Trabajo Fin de Grado

Grado en Ingeniería Informática

Desarrollo de la capa de Business Intelligence un ERP

Autor

Alejandro Gálvez Vicente

Director

Alfonso Puértolas Marcén

Escuela de Ingeniería y Arquitectura

Universidad de Zaragoza

Septiembre 2015

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. ALEJANDRO GÁLVEZ VICENTE,

con nº de DNI 77131407X en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) GRADO, (Título del Trabajo) DESARROLLO DE LA CAPA DE BUSINESS INTELLIGENCE DE UN ERP,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 22 SEPTIEMBRE 2015

Fdo: ALEJANDRO GALVEZ

Desarrollo de la capa de Business Intelligence del ERP EON.BS

Resumen Ejecutivo

Desde la aparición de los primeros aparatos electrónicos se ha buscado una forma de aplicarlos a la empresa y obtener beneficios de su potencia de cálculo. Para aprovechar dicha potencia se fueron desarrollando diferentes programas de carácter empresarial, como los MPC o los MRP. Conforme la potencia de las computadoras aumentaba, mayor era la capacidad de gestión de estos programas. Sin embargo la aparición de internet y el boom de la conectividad provocaron que todos los sistemas existentes hasta la fecha se volvieran obsoletos y fuera incapaces de generar valor frente a las empresas de la competencia.

Los ERP se desarrollaron para solventar este problema. Los ERP (Enterprise Resource Planning) integran una suite de productos que engloban programas como la gestión de la cadena de suministros (SCM) o la gestión de los clientes (CRM). Sin embargo, estos ERP sufren un problema clave: La representación de la información. Un ERP es un sistema gigantesco que trabaja con enormes volúmenes de información, los cuales son cada vez mayores.

Para solucionar este problema de representación de la información, se crearon los KPI (Key Performance Indicators) los cuales muestran de forma gráfica elementos clave en la gestión de la empresa. Además, estos sistemas cuentan con una “capa” especializada en el tratamiento de la información (capa de Business Intelligence), de forma que se obtiene información relevante de todos los datos con los que trabaja el ERP.

En este proyecto se ha programado la capa de Business Intelligence del ERP EON.BS y se han implementado 20 KPIs divididos en las categorías: interna, aprendizaje, financiera y clientes. De esta forma, los clientes dicho ERP contarán de base con una capa de BI totalmente genérica y válida para la gran mayoría de negocios.

El sistema se ha desarrollado formando parte del equipo de desarrollo del ERP EON.BS y se ha seguido la arquitectura de sistema Modelo-Vista-Controlador. El lenguaje de programación utilizado es Java junto a sus tecnologías orientadas a la empresa (JavaEAR y JavaEE). Para la representación gráfica se ha utilizado el framework “Vaadin”, el cual traduce código Java en JavaScript, de esta forma no es necesario que los usuarios instalen nada en su PC, simplemente deben conectarse al servidor donde se lanza el sistema mediante su navegador de internet. Para la persistencia de datos se ha utilizado una base de datos Oracle Enterprise, que es la misma que utiliza el ERP EON.BS para almacenar toda la información generada por el sistema.

Índice general

Índice de indicadores

Índice de diagramas

Índice de fragmentos de código

1. Introducción

Desde la aparición de los primeros aparatos electrónicos se ha buscado una forma de aplicarlos a la empresa y obtener beneficios de su potencia de cálculo. Los primeros aparatos utilizados en el mundo empresarial fueron los mainframes, éstos tenían potencia de cálculo limitada, aun así se utilizaron para empezar a usar programas de gestión empresarial como los MPC ("Manufacturing, Planning and Control") o los MRP ("Material Requeriment Planning).

Siguiendo el avance informático de la época de los 70s – 80s, se empezaron a desarrollar herramientas de gestión más complejas. Esto provocó que los MRP II ("Manufacturing Resource Planning") empezaran a sustituir a los sistemas MRP como principal sistema de control de manufactura de la empresa. Posteriormente se les agregó la capacidad de planificación, provocando que los MRP II fueran los primeros sistemas en los que se intentó integrar la gestión de materiales y la capacidad de producción. Durante la década de los 90 aparecen los MES ("Manufacturing Execution Systems"), sin embargo no son suficiente para subsistir frente al aumento de la competitividad global que supuso la aparición de internet y de las tecnologías del nuevo milenio.

Los sistemas ERP aparecen a inicios del 2000 y prometen un incremento en las ventas frente a sus predecesores. En un ERP se integran una suite de productos corriendo bajo una arquitectura común: la gestión de la cadena de suministros (SCM), gestión de los clientes (CRM), gestión de inventarios, contabilidad, etc.

Sin embargo, estos ERP sufren un problema clave: La representación de información. Son sistemas gigantescos que gestionan grandes cantidades de información que tiene que ser analizada por los responsables de la empresa. Debido al "boom" de internet y el aumento exponencial de la información disponible a nivel mundial, la representación de la información se convirtió en un elemento básico.

La aparición de los KPIs ("Key performance Indicators"), elementos gráficos que representan la información, permitieron el ahorro de mucho tiempo analizando tablas con datos. Además, estos KPIs se vieron acompañados por el auge del Business Intelligence, que se trata de un nuevo software integrado en el ERP encargado de analizar y tratar los datos e información que dispone una empresa en su poder.

1.1 Objetivos y alcance

Con el fin de agregar la capa de Business Intelligence del ERP EON.BS, el objetivo del proyecto es integrarse en el equipo de desarrollo del ERP EON.BS, propiedad de la empresa TAP Consultoría, y trabajar junto al resto de desarrolladores y los consultores en el desarrollo dicha capa de BI.

Siguiendo con el desarrollo de la capa de BI, será necesario elegir los KPIs (indicadores) adecuados para conseguir una capa genérica que se adapte a la mayor cantidad de clientes posible sin que sea necesario personalizarla. Además se deberá tener en cuenta en que tablas de la base de datos están situados los datos a utilizar en cada indicador e intentar optimizar las consultas realizadas para su extracción.

Por último, se empleará el lenguaje de programación utilizado en el desarrollo del EON.BS, se seguirá la arquitectura Modelo-Vista-Controlador durante el desarrollo y se intentará utilizar un código correcto que siga las pautas marcadas por el director de proyectos de la empresa, además de conseguir robustez antes fallos y posibles acciones imprevistas por parte del usuario del programa final.

Para hacerse una idea de la magnitud del ERP EON.BS, esta imagen muestra de forma resumida todos los módulos con los que cuenta el sistema.

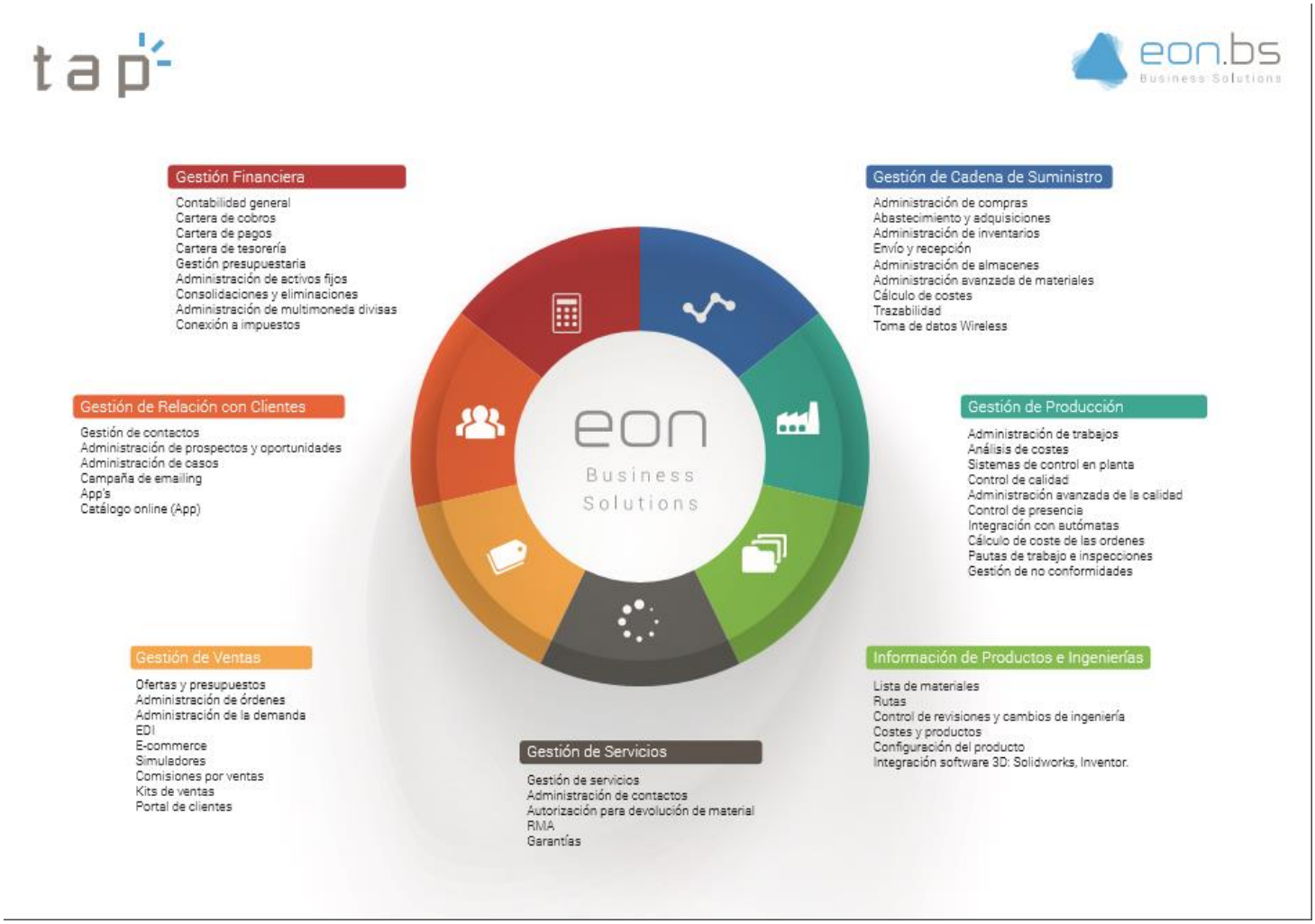


Figure 1: Módulos del ERP EON.BS

1.2 Marco de trabajo

El proyecto se ha realizado en las oficinas de TAP Consultoría situadas en el CEEI, próximo al Campus Rio Ebro. En dicha oficina se encuentra el equipo de consultoría y mantenimiento del ERP EON.BS. Cuando ha sido necesario, me he desplazado a Walqa, centro logístico situado en las aproximaciones de Huesca, para corregir detalles del software con el director de proyectos.

El hecho de haber compartido oficina con este equipo de trabajo significa que he estado en contacto directo con consultores que me han explicado el funcionamiento del día a día de una consultoría. He visto cómo actúan frente a los clientes y cómo es la comunicación “sección de consultoría <-> sección de programación”. También he podido ver en primera mano cómo se actúa frente a clientes que puedan resultar conflictivos o que exijan condiciones no presentes en el contrato.

Mi posición dentro de la oficina estaba junto al resto de programadores. De esta forma he podido preguntar dudas y curiosidades que me han sido resueltas por mis compañeros. He intentado integrarme en el día a día de la empresa compartiendo mi opinión en los temas que se trataban e intentando ayudar a mis compañeros mientras cumplía con mis tareas.

En general, considero que mi tiempo en las oficinas ha sido muy productivo, interesante y útil. No sólo he mejorado mis capacidades como programador, sino que he aprendido como se trabaja en una empresa “de verdad”, he estado en contacto con términos empresariales y de contabilidad y he trabajado en equipo con un grupo de personas muy diferentes a mis compañeros de clase.

1.3 Hoja de ruta

El presente documento contiene:

1. Breve introducción al contexto del proyecto, el entorno de trabajo y los objetivos marcados.
2. Explicación de conceptos clave para una correcta comprensión del alcance del proyecto y cuadrarlo en el contexto empresarial actual.
3. Descripción detallada del proceso de desarrollo de un KPI, mostrando todas las partes involucradas y su rol dentro del proceso de desarrollo.
4. Implementación de los KPIs elegidos por el equipo de desarrollo, divididos en las siguientes categorías:
    a. Categoría interna: Procesos internos de la empresa.
    b. Categoría clientes: Todo lo relacionado con clientes.
    c. Categoría aprendizaje: Información sobre el funcionamiento de la empresa.
    d. Categoría financiera: Información sobre el estado de las finanzas.
5. Diagramas con la arquitectura interna del proyecto:
    a. Vista de módulos: Muestra la división del sistema en distintos subsistemas.
    b. Diagrama de despliegue: Permite observar dónde están situados cada uno de los elementos de forma física.
    c. Diagrama de secuencia: Muestra las secuencia de mensajes y las interacciones entre los componentes del sistema.
6. Breve descripción de la gestión del proyecto, incluyendo metodología, herramientas utilizadas y planificación.
7. Conclusiones extraídas del proyecto y cómo se podría avanzar en un futuro.
8. Bibliografía utilizada para redactar la memoria.
9. Anexos con contenido que no tenía cabida en el resto de apartados.

2. Gestión de los recursos empresariales: El ERP

Un ERP (Enterprise Resource Planning) es un software de gestión empresarial, normalmente una suite de aplicaciones, que una empresa utiliza para recolectar, almacenar, gestionar e interpretar datos de las actividades del negocio. Estas actividades incluyen:

- Planificación de producción.
- Manufactura.
- Marketing y ventas.
- Gestión de inventario.
- Pagos y envíos.

Por tanto, se trata de un elemento crítico que afecta a todas las divisiones de la empresa. Es obligatorio contar con un ERP que se adapte a las necesidades de la empresa y que no suponga una amenaza para los intereses del negocio. Debido a esto, la implementación de un ERP puede convertirse en algo dramático que ha provocado la quiebra de un elevado número de empresas. En general y de forma muy resumida, las fases de implementación de un ERP son:

- Preparación de la empresa. Backups, entrenamiento, etc.
- Configuración del ERP.
- Instalación de extensiones.
- Migración de datos.

Uno de los problemas asociados a los ERP a lo largo de su historia ha sido la usabilidad, la gestión de la información y la representación de ésta. Los ERP se tratan de software demasiado complejo que requieren de muchas horas de entrenamiento para ser usados eficientemente. Además, la forma de extracción de los datos solía hacerse mediante listados de datos, los cuales debían ser interpretados por los gerentes de la empresa.

Uno de los intentos de mejorar estos aspectos de los ERP fue el añadir una nueva capa que se encargara de la gestión y representación de los datos. La capa de Business Intelligence.

2.1 Gestión de la información: Business Intelligence

El término Business Intelligence (BI) se refiere al conjunto de métodos y técnicas utilizados por las organizaciones para realizar decisiones tácticas y estratégicas. Desde un punto de vista más técnico, el BI es un proceso que se encarga de analizar datos y presentar los resultados del análisis de forma gráfica y fácilmente entendible que ayuda a los ejecutivos de la organización, todo ello con el uso de herramientas tecnológicas.

