muestra varios campos editables y dos botones, uno para guardar los cambios y otro para cancelar. 2. El usuario introduce los nuevos valores de los datos que quiere modificar. 3. (a) Si el usuario selecciona la opción de guardar los cambios, el sistema valida que cumplan los criterios esperados y se guarda la nueva contraseña. i. ***Flujo alternativo:*** *Si no se cubre algún campo obligatorio o se produce algún error de validación el sistema informa del error.* (b) Si el usuario selecciona la opción de cancelar se vuelve al caso de uso anterior B.19 (71).

Tabla B.21: CU-21 Editar ficha personal

Casos de uso Acceder a parametrización

Los casos de uso aquí descritos definen las funcionalidades de gestión de los distintos elementos englobados en las entidades de parametrización. Puesto que las funcionalidades son las mismas para todas las entidades de parametrización se muestra un único caso de uso genérico para todas.

CU-22	**Listar elementos de la entidad seleccionada de parametrización**
Descripción	Listar elementos de la entidad seleccionada de parametrización.
Actores	READ, WRITE y ADMIN.
Precondiciones	El usuario está conectado y ha seleccionado del menú la entidad a listar de la parametrización.
Postcondiciones	Se muestra el listado de la entidad seleccionada.
Flujo básico	1. El sistema muestra un listado paginado de todas las entidades del sistema del tipo seleccionado. Las cabeceras de los campos disponen de filtros para facilitar la búsqueda de una determinada entidad. 2. Si el usuario establece filtros, se muestran los datos que cumplen las condiciones seleccionadas en los mismos. 3. Si el usuario se correponde con el actor WRITE o ADMIN se muestra un botón de creación de nuevo elemento de la entidad, y sobre el listado de entidades existente se muestran botones de edición y borrado del registro del listado seleccionado. Todos estos botones dan acceso a sus respectivos casos de uso. 4. Si el usuario selecciona la opción crear nuevo se da paso al B.3 (58. 5. Si el usuario tiene pefil WRITE o ADMIN, selecciona un registro del listado y (a) Pulsa el botón de edición se da paso al caso de uso B.4 (página 59) (b) Pulsa el botón de borrado se da paso al caso de uso B.8 (página 63). 6. Finaliza el caso de uso.

Tabla B.22: CU-22 Listar elementos de entidades de parametrización

Casos de uso Acceder a catálogo

Los casos de uso aquí descritos definen las funcionalidades de gestión de los distintos elementos englobados en las entidades de catálogo.

Las entidades del catálogo se clasifican en dos tipos:

- Entidades simples - aquellas que almacenan un sólo registro por entidad en el sistema.
- Entidades con histórico - aquellas para las que pueden existen distintos registros de histórico.

Puesto que las funcionalidades son las mismas para todas las entidades de catálogo del mismo tipo se muestran dos casos de uso genericos atendiendo a la tipología a tratar.

CU-23	**Listar elementos de la entidad simple seleccionada del catálogo**
Descripción	Listar elementos de la entidad simple seleccionada del catálogo.
Actores	READ, WRITE y ADMIN.
Precondiciones	El usuario está conectado y ha seleccionado del menú la entidad simple a listar del catálogo.
Postcondiciones	Se muestra el listado de la entidad simple seleccionada.
Flujo básico	1. El sistema muestra un listado paginado de todas las entidades del sistema del tipo seleccionado. Las cabeceras de los campos disponen de filtros para facilitar la búsqueda de una determinada entidad. 2. Si el usuario establece filtros, se muestran los datos que cumplen las condiciones seleccionadas en los mismos. 3. Si el usuario se correponde con el actor WRITE o ADMIN se muestra un botón de creación de nuevo elemento de la entidad, y sobre el listado se muestran botones de edición y borrado del registro del listado seleccionado. Dichos botones dan acceso a los correspondientes casos de uso. 4. Si el usuario tiene pefil WRITE o ADMIN y selecciona la opción crear nuevo se da paso al B.3 (58. 5. Si el usuario tiene pefil WRITE o ADMIN, selecciona un registro del listado y (a) Pulsa el botón de edición se da paso al caso de uso B.4 (página 59) (b) Pulsa el botón de borrado se da paso al caso de uso B.8 (página 63). 6. Finaliza el caso de uso.

Tabla B.23: CU-23 Listar elementos de entidades simples del catálogo

CU-24	**Listar elementos de la entidad histórica seleccionada del catálogo**
Descripción	Listar entidad histórica de catálogo seleccionada.
Actores	READ, WRITE y ADMIN.
Precondiciones	El usuario está conectado y ha seleccionado del menú la entidad histórica a listar del catálogo.
Postcondiciones	Se muestra el listado de históricos de la entidad seleccionada.
Flujo básico	1. Se muestran un área de búsqueda con dos opciones, buscar histórico completo o buscar por fecha, y un botón para realizar la búsqueda de la entidad en función del criterio seleccionado. 2. Si el usuario tiene perfil WRITE o ADMIN se muestra un botón para crear una nueva entidad. 3. Si el cliente pulsa el botón de búsqueda se da paso al caso de uso B.10 y a continuación se muestra el listado del elemento seleccionado. 4. Si el usuario tiene pefil WRITE o ADMIN y selecciona la opción crear nuevo se da paso al B.3 (58 y a continuación se muestra el listado del nuevo elemento creado. 5. Si el usuario tiene pefil WRITE o ADMIN, sobre cada registro de histórico se mostrarán tres botones, uno para añadir registro de histórico, otro para editar el registro y otro para borrar el registro. Si el usuario selecciona un registro del listado y (a) Pulsa el botón de añadir se da paso al caso de uso B.7 (página 62) (b) Pulsa el botón de edición se da paso al caso de uso B.4 (página 59) (c) Pulsa el botón de borrado se da paso al caso de uso B.8 (página 63). 6. Finaliza el caso de uso.

Tabla B.24: CU-24 Listar elementos de la entidad histórica seleccionada del catálogo

Casos de uso Acceder a relaciones del catálogo

Los casos de uso aquí descritos definen las funcionalidades de gestión de los distintos elementos englobados en las relaciones existentes entre las distintas entidades de catálogo. Dichas relaciones están definida por un elemento padre y los distintos elementos que guardan

relación con dicho padre.

Atendiendo al tipo de elemento del padre, las relaciones de entidades del catálogo se clasifican en dos tipos:

- Relaciones en las que el padre de la relación es una entidad simple.
- Relaciones en las que el padre de la relación es una entidad con histórico.

Puesto que las funcionalidades son las mismas para todas las relaciones del catálogo del mismo tipo se muestran dos casos de uso generéricos atendiendo a la tipología a tratar.

CU-25	**Listar relaciones siendo el padre la entidad simple seleccionada del catálogo**
Descripción	Listar elementos de la entidad simple seleccionada del catálogo.
Actores	READ, WRITE y ADMIN.
Precondiciones	El usuario está conectado y ha seleccionado del menú la relación de entidad simple a listar del catálogo.
Postcondiciones	Se muestra toda la información relevante de la relación.
Flujo básico	1. El sistema muestra un botón de buscar. Al pulsarlo se da paso al caso de uso B.11 (página B.11). 2. Una vez obtenida la información de la relación se muestra la información relevante de la entidad padre y el listado de entidades relacionadas con el padre. 3. Si el usuario tiene perfil WRITE o ADMIN se muestra el listado de entidades candidatas a formar relación con la entidad padre y dos botones: uno para añadir una nueva relación con la entidad candidata previamente seleccionada y otro para eliminar una relación existente previamente seleccionada (a) Pulsa el botón de añadir relación se da paso al caso de uso B.17 (página 70) i. ***Flujo alternativo:*** *Si no se ha seleccionado ningún elemento del listado de candidatos el sistema informa de la necesidad de realizar dicha selección.* (b) Pulsa el botón de eliminar relación se da paso al caso de uso B.18 (página 71). i. ***Flujo alternativo:*** *Si no se ha seleccionado ningún elemento del listado de relacionados el sistema informa de la necesidad de realizar dicha selección.* 4. Finaliza el caso de uso.