Un BI agrupa una gran cantidad de herramientas, aplicaciones y metodologías que permiten a la organización recolectar datos de fuentes internas y externas, prepararlos para su análisis, desarrollar y ejecutar consultas sobre esos datos que nos devuelvan información útil y, por último, crear informes que dispongan de tablas y gráficas que permitan visualización de los datos y hacer un análisis de los resultados obtenidos.

2.2 Representación de la información: KPIs

Un KPI (Key Performance Indicator), o “indicador de rendimiento” en español, ayuda a una organización a definir y medir el progreso alcanzado respecto a las metas propuestas.

Los KPIs son medidas cuantificables, acordadas de antemano, que refleja los factores de éxito críticos de una empresa. Estos KPIs son elegidos la empresa empresa cliente y el consultor encargado de la instalación del sistema y se suelen utilizar para:

- Medir el nivel de satisfacción de los clientes.
- Rentabilidad de un proyecto.
- Calidad de la gestión de la empresa.
- Tiempo de entrega de productos.
- Etc.

Los KPIs deberán ser:

- Específicos.
- Medibles.
- Alcanzables.
- Relevantes.
- Temporales.

3. Proceso de desarrollo de un KPI

Desarrollar KPIs es un proceso complejo en el que intervienen un elevado número de personas y se debe tener en cuenta la usabilidad del KPI y las necesidades del cliente.

Mantener comunicación constante con el cliente para entender correctamente qué está pidiendo, trabajar conjuntamente con la sección de consultoría y seguir el diseño creado por el equipo de diseñadores se antoja fundamental para desarrollar un buen KPI.

En general, el proceso de desarrollo de un KPI cuenta con 3 partes:

- Captura de las necesidades del cliente.
- Análisis de información y estructuración.
- Fase de desarrollo.

3.1 Captura de las necesidades del cliente

¿Qué KPIs serán útiles para el cliente? Esta es la pregunta clave a la hora de desarrollar los KPIs de la capa de BI. Normalmente, los KPIs a desarrollar serán los que el propio cliente elija, sin embargo, es posible que el cliente tan solo demande un número de indicadores determinado y deje el resto a elección del consultor que lleva la instalación del sistema.

Por esta razón, existen una serie de KPIs “estándar” u “obligatorios” que se implementan en la gran mayoría de sistemas. En otras ocasiones, a estos KPIs hay que realizarles algunas modificaciones para adaptarlos a determinadas particularidades del cliente.

Una correcta toma de requisitos funcionales y no funcionales es indispensable para conseguir un conjunto de KPIs que sean útiles para el cliente.

3.2 Análisis de información y estructuración

Una vez acordados los KPIs con el cliente es el momento de analizar qué información se mostrará en cada uno de los indicadores y cómo se mostrará esta información.

En esta fase entra en juego la cooperación entre las distintas secciones de la empresa:

- Consultoría decide qué información debe mostrarse en cada indicador.
- Programación y consultoría trabajan de forma conjunta en la búsqueda de las tablas necesarias para obtener la información.
- Indican a la sección diseño qué información se extraerá.
- Diseño dará a programación un boceto de cómo deberá ser la estructuración gráfica de la información.

3.3 Fase de desarrollo

Una vez se cuenta con las tablas que contienen la información que se debe mostrar y el boceto de cómo se debe mostrar la información, se entra en la fase de desarrollo del KPI. La sección de programación se encarga de implementar en el sistema el nuevo indicador, integrándolo con el resto del sistema.

El enfoque que se da a la implementación es iterativa e incremental:

- De la toma de requisitos y los casos de uso se encarga la sección de consultoría. Sin embargo, es necesario que la sección de programación revise dichos requisitos y aplique los cambios que crea oportuno.

- Una vez analizados los requisitos y recibido el diseño que deberá tener el indicador, se estudian las clases que se deberán crear para cada indicador y se sitúan en la arquitectura del sistema. La arquitectura del sistema se puede ver con más detalle en el punto 5 de este documento.

- Se implementa el indicador, manteniendo contacto constante con el consultor encargado del proyecto. Entre programación y consultoría se comprueba que las tablas contienen datos correctos, que los datos extraídos tienen sentido, qué datos no valorados en la fase de análisis podría ser conveniente representar, etc. También se discute con la sección de diseño diferentes alternativas a la proporcionada durante la fase de análisis.

- Se procede a la fase de testeo, donde se comprueba junto a consultoría que los datos obtenidos por el indicador son correctos. Si bien parece una fase trivial, es una de las más costosas. Al tratarse de un ERP, la base de datos es gigantesca, esto provoca que comprobar la calidad de los datos o, peor incluso, encontrar la causa de posibles errores, sea costoso en tiempo y recursos.

- Se ejecuta un “commit” en el repositorio del proyecto. El director del proyecto se encarga de comprobar:
    - Se sigue la nomenclatura establecida en el inicio del proyecto.
    - La calidad del código cumple un mínimo.
    - Se han creado las clases necesarias y no se producen redundancias.
    - Las clases están bien estructuradas y siguen diseño descendente.
    - Funciona correctamente al integrar el indicador en el sistema.

Una vez acabada la implementación del KPI, empieza el ciclo de nuevo. Sin embargo, el indicador terminado no “queda en el olvido”. Al analizar los requisitos para un nuevo indicador cabe la posibilidad de encontrar representaciones que podrían ajustarse bien en un indicador acabado. Si esto ocurre, entre programación y consultoría se establecerá los pros y contras de rehacer la UI de un indicador y decidirán.

4. Implementación de los KPIs

Una vez completado el contexto del proyecto y explicado el trabajo que conlleva desarrollar un KPI, es el momento de explicar los KPIs implementados. Los KPIs se han dividido en 4 categorías, dado que no todas las categorías son igual de importantes ni tienen el mismo impacto a la hora de satisfacer las necesidades del cliente, cada una tiene un número diferente de indicadores. Las categorías se dividen en:

- Categoría interna.
- Categoría clientes.
- Categoría aprendizaje.
- Categoría financiera.

Al iniciar sesión, el usuario dispondrá de un menú principal que lista todos los indicadores disponibles:

4.1 Indicadores categoría interna

Los indicadores de la categoría interna muestran información relevante sobre las operaciones de compra de la empresa.

Para representar la información se emplean indicadores que muestran porcentajes. Los indicadores implementados son análogos a algunos de los implementados en la categoría clientes.

- Adelanto medio en la entrega de productos de los proveedores.
- Retraso medio en la entrega de productos de los proveedores.

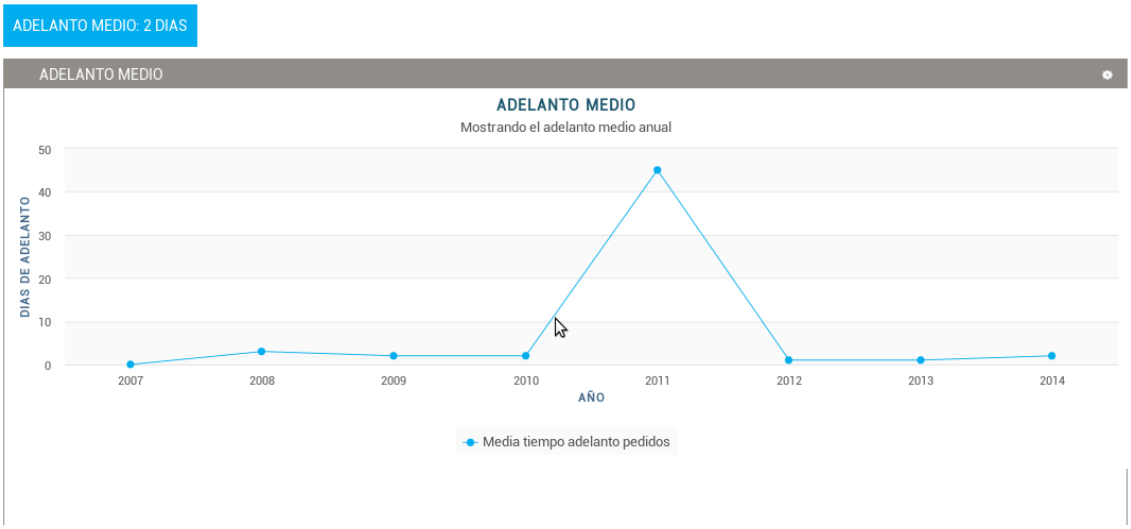
A la hora de mostrar gráficamente los datos en los indicadores, se ha intentado primar la coherencia y uniformidad.

4.1.1 Adelanto medio en la entrega (proveedores)

Muestra el tiempo medio de adelanto en la entrega de pedidos por parte de los proveedores en caso de que se haya producido un adelanto. Es decir, sólo se tienen en cuenta los pedidos a proveedores cuya fecha de entrega del albarán sea anterior a la fecha de entrega indicada en la factura de compra.

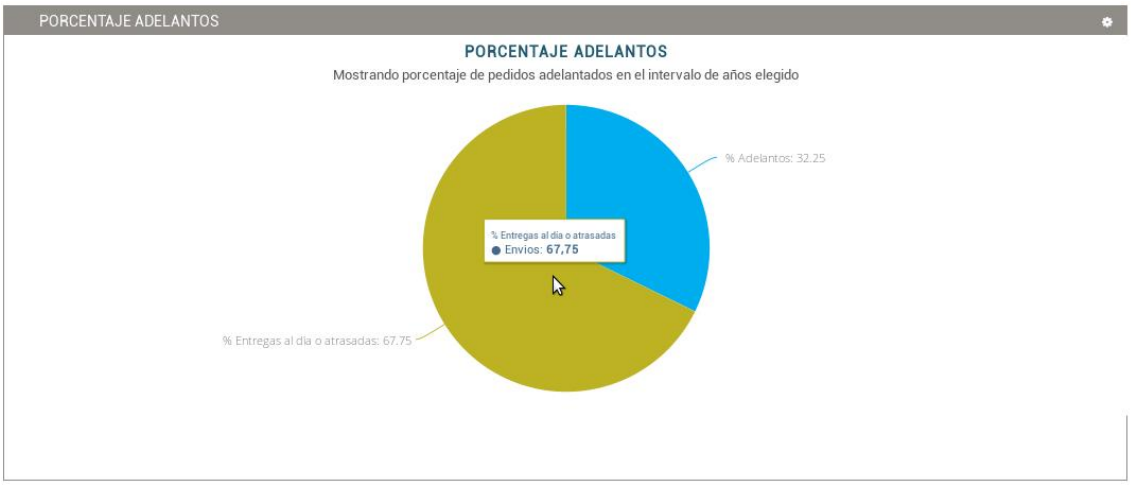
La información se muestra en dos gráficas diferentes:

1. La gráfica de líneas muestra la evolución del tiempo de adelanto a lo largo de los años. La etiqueta azul de la zona superior muestra el adelanto en días del año de la consulta o del último año indicado en el filtro.



Indicador 1: Adelanto medio en la entrega (proveedores) - líneas

2. La gráfica de quesos se muestra un porcentaje con el número de envíos adelantados en el año de la consulta o del último año indicado en el filtro.



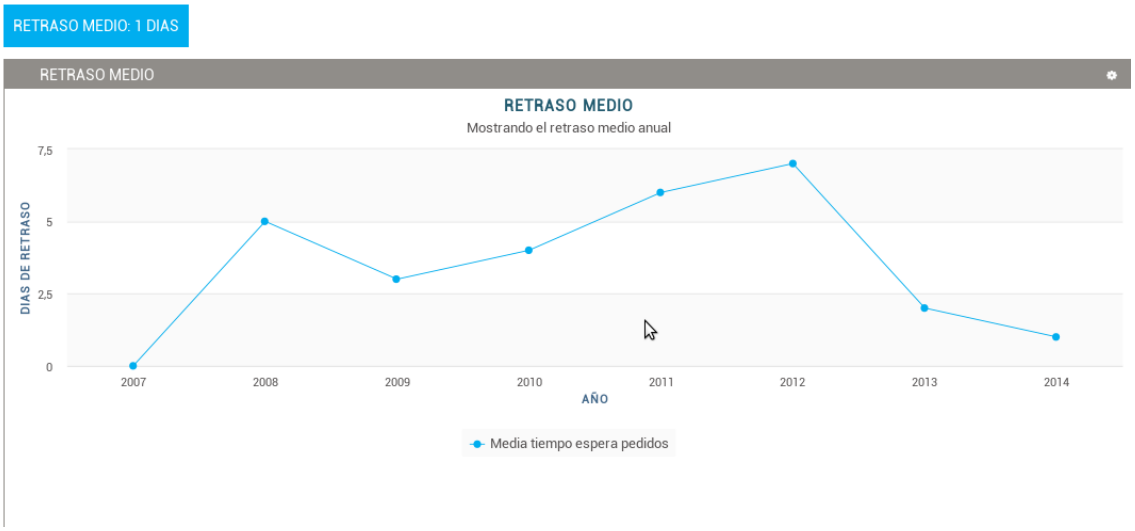
Indicador 2: Adelanto medio en la entrega (proveedores) - quesos

4.1.2 Retraso medio en la entrega (proveedores)

Muestra el tiempo medio de retraso en la entrega de pedidos por parte de los proveedores en caso de que se haya producido un retraso. Es decir, sólo se tienen en cuenta los pedidos a proveedores cuya fecha de entrega del albarán sea posterior a la fecha de entrega indicada en la factura de compra.

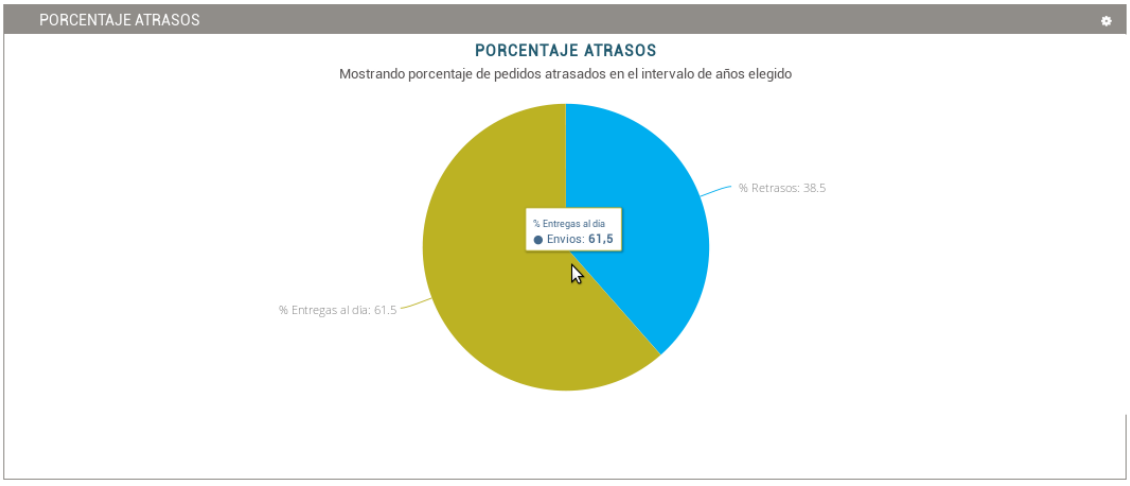
La información se muestra en dos gráficas diferentes:

1. La gráfica de líneas muestra la evolución del tiempo de retraso a lo largo de los años. La etiqueta azul de la zona superior muestra el retraso en días del año de la consulta o del último año indicado en el filtro.



Indicador 3: Retraso medio en la entrega (proveedores) - líneas

2. La gráfica de quesos muestra un porcentaje con el número de envíos retrasados en el año de la consulta o del último año indicado en el filtro.



Indicador 4: Retraso medio en la entrega (proveedores) - quesos

4.2 Indicadores categoría clientes

Los indicadores de la categoría clientes muestran información relevante sobre los clientes de la empresa y las acciones a éstos.

Para representar la información se utilizan múltiples tipos de indicadores: Indicadores que muestran la evolución de valores absolutos a lo largo del tiempo, porcentajes de acciones y tablas con todas las ventas efectuadas.

En total, se han implementado 7 indicadores de la categoría clientes:

- Número de clientes.
- Ventas anuales.
- Ventas mensuales.
- Número de acciones a clientes.
- Ranking de clientes.
- Adelanto medio en la entrega de productos.
- Retraso medio en la entrega de productos.

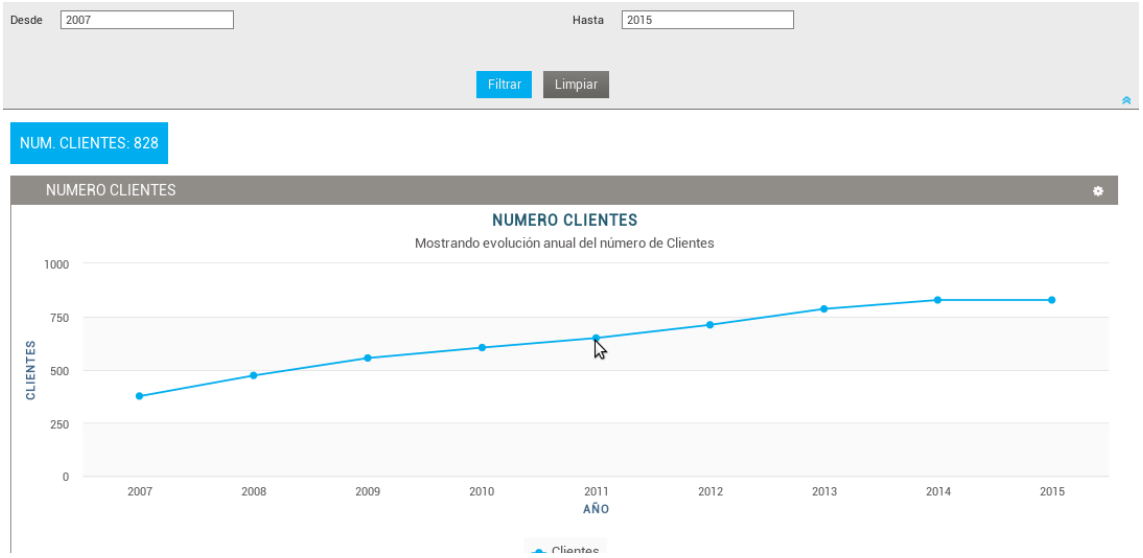
A la hora de mostrar gráficamente los datos en los indicadores, se ha intentado primar la coherencia y uniformidad.

4.2.1 Número de clientes

Muestra la evolución del número de clientes a lo largo de los años elegidos por el usuario en el filtro.

La información se muestra en dos gráficas diferentes:

1. La gráfica de líneas muestra la evolución del número de clientes de la empresa a lo largo de los años. La etiqueta azul de la zona superior muestra el número de clientes del año de la consulta o del último año indicado en el filtro.



Indicador 5: Número de clientes - líneas

2. La gráfica de barras muestra el número de clientes nuevos en cada año del intervalo elegido por el usuario en el filtro.

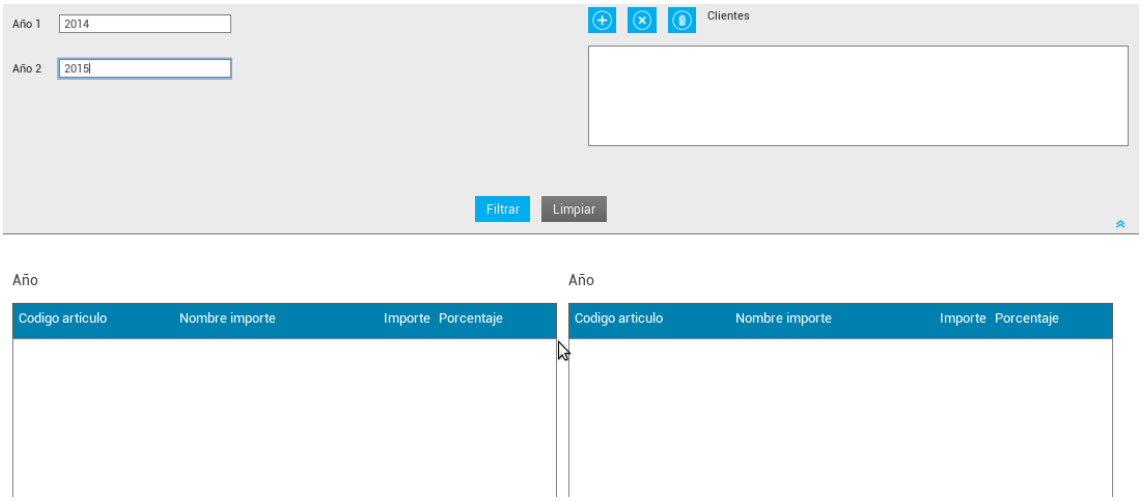
Indicador 6: Número de clientes- barras verticales

4.2.2 Ventas anuales

Se trata de uno de los indicadores más complejo de todos los implementados. Muestra las ventas a un cliente en concreto en los dos años elegidos por el cliente en el filtro. Los datos mostrados pueden ordenarse por código, descripción, importe y porcentaje del importe de un artículo respecto del total.

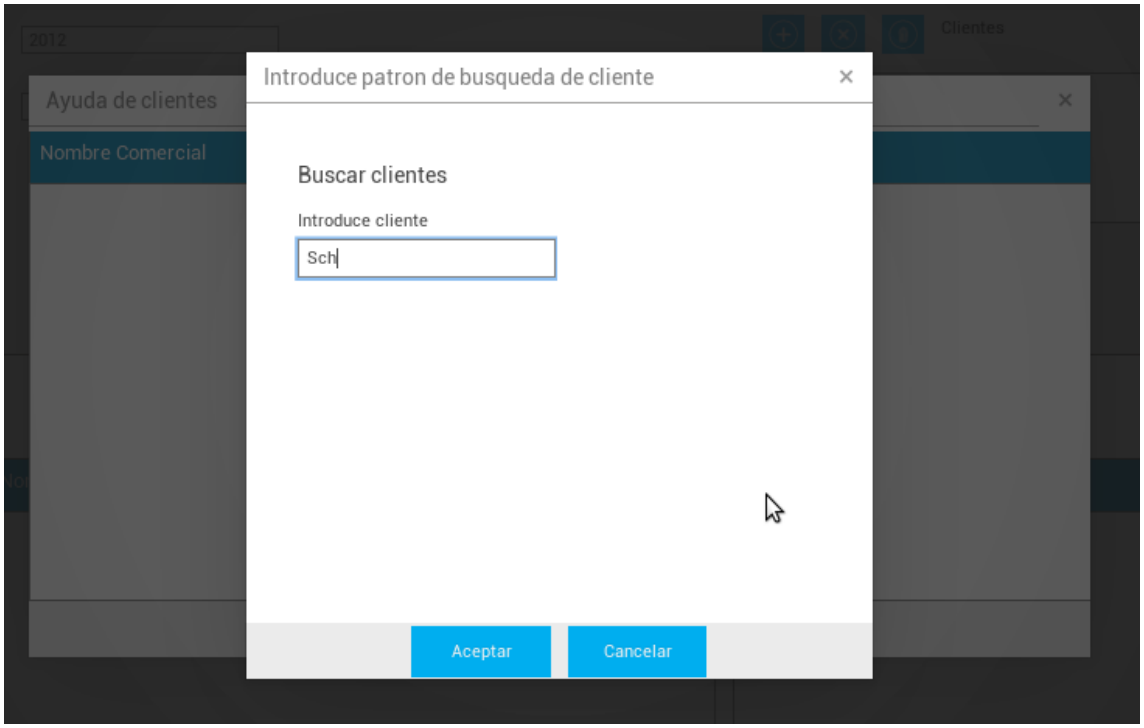
La información se muestra en dos tablas de 4 columnas y un filtro especial que permite seleccionar clientes desde la base de datos (por motivos de privacidad, se ha tachado el importe de los productos listados):

1. Se muestra al cliente el filtro sin clientes, además de los dos años a elegir.



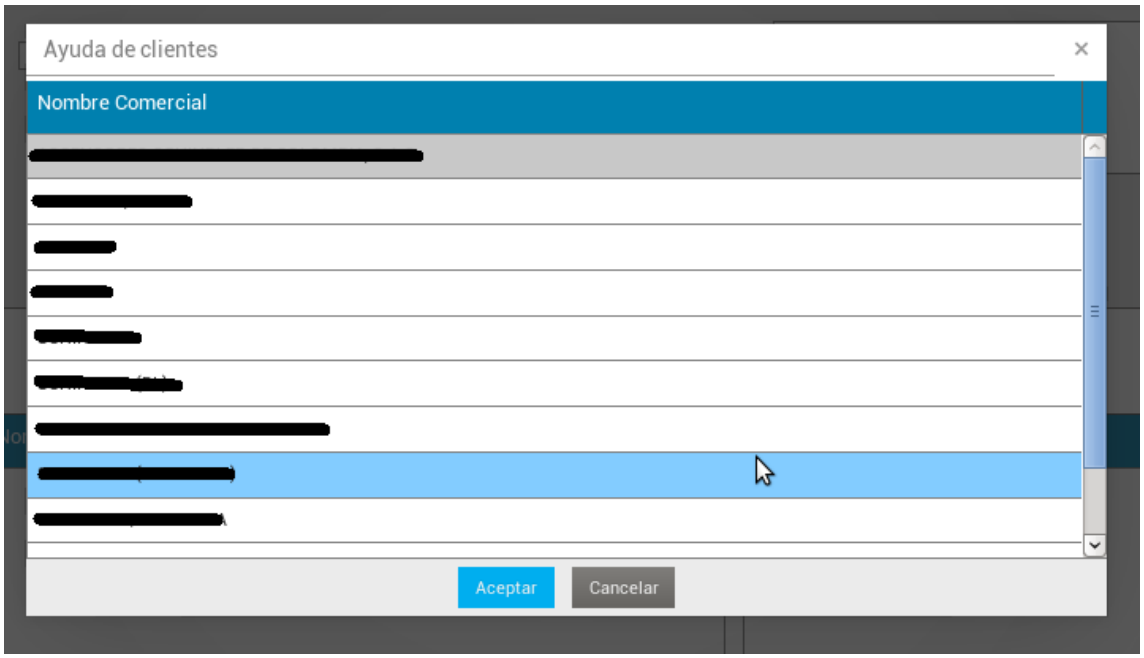
Indicador 7: Ventas anuales - filtro

2. Al pulsar el botón de “Añadir cliente” se pide al usuario que introduzca el nombre del cliente que desea buscar en una caja de búsqueda. No es necesario introducir el nombre completo.



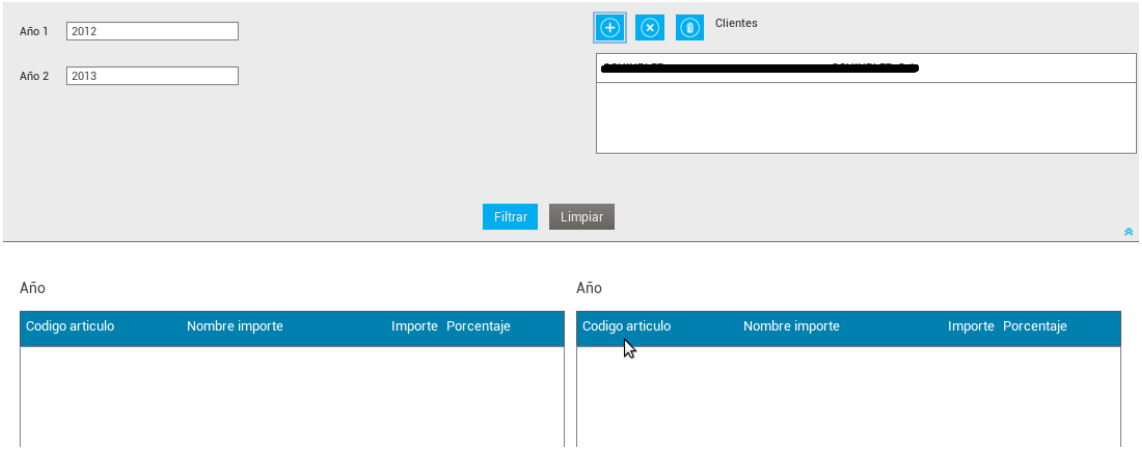
Indicador 8: Ventas anuales - buscar cliente

3. Se realiza una consulta a la base de datos, que devuelve un listado con todos los clientes que contienen el string introducido en la caja de búsqueda.



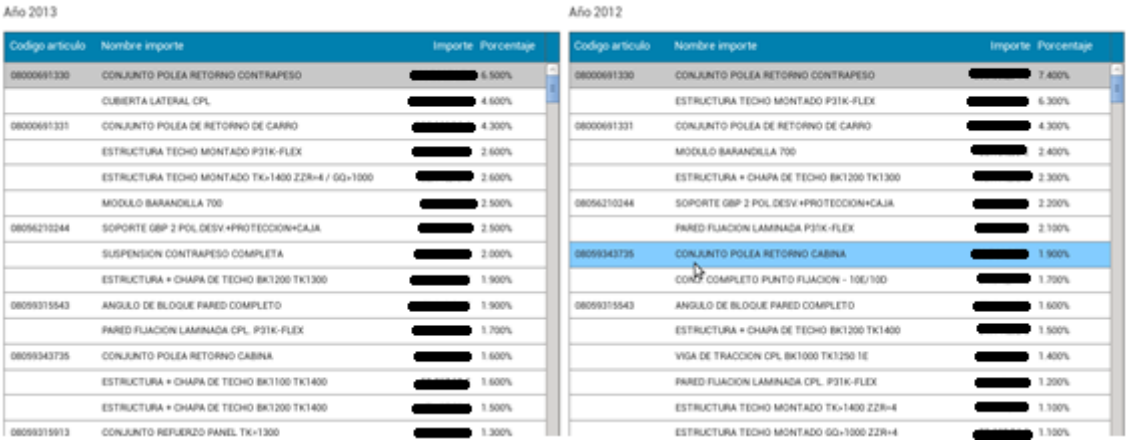
Indicador 9: Ventas anuales - elegir cliente

4. Una vez seleccionado el cliente, éste aparece en el filtro. El usuario puede borrarlo si desea y añadir uno nuevo.



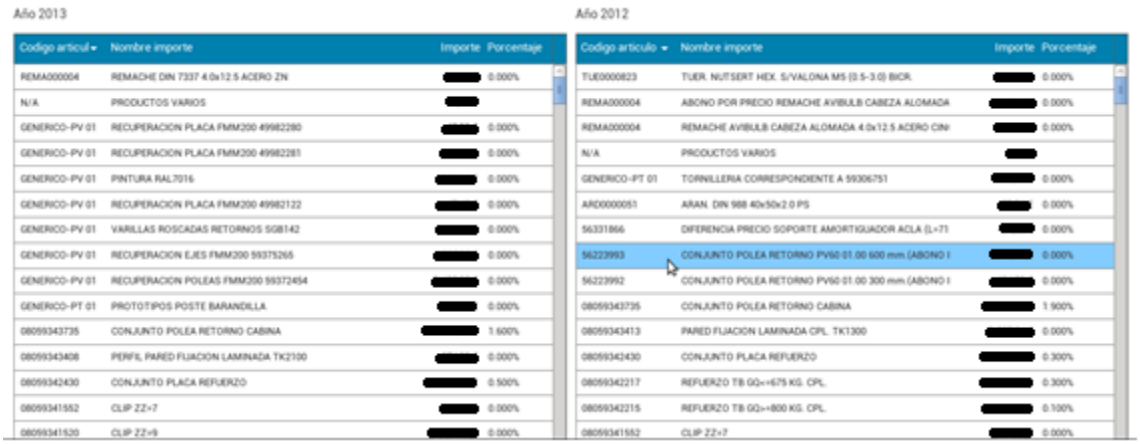
Indicador 10: Ventas anuales - cliente elegido

5. Una vez pulse el botón de filtrado aparecerán los datos de ventas a ese cliente en las tablas. Por defecto, aparecen los artículos ordenados por porcentaje. En la parte superior de la tabla se muestra el año del que se muestran los datos.