Tabla B.25: CU-25 Listar relaciones para elementos simples del catálogo

CU-26	**Listar relaciones siendo el padre la entidad con históricos seleccionada del catálogo**
Descripción	Listar elementos de la entidad con históricos seleccionada del catálogo.
Actores	READ, WRITE y ADMIN.
Precondiciones	El usuario está conectado y ha seleccionado del menú la relación de entidad con históricos a listar del catálogo.
Postcondiciones	Se muestra toda la información relevante de la relación.
Flujo básico	1. El sistema muestra un botón de buscar. Al pulsarlo se da paso al caso de uso B.12 (página B.12). 2. Una vez obtenida la información de la relación se muestra la información relevante de la entidad padre y el listado de entidades relacionadas con el padre. 3. Si el usuario tiene perfil WRITE o ADMIN se muestra el listado de entidades candidatas a formar relación con la entidad padre y dos botones: uno para añadir una nueva relación con la entidad candidata previamente seleccionada y otro para eliminar una relación existente previamente seleccionada (a) Pulsa el botón de añadir relación se da paso al caso de uso B.17 (página 70) i. ***Flujo alternativo:*** *Si no se ha seleccionado ningún elemento del listado de candidatos el sistema informa de la necesidad de realizar dicha selección.* (b) Pulsa el botón de eliminar relación se da paso al caso de uso B.18 (página 71). i. ***Flujo alternativo:*** *Si no se ha seleccionado ningún elemento del listado de relacionados el sistema informa de la necesidad de realizar dicha selección.* 4. Finaliza el caso de uso.

Tabla B.26: CU-26 Listar relaciones para elementos con histórico del catálogo

Caso de uso Acceder a instancia

Los casos de uso aquí descritos definen las funcionalidades de gestión de los distintos elementos englobados en las entidades de instancia. Puesto que las funcionalidades son las mismas para todas las entidades de parametrización se muestra un único caso de uso genérico para todas.

CU-27	**Listar la instancia de la entidad seleccionada**
Descripción	Listar la instancia seleccionada.
Actores	READ, WRITE y ADMIN.
Precondiciones	El usuario está conectado y ha seleccionado del menú la instancia de la entidad a listar.
Postcondiciones	Se muestra el listado de la instancia de la entidad.
Flujo básico	1. Se muestra un área de búsqueda con diversos criterios de búsqueda, siendo el de fecha de búsqueda criterio obligatorio y un botón para realizar la búsqueda de la instancia en función del criterio seleccionado. Al ser las instancias elementos jeraquizados entre los criterios de búsqueda estarán los datos que definen a la 2. Si el cliente pulsa el botón de búsqueda se da paso al caso de uso B.10. 3. Una vez obtenida la información de la instancia se muestra la información relevante de la entidad y el listado de históricos de la instancia. 4. Si el usuario tiene perfil WRITE o ADMIN se muestra un botón para crear una nueva entidad. Si pulsa dicho botón se da paso al caso de uso B.3. 5. Si el usuario tiene pefil WRITE o ADMIN y selecciona la opción crear nuevo se da paso al B.3 (58 y a continuación se muestra el listado del nuevo elemento creado. 6. Si el usuario tiene pefil WRITE o ADMIN, sobre el listado de registros de la instancia se mostrarán tres botones, uno para añadir registro de histórico, otro para editar el registro y otro para borrar el registro. Si el usuario selecciona un registro del listado y (a) Pulsa el botón de añadir se da paso al caso de uso B.7 (página 62) (b) Pulsa el botón de edición se da paso al caso de uso B.4 (página 59) (c) Pulsa el botón de borrado se da paso al caso de uso B.8 (página 63). 7. Finaliza el caso de uso.

Tabla B.27: CU-27 Listar la instancia de la entidad seleccionada

Caso de uso Gestionar usuarios

El caso de uso Gestionar usuarios sólo está disponible para el actor ADMIN. A continuación se describe dicho caso de uso

CU-28	**Gestión de usuarios**
Descripción	Lista los usuarios del sistema y permitiendo su edición y borrado, así como la creación de nuevos usuarios.
Actores	ADMIN.
Precondiciones	El usuario está conectado y ha seleccionado del menú la gestión de usuarios.
Postcondiciones	Se muestra el listado de los usuarios del sistema.
Flujo básico	1. El sistema muestra un listado paginado de todos los usuarios del sistema las entidades del sistema. Las cabeceras de los campos disponen de filtros para facilitar la búsqueda de una determinada entidad. 2. Si el usuario establece filtros, se muestran los datos que cumplen las condiciones seleccionadas en los mismos. 3. Se muestra un botón de creación de nuevo elemento de la entidad, y sobre el listado de entidades existente se muestran botones de edición y borrado del registro del listado seleccionado. Todos estos botones dan acceso a sus respectivos casos de uso. 4. Si el usuario selecciona la opción crear nuevo se da paso al B.3 (58. 5. Si el usuario pulsa el botón de edición se da paso al caso de uso B.4 (página 59) 6. Si el usuario pulsa el botón de borrado se da paso al caso de uso B.8 (página 63). 7. Finaliza el caso de uso.