Indicador 11: Ventas anuales - filtrado

6. Al pulsar en la cabecera de una columna, los resultados se re-ordenan según la cabecera pulsada.



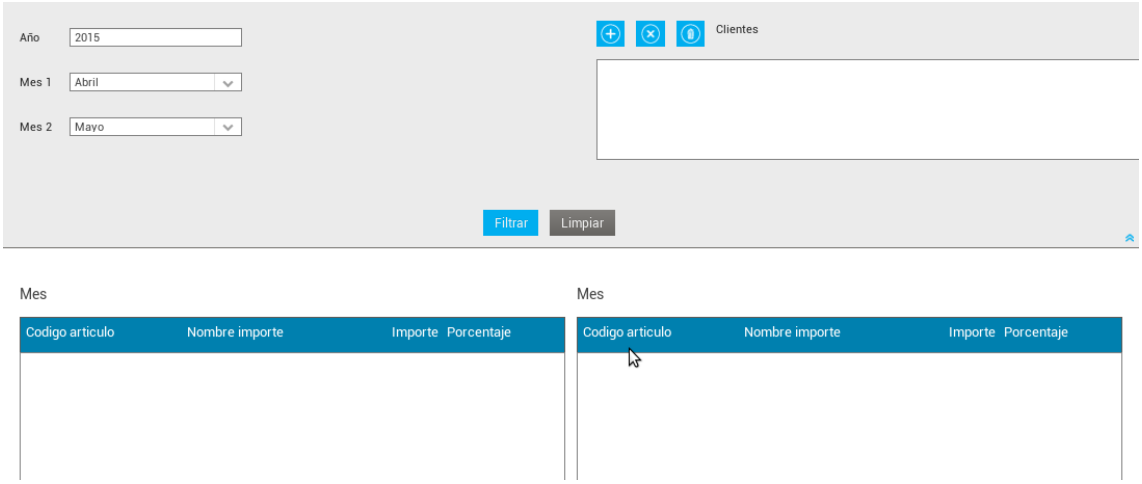
Indicador 12: Ventas anuales - filtrado y ordenado

4.2.3 Ventas mensuales

Se trata de uno de los indicadores más complejo de todos los implementados. Muestra las ventas a un cliente en concreto en los dos meses del año elegidos por el cliente en el filtro. Los datos mostrados pueden ordenarse por código, descripción, importe y porcentaje del importe de un artículo respecto del total.

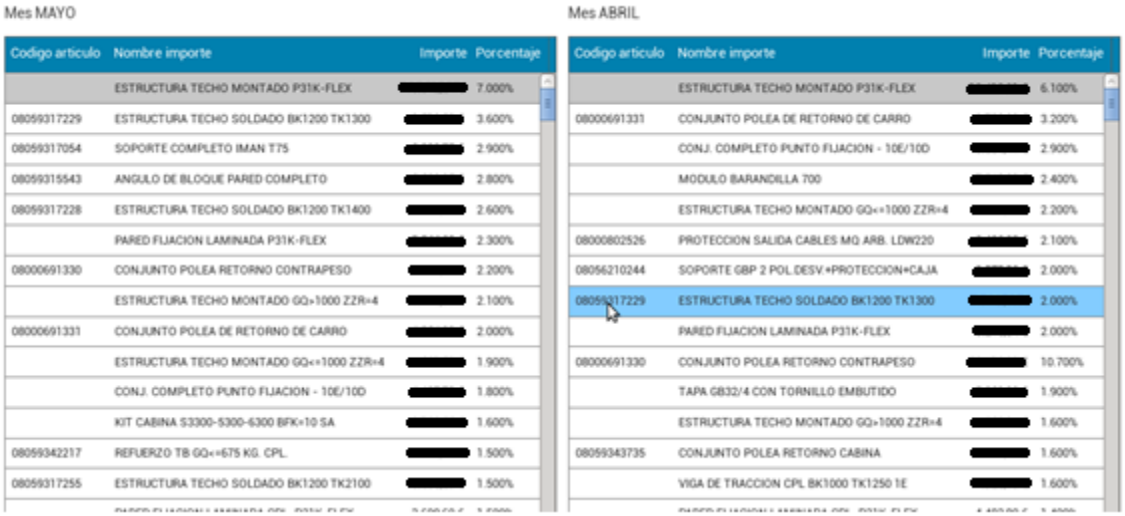
La información se muestra en dos tablas de 4 columnas y un filtro especial que permite seleccionar clientes desde la base de datos. Es idéntica a la del indicador anterior, salvo que se muestran datos mensuales en vez de anuales.

1. Se muestra al cliente el filtro sin clientes, además de los dos meses a elegir y el año.



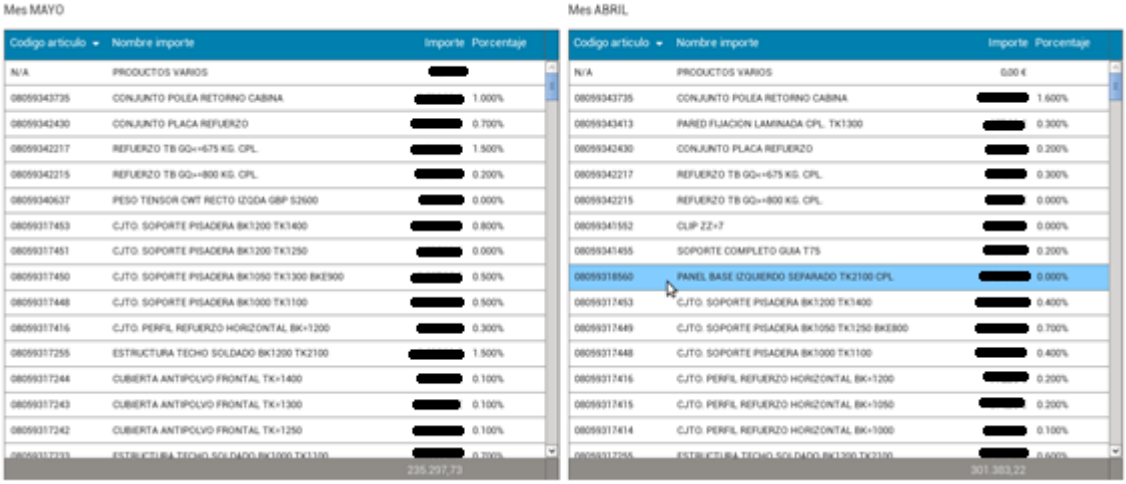
Indicador 13: Ventas mensuales - filtro

2. Una vez filtrado el cliente, se muestran los datos mensuales. Por defecto, aparecen los artículos ordenados por porcentaje. En la parte superior de la tabla se muestra el mes del que se muestran los datos.



Indicador 14: Ventas mensuales - filtrado

3. Al pulsar en la cabecera de una columna, los resultados se re-ordenan según la cabecera pulsada.



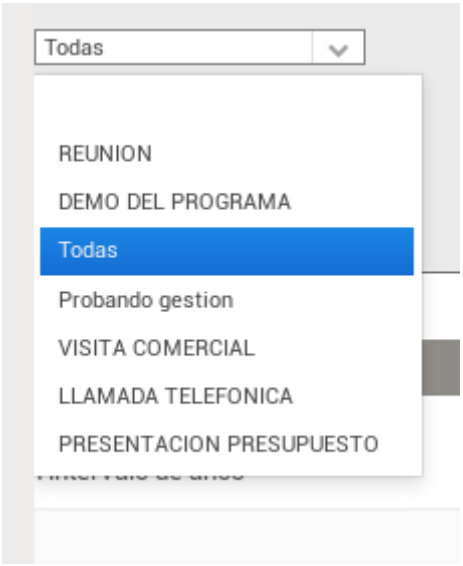
Indicador 15: Ventas mensuales - filtrado y ordenado

4.2.4 Número de acciones a clientes

Muestra la evolución del número de acciones de un determinado tipo elegido por el cliente a lo largo de los años elegidos por el usuario en el filtro, además de mostrar la suma acumulada de cada tipo de acción.

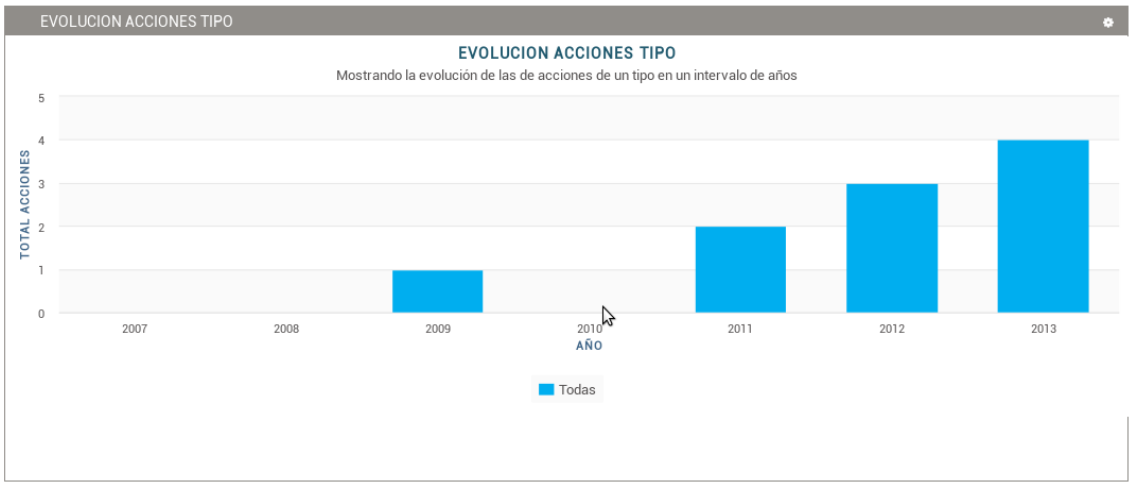
La información se muestra en dos gráficas de barras verticales:

1. El filtro de este indicador cuenta con un selector especial. Los valores de este selector se obtienen mediante una consulta a la base de datos que devuelve los diferentes tipos de acciones ofrecidos a los clientes.



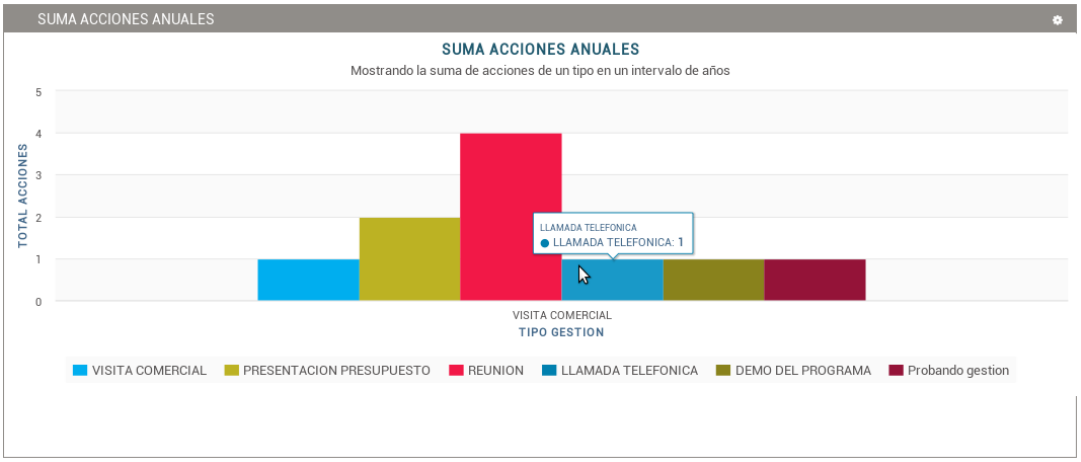
Indicador 16: Número de acciones a clientes - filtro selector

2. La gráfica de barras verticales monocolor muestra la evolución del número de acciones de un tipo a lo largo de los años.



Indicador 17: Número de acciones a clientes - barras verticales

3. La gráfica de barras verticales multicolor muestra el número total de acciones de cada tipo. Los datos son totales dentro del intervalo de años elegido por el usuario en el filtro, es decir, las acciones fuera del intervalo de años elegido no se representan en el indicador.



Indicador 18: Número de acciones a clientes - barras verticales multicolor

4.2.5 Ranking de clientes

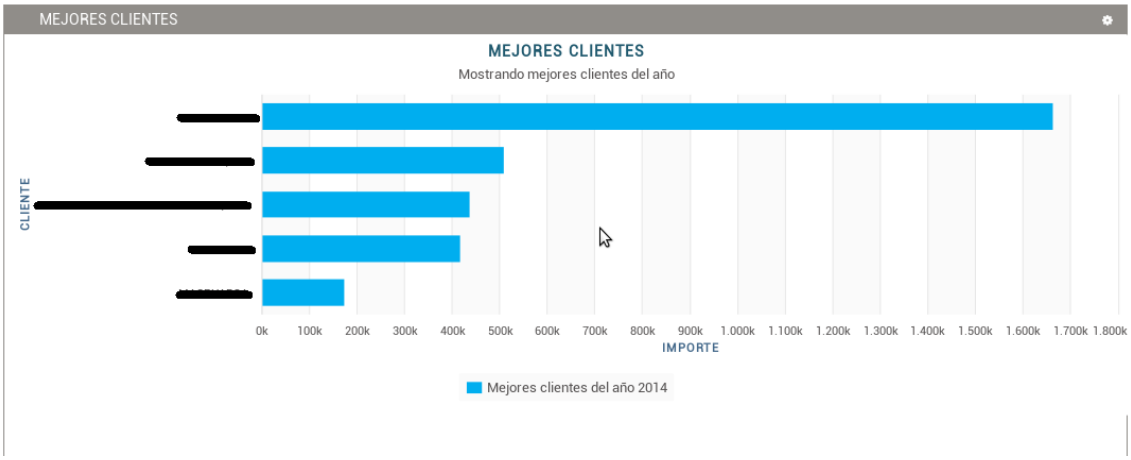
Muestra los mejores clientes de la empresa y el total facturado a dichos clientes en el año elegido por el cliente en el filtro, además de mostrar la evolución de ingresos obtenidos por esos clientes en los últimos años. Es decir, si el cliente “X” gastó “10” en el año 2015, el indicador mostrará el gasto del cliente “X” en los años anteriores, aunque el gasto de dicho cliente no entrara en el TOP5 de esos años.

La información se muestra en dos gráficas de barras verticales (Para este indicador los nombres de los clientes aparecen tachados a petición de la empresa para mantener la confidencialidad del cliente usado como ejemplo):

1. La gráfica de barras multicolor agrupa por año los resultados de cada cliente.

Indicador 19: Ranking clientes - barras verticales multicolor

2. La gráfica de barras horizontales monocolor permite observar los mejores clientes de la empresa y el total facturado a dichos clientes en el último año seleccionado en el filtro.



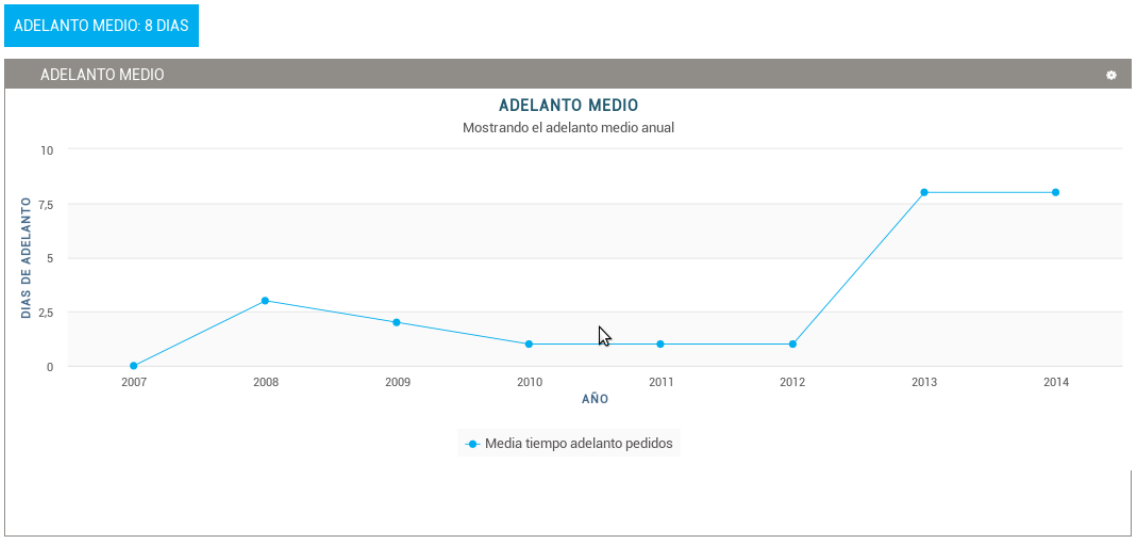
Indicador 20: Ranking clientes - barras horizontales

4.2.6 Adelanto medio en la entrega

Muestra el tiempo medio de adelanto en la entrega de pedidos por parte de la empresa a clientes en caso de que se haya producido un adelanto. Es decir, sólo se tienen en cuenta los pedidos de clientes cuya fecha de entrega del albarán sea anterior a la fecha de entrega indicada en la factura de compra.

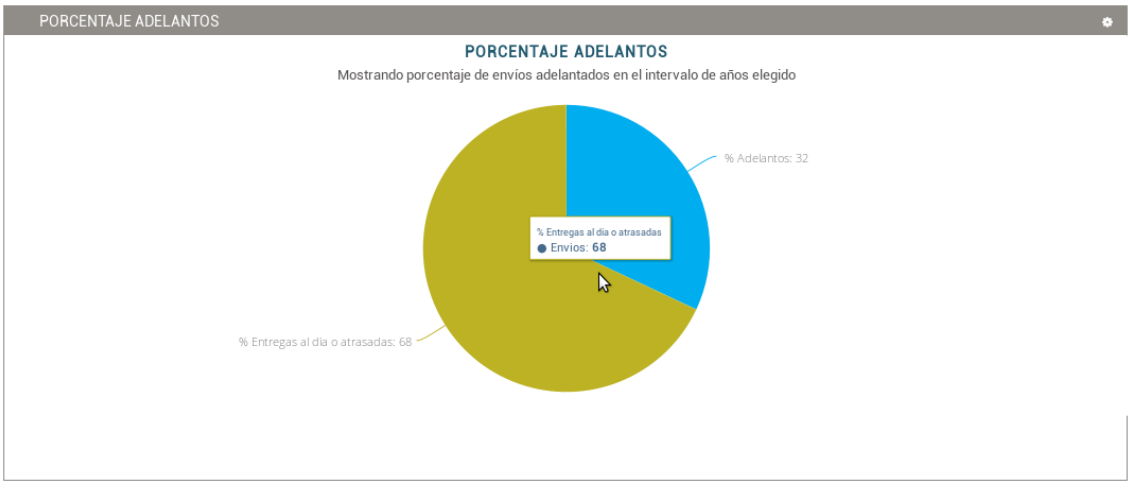
La información se muestra en dos gráficas diferentes:

1. La gráfica de líneas muestra la evolución del tiempo de adelanto a lo largo de los años. La etiqueta azul de la zona superior muestra el adelanto en días del año de la consulta o del último año indicado en el filtro.



Indicador 21: Adelanto medio en la entrega - líneas

2. La gráfica de quesos se muestra un porcentaje con el número de envíos retrasados en el año de la consulta o del último año indicado en el filtro



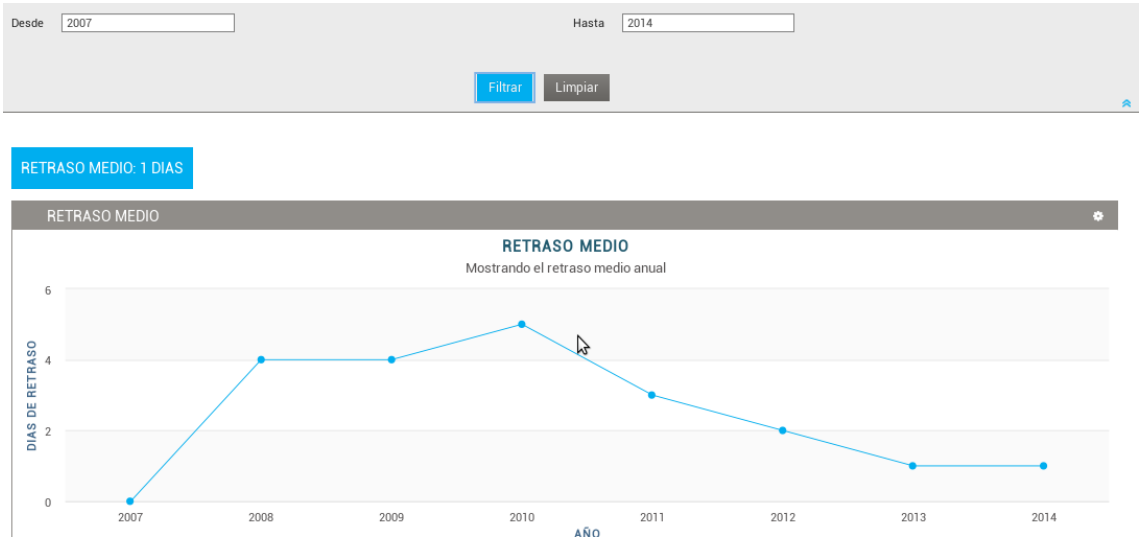
Indicador 22: Adelanto medio en la entrega - quesos

4.2.7 Retaso medio en la entrega

Muestra el tiempo medio de retraso en la entrega de pedidos por parte de la empresa a clientes en caso de que se haya producido un retraso. Es decir, sólo se tienen en cuenta los pedidos de clientes cuya fecha de entrega del albarán sea posterior a la fecha de entrega indicada en la factura de compra.

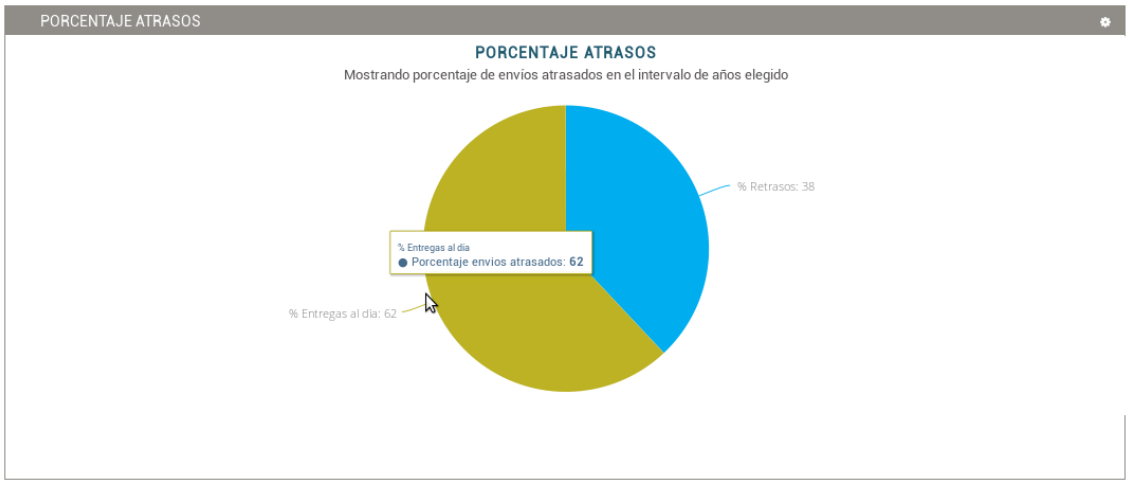
La información se muestra en dos gráficas diferentes:

1. La gráfica de líneas muestra la evolución del tiempo de retraso a lo largo de los años. La etiqueta azul de la zona superior muestra el retraso en días del año de la consulta o del último año indicado en el filtro.



Indicador 23: Retraso medio en la entrega - líneas

2. La gráfica de quesos muestra un porcentaje con el número de envíos retrasados en el año de la consulta o del último año indicado en el filtro.



Indicador 24: Retraso medio en la entrega - quesos

4.3 Indicadores categoría aprendizaje

Los indicadores de la categoría aprendizaje muestran información relevante sobre el funcionamiento de la empresa y sus empleados en el momento de la consulta.

Para representar la información se emplean dos tipos de indicadores: Los que muestran valores absolutos que evolucionan a lo largo del tiempo y los que muestran porcentajes relativos a los datos.

En total, se han implementado 4 indicadores de aprendizaje:

- Número de empleados.
- Edad media de los empleados.
- Porcentaje de empleados menores de X años.
- Porcentaje de empleados menores de 40 años.

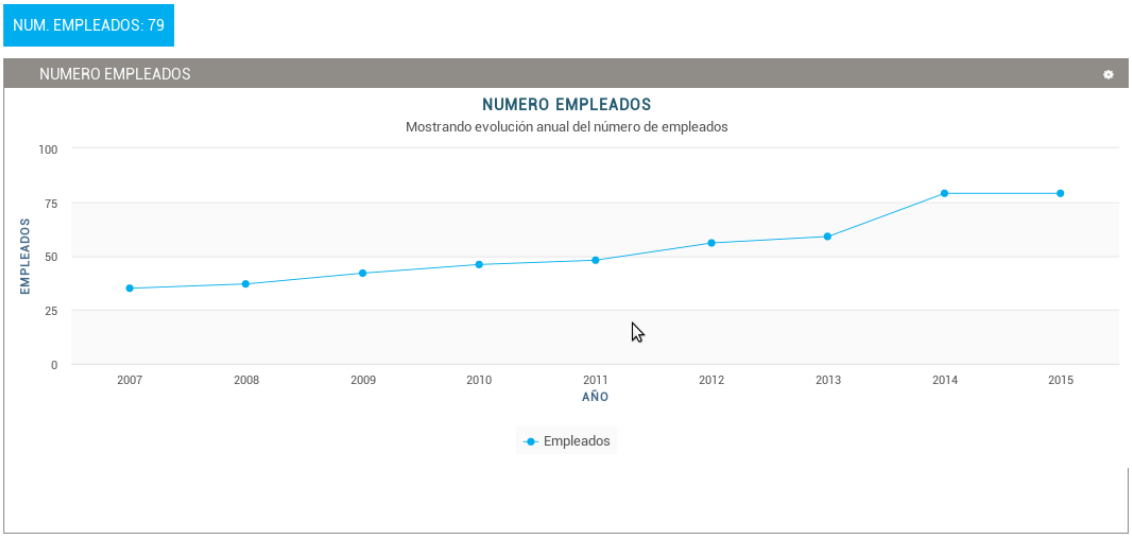
A la hora de mostrar gráficamente los datos en los indicadores, se ha intentado primar la coherencia y uniformidad.

4.3.1 Número de empleados

Muestra la evolución del número de empleados a lo largo de los años elegidos por el usuario en el filtro.

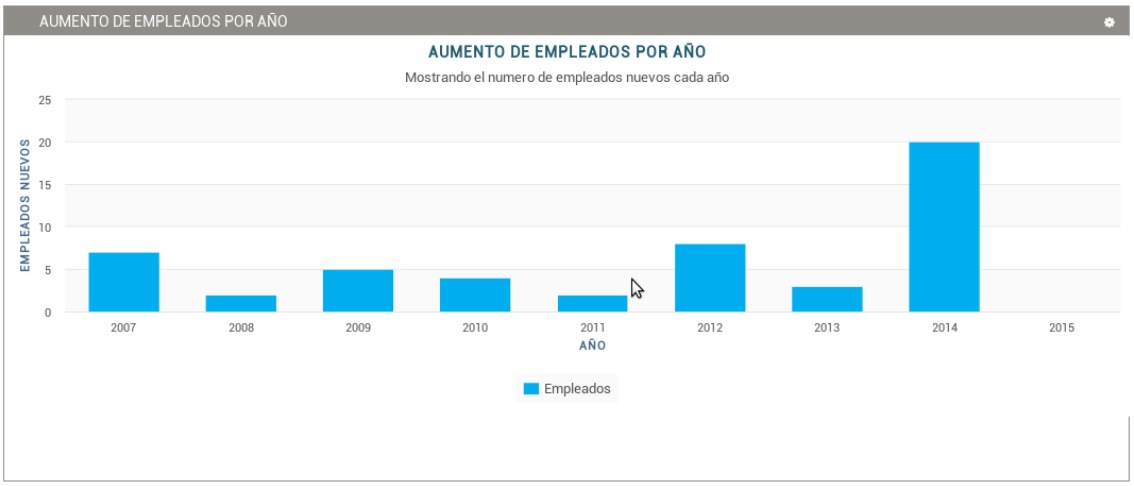
La información se muestra en dos gráficas diferentes:

1. La gráfica de líneas muestra la evolución del número de empleados de la empresa a lo largo de los años. La etiqueta azul de la zona superior muestra el número de empleados del año de la consulta o del último año indicado en el filtro.