Tabla B.28: CU-28 Gestión de usuarios

B.2 Diagrama E-R

El sistema utiliza una base de datos en PostgreSQL como almacén de datos. En la nomenclatura de las tablas se ha utilizado la separación de palabras con guión bajo.

Todas las tablas salvo las tablas que contienen registros de histórico utilizan como clave primaria un identificador generado de forma incremental mediante secuencias para todos

los casos salvo para la tabla maestra mt_menu, que almacena la información del menú de la aplicación, que es un valor generado de forma manual.

Puesto que las tablas con registros de histórico pueden contener distintos registros con el mismo identificador de para un elemento de la entidad se ha definido una nueva tabla por cada una de estas entidades con registro de histórico que albergará el identificador de cada uno de los elementos de la entidad existentes, este sí generado de forma incremental mediante secuencias. La clave primaria para estas entidades se define como una clave compuesta por el identificador, la fecha de inicio y la fecha de fin.

Ya que las entidades de la aplicación se agrupan en tres grandes grupos: parametrización, catálogo y contratación (instancias) se ha acordado la siguiente nomenclatura para cada una de las tablas del grupo indicado:

- pt_<entidad> hace referencia a las entidades englobadas en los elementos parametrizables (*parameterization table*).
- ct_<entidad>_type hace referencia a las entidades englobadas en el catálogo (*catalog table*).
- it_<entidad> hace referencia a las entidades englobadas en el catálogo (instancias)(*instances table*).
- idt_<entidad> hace referencia a las tablas auxiliares que almacenan el identificador de cada uno de los elementos de la entidad para la que se definen registros de histórico (*identify table*).

A continuación se presentan los distintos diagramas de la E-R. Se han omitido las relaciones con la tabla de estados (pt_status) y de entidades (pt_entity), para hacer más legible los diagramas, ya que la inmensa mayoría de tablas guardan relación con estas tablas a través de una clave foránea.

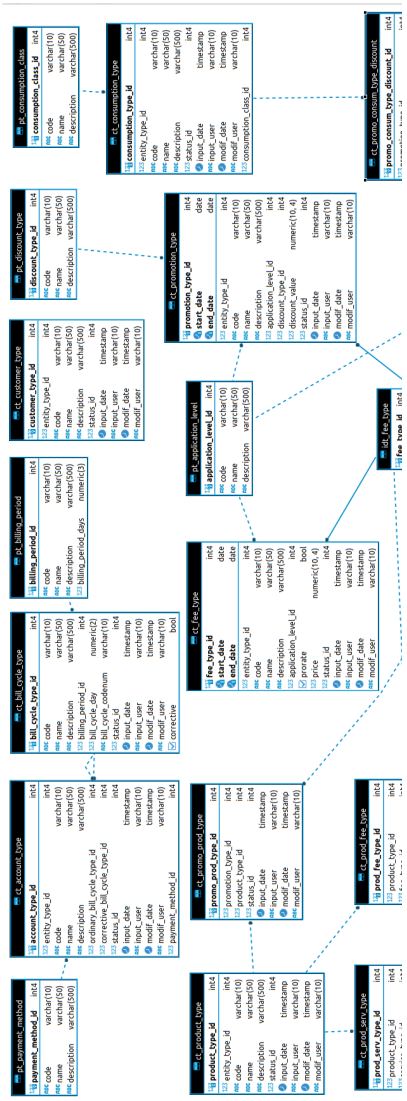


Figura B.5: Diagrama ER del catálogo y la parametrizacion (I)

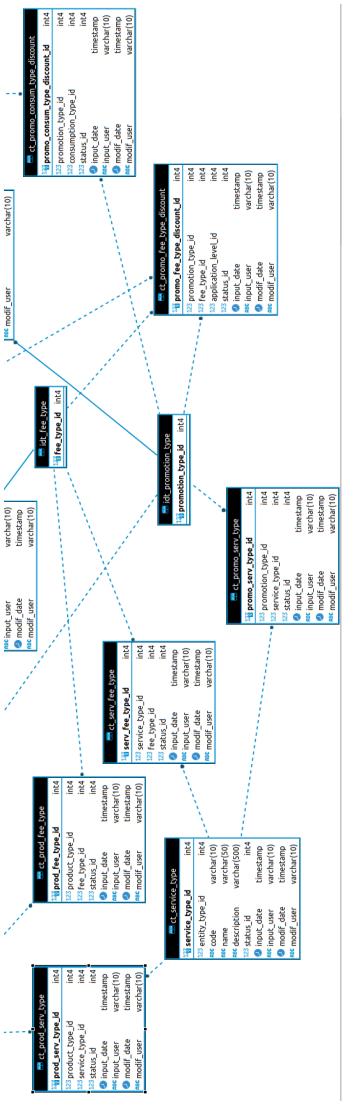


Figura B.6: Diagrama ER del catálogo y la parametrizacion (y II)