Indicador 25: Número de empleados - líneas

2. La gráfica de barras muestra el número de empleados nuevos en cada año del intervalo elegido por el usuario en el filtro.



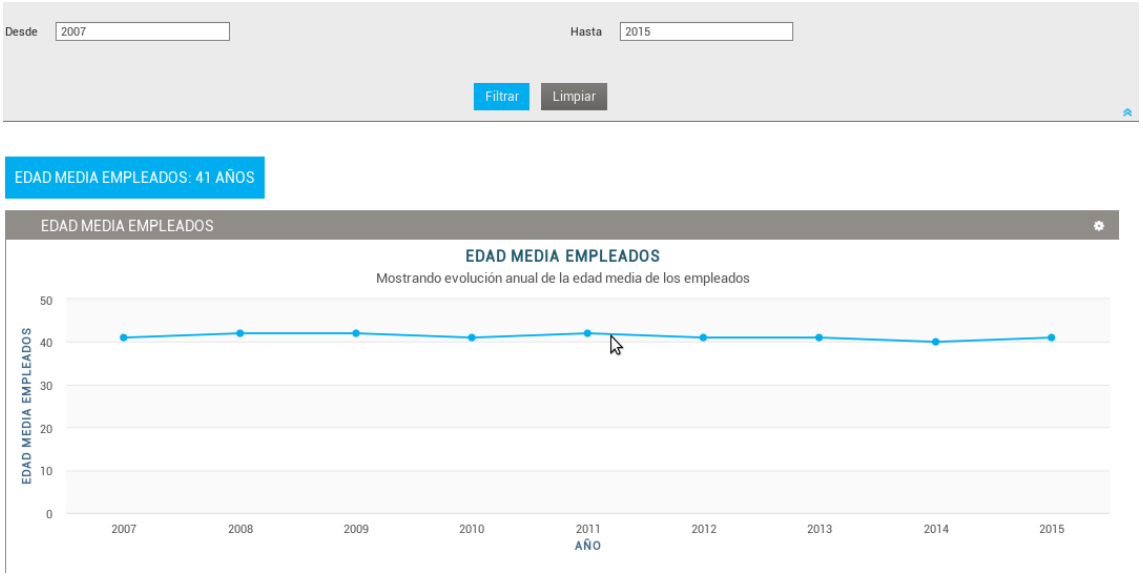
Indicador 26: Número de empleados - barras verticales

4.3.2 Edad media de los empleados

Muestra la evolución de la edad media de los empleados a lo largo de los años elegidos por el usuario en el filtro.

La información se muestra en una gráfica:

1. La gráfica de líneas muestra la evolución de la edad medio de los empleados a lo largo de los años. La etiqueta azul de la zona superior muestra el valor de la edad media de los empleados del año de la consulta o del último año indicado en el filtro.



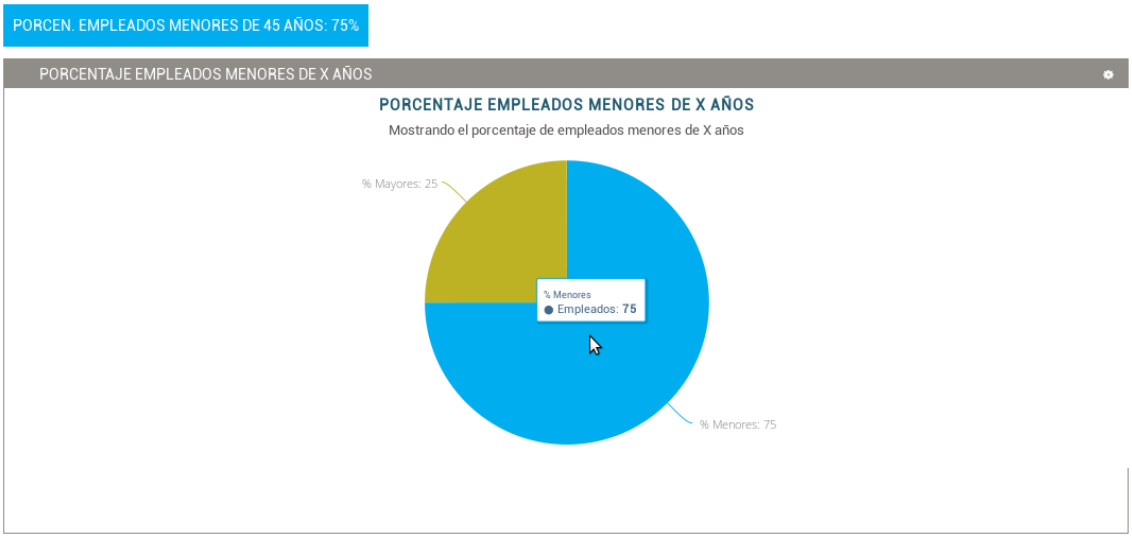
Indicador 27: Edad media de los empleados

4.3.3 Porcentaje empleados menores de X años

Muestra la evolución del porcentaje de empleados mayores y menores de X años, siendo X un valor elegido por el usuario en el filtro.

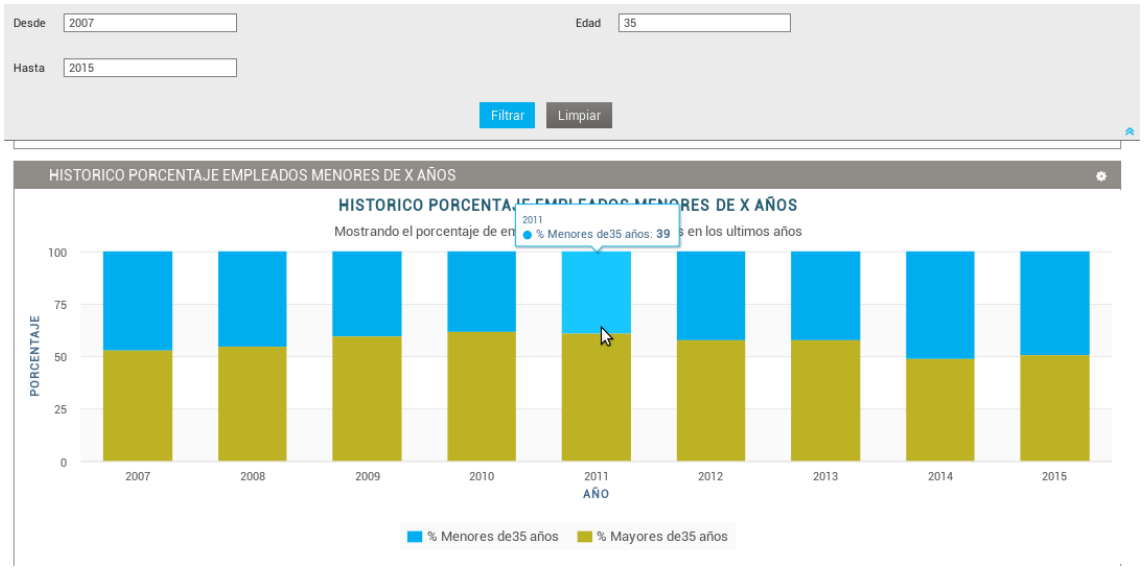
La información se muestra en dos gráficas diferentes:

1. La gráfica de quesos muestra un porcentaje de empleados menores de X años en el año de la consulta o del último año indicado en el filtro. La etiqueta azul de la zona superior muestra porcentaje de forma numérica.



Indicador 28: Porcentaje empleados menores de X años - quesos

2. La gráfica de barras “stacked” bicolor muestra la evolución del porcentaje de empleados mayores y menores de X años en el intervalo de años elegido por el usuario en el filtro.



Indicador 29: Porcentaje empleados menores de X años – barras verticales staked

4.3.4 Porcentaje empleados menores de 40 años

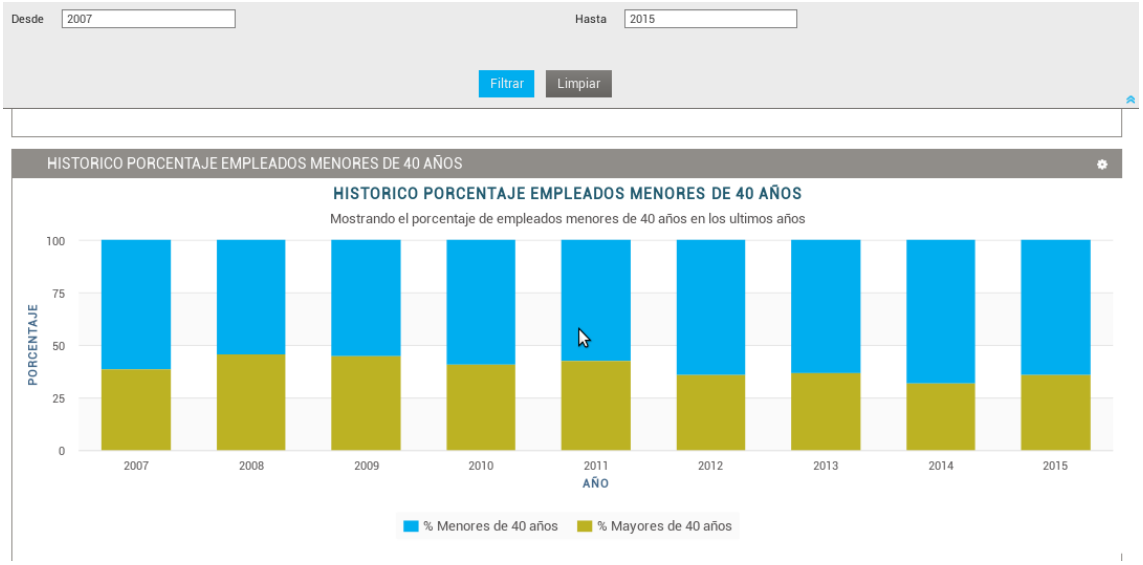
Muestra la evolución del porcentaje de empleados mayores y menores de 40 años.

La información se muestra en dos gráficas diferentes:

1. La gráfica de quesos muestra un porcentaje de empleados menores de 40 años en el año de la consulta o del último año indicado en el filtro. La etiqueta azul de la zona superior muestra porcentaje de forma numérica.

Indicador 30: Porcentaje empleados menores de 40 años - quesos

2. La gráfica de barras “stacked” bicolor muestra la evolución del porcentaje de empleados mayores y menores de 40 años en el intervalo de años elegido por el usuario en el filtro.



Indicador 31: Porcentaje empleados menores de 40 años - barras verticales stacked

4.4 Indicadores categoría financiera

Los indicadores de la categoría financiera muestran información relevante sobre la situación financiera de la empresa en el momento de la consulta.

Para representar la información se utilizan dos tipos de indicadores: Los que muestran la evolución de la información en un intervalo de tiempo y los que muestran la información concreta de un mes.

En total, se han implementado 7 indicadores financieros:

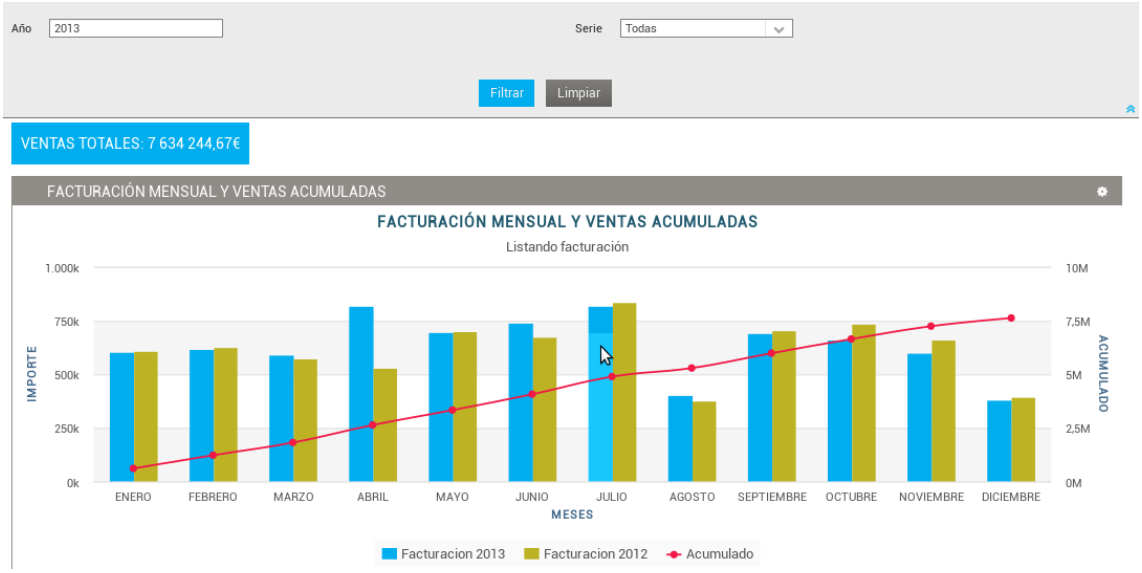
- Facturación mensual y acumulada.
- Resumen anual de ventas y gastos.
- Ratio de liquidez.
- Ratio de autonomía.
- Ratio de endeudamiento.
- Radio de rentabilidad de ventas.
- Balance de situación.

A la hora de mostrar gráficamente los datos en los indicadores, se ha intentado primar la coherencia y uniformidad. Es decir, todos los ratios presentaran un aspecto gráfico similar, al igual que los balances, para no confundir al usuario.

4.4.1 Facturación mensual y acumulada

Muestra la facturación mensual y la facturación acumulada a lo largo de los meses del mes y serie de factura elegidos en el filtro. La información se muestra en una gráfica 2x1:

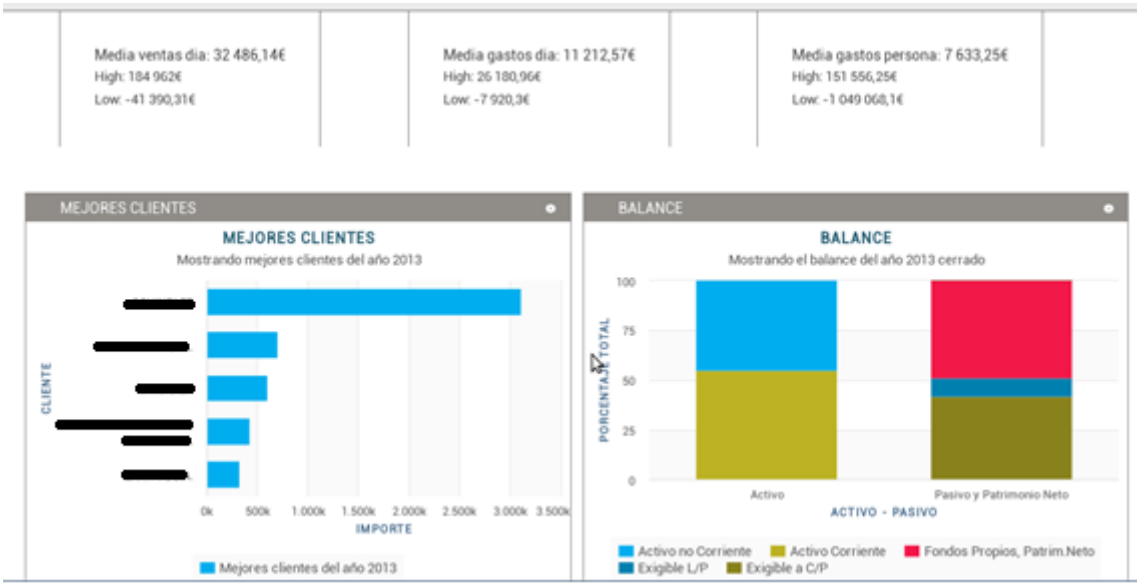
1. La gráfica de barras bicolor muestra la facturación de cada mes del año y serie de factura elegidos.
2. La gráfica de líneas de color rojo muestra la facturación acumulada a lo largo de los meses.



Indicador 32: Facturación mensual y acumulada

4.4.2 Resumen anual de ventas y gastos

Se trata de uno de los indicadores más complejo de todos los implementados. Muestra el estado financiero actual de la empresa. El objetivo es que el usuario sólo necesite un vistazo para saber la información más importante.



Indicador 33: Resumen anual de ventas y gastos

La representación gráfica consta de 5 secciones:

- Zona de ventas: Muestra la media de venta diaria, el mayor gasto mensual y el menor gasto mensual. Si se muestra un valor negativo se trata de una devolución.
- Zona de gastos: Idéntico a la zona de ventas, solo que con gastos.
- Zona de gastos de personal: Muestra el gasto medio en personal de la empresa, además del mes con más gastos en personal y el que menos.
- Zona de mejores clientes: Muestra una gráfica de barras horizontales donde se puede observar los mejores clientes de la empresa y el total facturado a dichos clientes.
- Balance: Muestra un balance típico con el balance de la empresa.

Todos los datos mostrados corresponden al año elegido por el usuario en el filtro y tienen en cuenta la variable de que el año esté “Cerrado” financieramente o siga “Abierto”.

Por último, los nombres de los clientes aparecen tachados a petición de la empresa para mantener la confidencialidad del cliente usado como ejemplo.

4.4.3 Ratio de liquidez

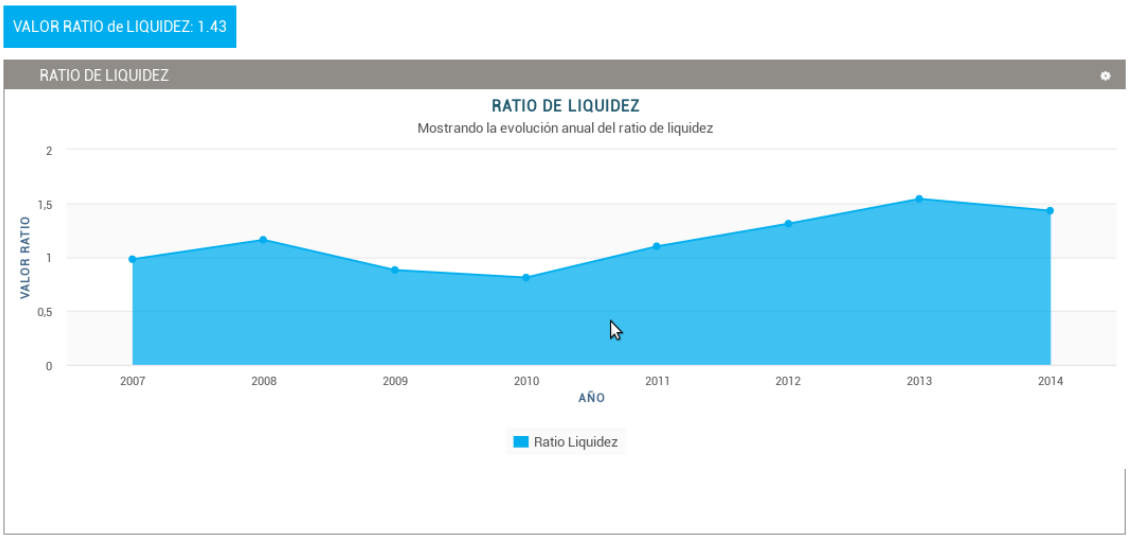
Muestra la evolución del ratio de liquidez a lo largo de un periodo de años elegido por el usuario. En la parte superior se muestra de forma numérica el valor del ratio de liquidez en el último año del periodo.

La fórmula utilizada para obtener el ratio de liquidez es:

```
R. Liquidez = Activo corriente/Pasivo corriente
```

La información se representa en una gráfica:

1. La gráfica de área muestra la evolución del valor del ratio de liquidez a lo largo de los años. La etiqueta azul de la zona superior muestra el valor numérico del ratio en el año de la consulta o del último año indicado en el filtro.



Indicador 34: Ratio de liquidez

4.4.4 Ratio de autonomía

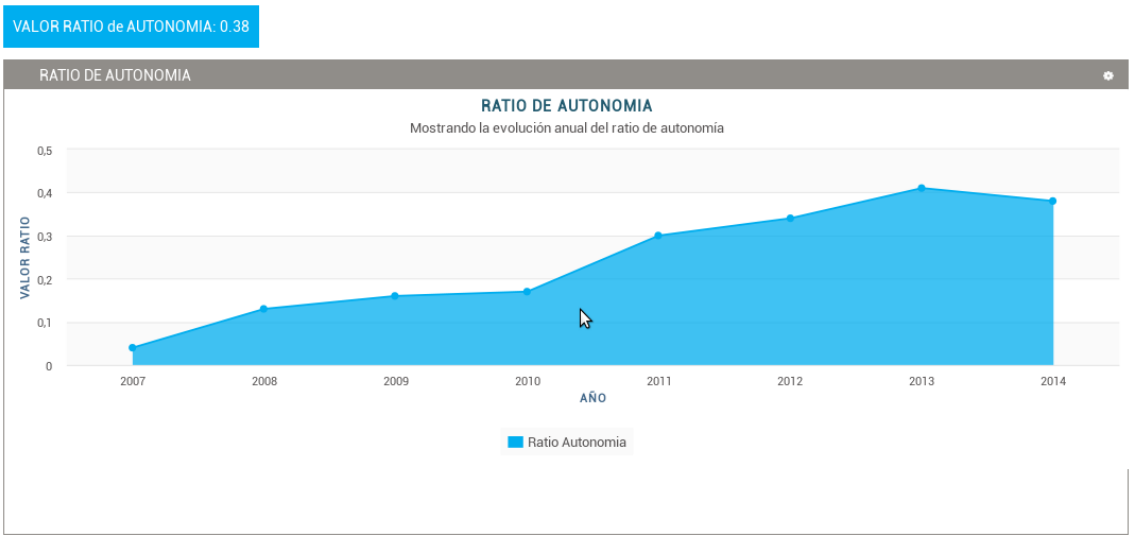
Muestra la evolución del ratio de autonomía a lo largo de un periodo de años elegido por el usuario. En la parte superior se muestra de forma numérica el valor del ratio de autonomía en el último año del periodo.

La fórmula utilizada para obtener el ratio de autonomía es:

```
R. Autonomía = Patrimonio Neto/Total Activo
```

La información se representa en una gráfica:

1. La gráfica de área muestra la evolución del valor del ratio de autonomía a lo largo de los años. La etiqueta azul de la zona superior muestra el valor numérico del ratio en el año de la consulta o del último año indicado en el filtro.



Indicador 35: Ratio de autonomía

4.4.5 Ratio de endeudamiento

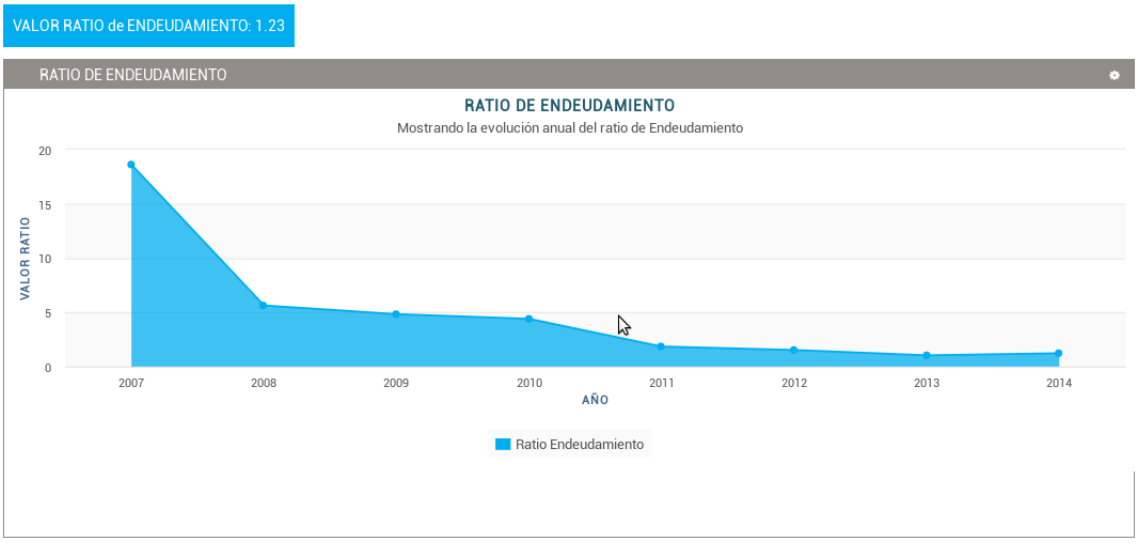
Muestra la evolución del ratio de endeudamiento a lo largo de un periodo de años elegido por el usuario. En la parte superior se muestra de forma numérica el valor del ratio de endeudamiento en el último año del periodo.

La fórmula utilizada para obtener el ratio de endeudamiento es:

```
R. Endeudamiento = Total Pasivo/Patrimonio neto
```

La información se representa en una gráfica:

1. La gráfica de área muestra la evolución del valor del ratio de endeudamiento a lo largo de los años. La etiqueta azul de la zona superior muestra el valor numérico del ratio en el año de la consulta o del último año indicado en el filtro.



Indicador 36: Ratio de endeudamiento

4.4.6 Ratio de rentabilidad de ventas

Muestra la evolución del ratio de rentabilidad de ventas a lo largo de un periodo de años y el mes elegido por el usuario. En la parte superior se muestra de forma numérica el valor del ratio de rentabilidad de ventas en el último año del periodo.

La fórmula utilizada para obtener el ratio de rentabilidad de ventas es:

```
R. Rent. Ventas = Resultado neto/Ventas
```

La información se representa en una gráfica:

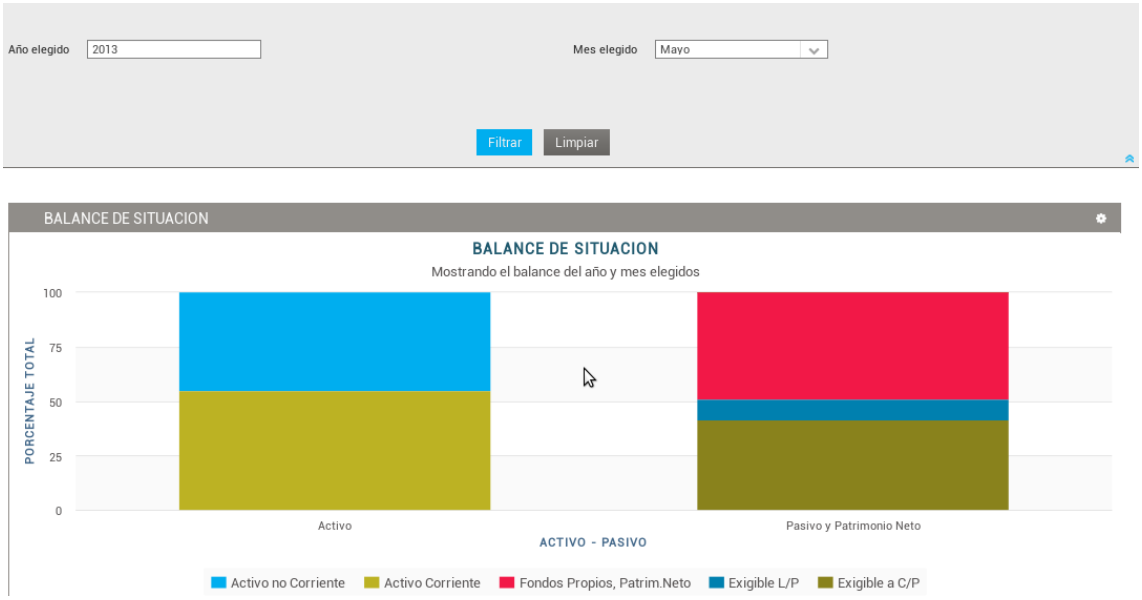
1. La gráfica de área muestra la evolución del valor del ratio de rentabilidad de ventas a lo largo de los años. La etiqueta azul de la zona superior muestra el valor numérico del ratio en el año de la consulta o del último año indicado en el filtro.

Indicador 37: Ratio de rentabilidad de ventas

4.4.7 Balance de situación

Muestra el balance de situación del mes y año elegidos. De esta forma el usuario puede realizar comparaciones de balance mes a mes.

La representación gráfica utilizada es la misma que la utilizada en el indicador “Resumen anual de ventas y gastos”.



Indicador 38: Balance de situación

4.5 Particularidades y decisiones de diseño

Se ha intentado mantener un diseño uniforme a la hora de representar gráficamente la información. Tras reunirnos la sección de diseño, consultoría y programación se llegó a conclusión de que era mejor priorizar el uso de gráficas similares que facilitaran la comprensión a usar distintos tipos de gráfica en todos los indicadores.

De esta forma, todos los ratios tienen una representación parecida, al igual que las gráficas que muestran la evolución de porcentajes en un intervalo de tiempo o las que muestran el balance de situación.

Respecto a los filtros, se ha intentado utilizar etiquetas de nombrado lo más simples y auto descriptivas posible. También se intenta facilitar la vida del usuario limitando a 4 la cantidad de dígitos que se pueden introducir en los cajetines de introducción de años o dando a elegir valores predefinidos a campos como las series de facturas o los meses del año.