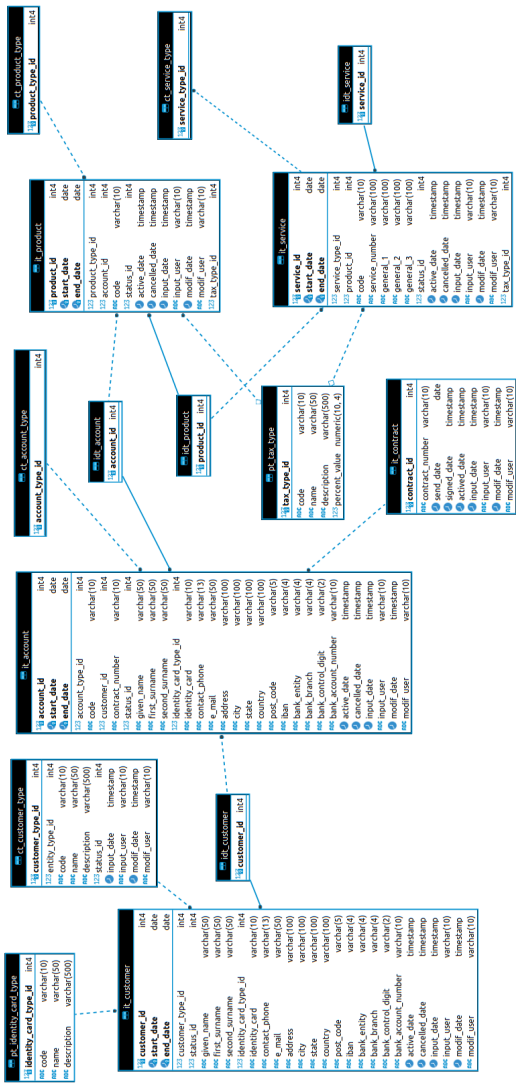


Figura B.7: Diagrama ER de la contratación, el catálogo y la parametrizacion (I)

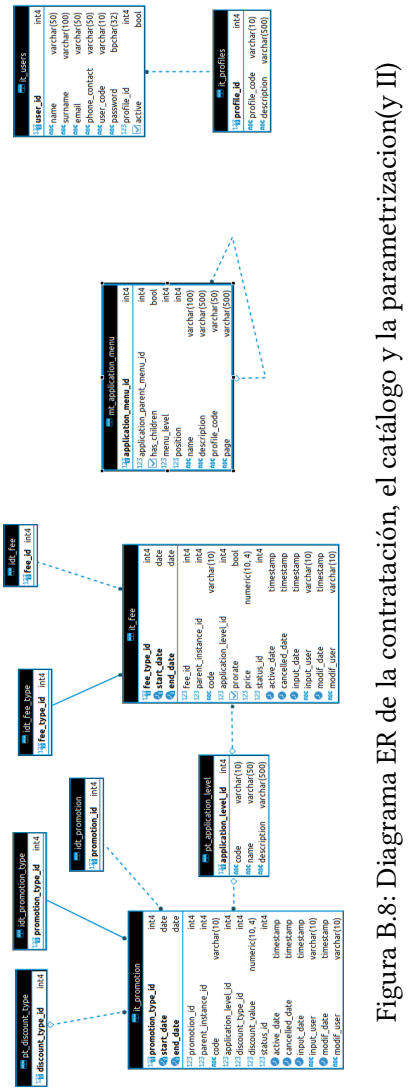


Figura B.8: Diagrama ER de la contratación, el catálogo y la parametrizacion(y II)

B.3 Estructura del código

El proyecto se ha desarrollado usando Maven, por lo que la estructura del mismo es la siguiente:

Carpeta	Descripción
CoMaSw	Carpeta raíz de la aplicación. Contiene el fichero ***pom.xml***, encargado de indicar las diferentes dependencias a nivel de proyecto, posibles tareas que se pueden realizar con el proyecto (compilar, desplegar, etc), así como los módulos de los que puede constar un proyecto.
CoMaSw/src/main/java	Carpeta que almacena los distintos paquetes generados para la aplicación
CoMaSw/src/main/resources	Carpeta que almacena distintos recursos de la aplicación.
CoMaSw/src/main/webapp	Carpeta que almacena las páginas XHTML y ficheros de configuración orientados a la web.
CoMaSwtarget/generated-sources/jooq	Carpeta que almacena la información relativa a las páginas web.

Tabla B.29

A continuación se detalla el contenido de las distintas carpetas.

B.3.1 Directorio /src/main/resources - Recursos de la aplicacion

El directorio ***/src/main/resources*** contiene los distintos recursos usados por la aplicación. A continuación se ofrece un listado del contenido del mismo.

Tabla B.30: Recursos

Paquete	***com.comasw.properties*** Ficheros que contienen las distintas propiedades utilizadas en la aplicación. *dataBaseDefinitions.properties* que contiene información relevante para la identificación de ciertos elementos en la base de datos, *general.properties* que contiene información general para la aplicación. *modifyingProfile.properties* que indica si un perfil tiene o no asociada la funcionalidad de creación y edición de información. *pageTitle.properties* que contiene el nombre de las distintas páginas XHTML creadas. *uiValues.properties* que contiene valores genéricos de componentes usados en las páginas XHTML. *urlPage.properties* que contiene informaciones relativas a distintas urls.
Carpeta	***db*** Contiene los ficheros necesarios para exportar la base de datos.
Fichero	***log4j2.xml*** Fichero de configuración de Log4j.
Carpeta	***META-INF*** Contiene el fichero persistence.xml.

B.3.2 Directorio /target/generated-sources/jooq - Clases generadas

El directorio ***/target/generated-sources/jooq*** contiene las distintas clases generadas por el generador de clases de JOOQ. A continuación se presenta el listado con el contenido del mismo

Tabla B.31: Clases generadas

Paquete	***com/comasw/model*** Contiene las distintas clases que definen la base de datos. *DefaultCatalog.java*, *Keys.java*, *Public.java*, *Routines.java*, *Sequences.java*, *Tables.java*,

Paquete	***com.comasw.model.tables*** Contiene las distintas clases que definen las distintas tablas y vistas de la base de datos. *CtAccountType.java*, *CtBillCycleType.java*, *CtConsumptionType.java*, *CtCustomerType.java*, *CtFeeType.java*, *CtProdFeeType.java*, *CtProdServType.java*, *CtProductType.java*, *CtPromoConsumTypeDiscount.java*, *CtPromoFeeTypeDiscount.java*, *CtPromoProdType.java*, *CtPromoServType.java*, *CtPromotionType.java*, *CtServFeeType.java*, *CtServiceType.java*, *IdtAccount.java*, *IdtCustomer.java*, *IdtFee.java*, *IdtFeeType.java*, *IdtProductFee.java*, *IdtProduct.java*, *IdtProductPromotion.java*, *IdtProductService.java*, *IdtPromotion.java*, *IdtPromotionType.java*, *IdtServiceFee.java*, *IdtService.java*, *IdtServicePromotion.java*, *ItAccount.java*, *ItContract.java*, *ItCustomer.java*, *ItFee.java*, *ItProduct.java*, *ItProfiles.java*, *ItPromotion.java*, *ItService.java*, *ItUsers.java*, *MtApplicationMenu.java*, *PtApplicationLevel.java*, *PtBillingPeriod.java*, *PtConsumptionClass.java*, *PtDiscountType.java*, *PtEntityType.java*, *PtIdentityCardType.java*, *PtPaymentMethod.java*, *PtStatus.java*, *PtTaxType.java*, *VwAccountInstance.java*, *VwCustomerInstance.java*, *VwFeeInstance.java*, *VwProductFeeType.java* , *VwProductInstance.java*, *VwProductServiceType.java*, *VwPromoConsumTypeDiscount.java*, *VwPromotionFeeTypeDiscount.java*, *VwPromotionInstance.java*, *VwPromotionProductType.java*, *VwPromotionServiceType.java*, *VwServiceFeeType.java*, *VwServiceInstance.java*, *VwUsers.java*.
Paquete	***com.comasw.model.tables.pojos*** Clases POJO de las distintas tablas de la aplicación.
Paquete	***com.comasw.model.tables.daos*** Clases DAO de las distintas tablas de la aplicación.
Paquete	***com.comasw.model.tables.record*** Clases RECORD de las distintas tablas de la aplicación.
Paquete	***com.comasw.model.tables.interfaces*** Interfaces para distintas tablas de la aplicación.