5. Arquitectura de la capa de BI

5.1 Vista de módulos

La vista de módulos (con un alto nivel de abstracción) del sistema desarrollado es la siguiente:

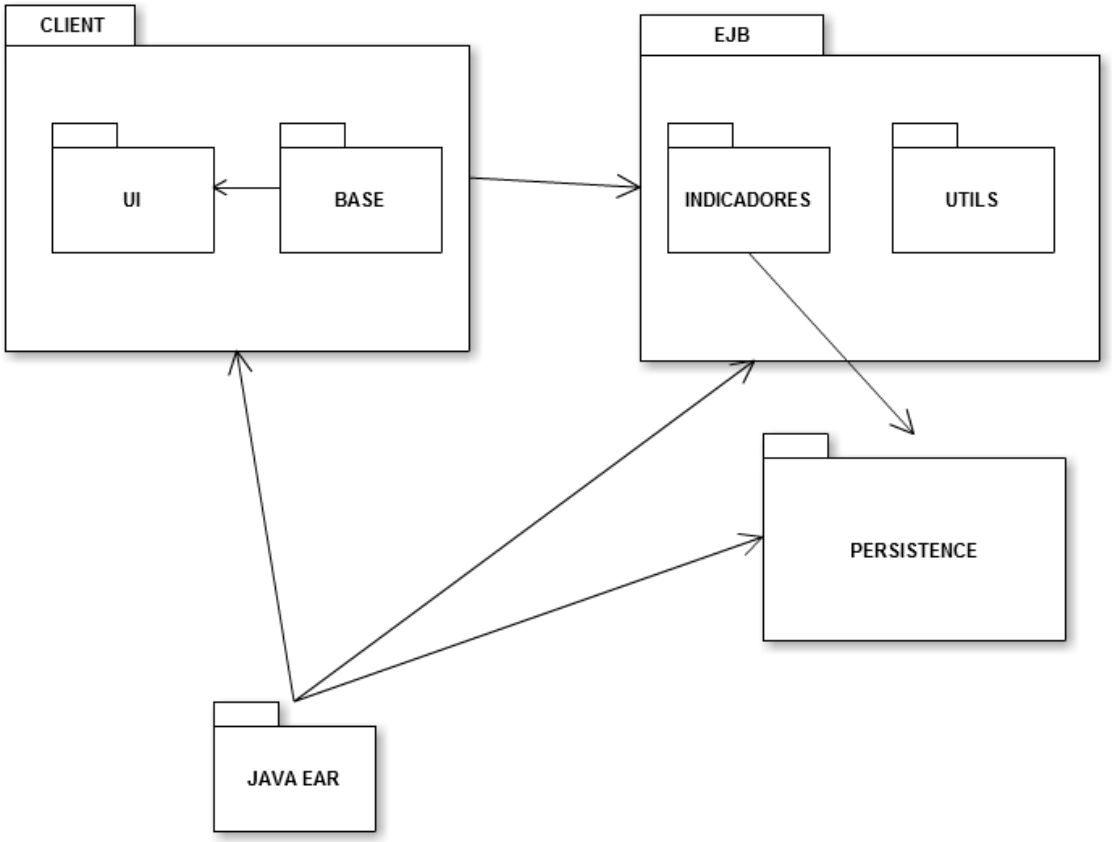


Figure 2: Vista de módulos

Se puede apreciar que el sistema cuenta con tres grandes módulos:

- Client: Contiene la vista del sistema. En las clases de este módulo se elige la repesentación que tendrá la información enviada por el controlador, además de los colores, fuentes etc. El framework “Vaadin” se encarga de traducir el código a “JavaScript” (para más información respecto al framework “Vaadin” se recomienda la lectura del Anexo VI). Contiene dos submódulos a tener en cuenta:
  - Base: Con el código base para representar la información.
  - UI: Con las clases básicas que marcan el estilo del sistema.
- EJB: Contiene la lógica de la aplicación, extraer, trata y gestiona los datos de la base de datos y envía la información a la vista para que se represente por pantalla. Entre sus submódulos destacan:
  - Utils: Cuenta con clases útiles para el resto de módulos.
  - Indicadores: Contiene la lógica de los indicadores implementados.

- Persistence: Contiene las clases necesarias para trabajar con la base de datos. Cada paquete cuenta con una clase que extrae la información de una tabla. La conexión con la base de datos Oracle se implementó con el uso del driver JDBC.

Además hay un cuarto módulo que correspondo al paquete de JavaEAR (Java Enterprise Archive) usado por JavaEE para encapsular varios módulos independientes en un solo archivo. Este archivo se sube al servidor elegido, en este caso JBoss. La configuración y el deploy del sistema ya estaban hechas cuando me incorporé al equipo de desarrollo, así que sólo tuve que hacer pequeñas modificaciones.

Para más detalles sobre el código fuente del proyecto, se recomienda la revisión de los Anexos III, IV y V.

5.2 Diagrama de despliegue

El diagrama de despliegue del sistema desarrollado es el siguiente:

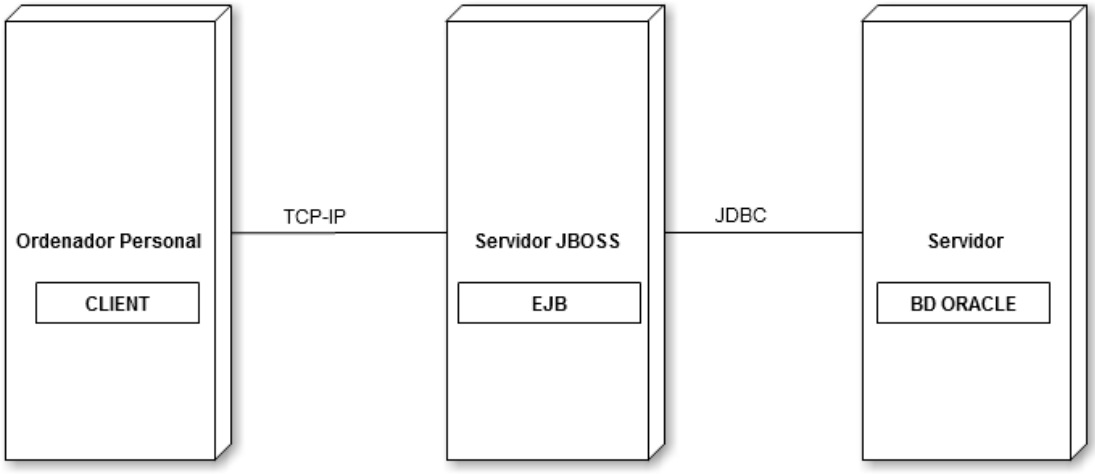


Figure 3: Vista de despliegue

Se pueden apreciar que el despliegue del sistema se divide en tres nodos de ejecución diferentes:

- Navegador del cliente: Se ejecuta la vista del sistema. El código Java se transforma en Javascript mediante el uso del framework “Vaadin” y se ejecuta en el navegador del cliente.
- Servido: JBoss está ejecutándose en un servidor virtual. En él se despliega el archivo “.ear” generado por JavaEE. Aquí se encuentra toda la lógica del sistema. Se extraen los datos, se tratan y se ponen a punto para su representación.
- Base de datos: La base de datos Oracle contiene cientos de tablas con información que se extrae para mostrarla en los indicadores. La conexión se implementó con el uso del driver JDBC.

5.3 Diagrama de secuencia

La interacción y paso de mensajes entre los componentes del sistema es la siguiente:

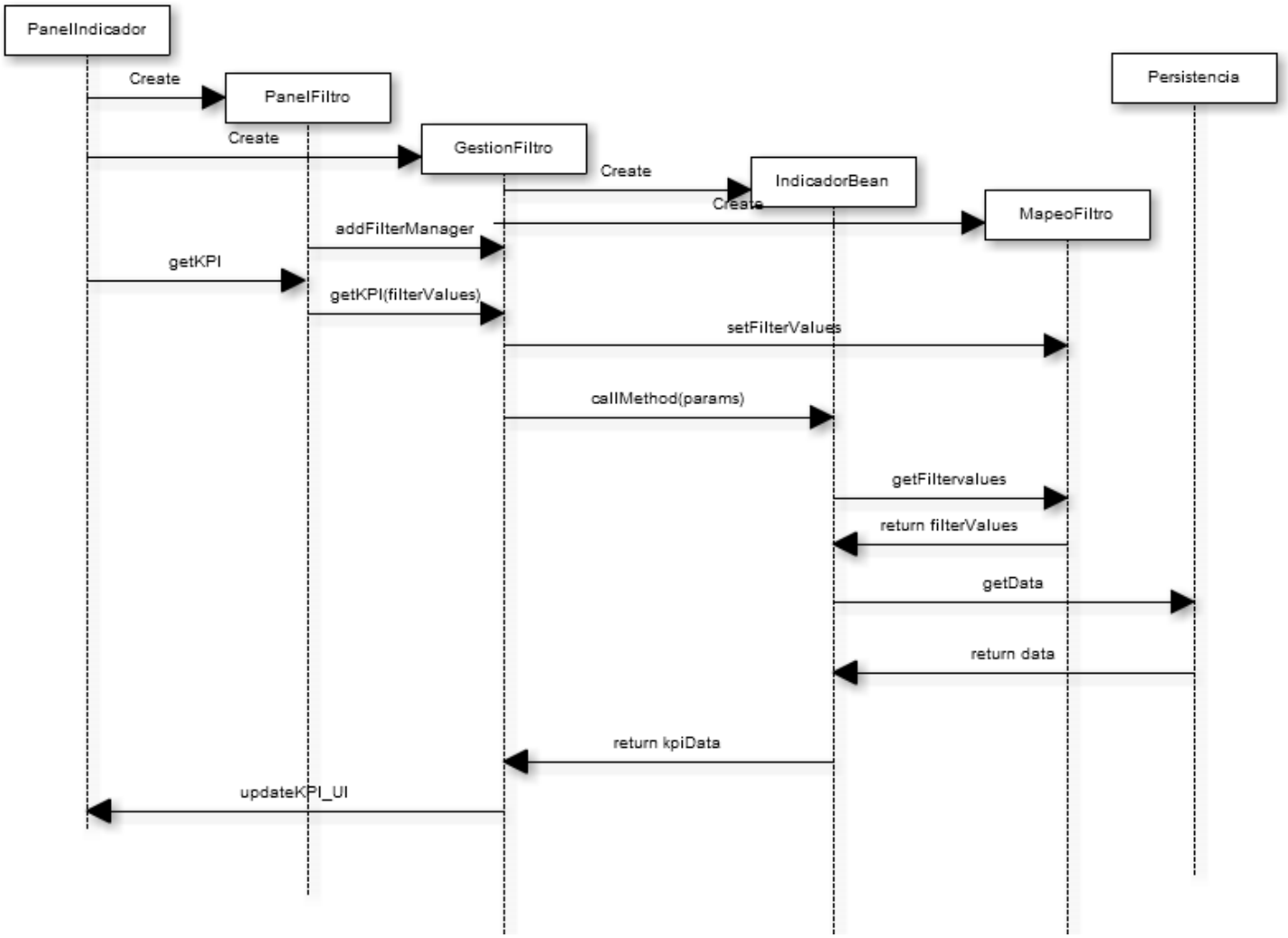


Figure 4: Diagrama de secuencia

Se puede apreciar la interacción entre las distintas capas del sistema:

- La vista está formada por las clases PanelIndicador y PanelFiltro.
- El controlador está formado por las clases GestionFiltro e IndicadorBean.
- El modelo está formado por la clase Persistencia y la base de datos.
- La clase MapeoFiltro es auxiliar y su función es guardar los datos introducidos por el usuario en el filtro.

Generar requiere de una serie de acciones y paso de mensajes que se dividen en tres fases:

Primero, los objetos de las clases PanelFiltro y GestionFiltro se crear una vez seleccionamos un indicador, cuando el usuario pulsa el botón “Filtrar” en la interfaz gráfica la clase GestionFiltro crea inmediatamente un objeto IndicadorBean y MapeoFiltro.

Segundo, se debe crear una conexión entre GestionFiltro y el “Bean” del indicador que se esté utilizando. El link entre GestionFiltro e IndicadorBean está gestionado por la configuración del sistema junto al servidor JBoss, de esta forma se ejecuta la comunicación “Vista-Controlador”. El objeto IndicadorBean ejecutará el método indicado por GestorFiltro y usará los datos obtenidos desde el filtro y que se han guardado en un objeto de la clase MapeoFiltro.

Por último, una vez efectuadas las consultas pertinentes sobre la base de datos, IndicadorBean devuelve los resultados a GestiónFiltro, que se encargará de avisar a PanelIndicador de que los datos se han actualizado y que se debe volver a generar la representación gráfica.

En el Anexo V se explica con más detalle esta interacción, usando ejemplos del código.

6. Gestión del proyecto

6.1 Metodología

La metodología de trabajo ha seguido la filosofía de las metodologías “ágiles”. He formado parte del equipo de desarrollo del EON.BS, de esta forma he contado con la ayuda y soporte de otros programadores y de la guía de consultores; además la dirección del consultor a cargo de la supervisión del proyecto ha sido frecuente.

El desarrollo del proyecto se planificó en base a las categorías de indicadores implementados, dedicando una o dos iteraciones para cada categoría, dependiendo de la complejidad de los indicadores implementados o del número de éstos.

6.2 Herramientas y tecnologías

Las herramientas y tecnologías empleadas en el desarrollo de este proyecto son las mismas que utiliza el equipo de desarrollo del EON.BS. Además, debido a la naturaleza del proyecto, ha sido necesario añadir tecnologías como JavaEE junto con las librerías y frameworks necesarios para desarrollar la capa de BI.

El listado de las tecnologías y herramientas utilizadas es:

- Debian como Sistema Operativo.
- Eclipse como IDE de desarrollo.
- JBoss como servidor de despliegue de aplicaciones.
- JavaEE: Tecnología Java orientada a grandes aplicaciones empresariales.
- Vaadin como framework de representación gráfica de datos. Su función es traducir el código Java a JavaScript, usando ficheros CSS previamente definidos. De esta forma se genera la interfaz web de la aplicación.
- SQLDeveloper para la obtención, inserción y edición de datos.
- EclipseLink + Criteria para el desarrollo de la capa genérico de acceso a la base de datos. El uso de estas tecnologías, junto con el intérprete de Hibernate, permiten crear una capa de acceso a BD completamente genérica, de forma que no es necesario modificarla al cambiar de SGBD.
- Email y Skype como herramienta de comunicación con los miembros del equipo.

6.3 Planificación

El proyecto ha contado con 10 fases:

I. Estudio de la documentación del proyecto e integración en el entorno de trabajo.
II. Estudio de la estructura de la base de datos y de las tablas más relevantes.
III. Implementación de los primeros dos indicadores.
IV. Elección y especificación de las categorías del resto de indicadores.
V. Elección e implementación de los indicadores de la categoría clientes.
VI. Elección e implementación de los indicadores de la categoría aprendizaje.
VII. Elección e implementación de los indicadores de la categoría financiera.
VIII. Elección e implementación de los indicadores de la categoría interna.
IX. Testeo y mejoras en la interfaz de usuario.
X. Presentación del proyecto

El diagrama de Gantt del proyecto puede verse en el Anexo II.

7. Conclusiones y trabajo futuro

Como conclusión, es importante resaltar que todos los objetivos que se habían planteado para este trabajo se han cumplido satisfactoriamente:

- El primero de los objetivos que se había establecido era la **integración** en el equipo de desarrollo del ERP EON.BS y estudiar tanto la arquitectura del sistema a desarrollar como el entramado de tablas de la Base de Datos. Tras asistir prácticamente de forma diaria a la oficina, de preguntar a otros **desarrolladores** dudas técnicas sobre la arquitectura y aclarar la estructura de tablas con la ayuda de los **consultores**, este objetivo se ha cumplido completamente.

- Otro de los objetivos de este trabajo era definir un conjunto de categorías de KPIs trabajando en **equipo** con el consultor encargado del proyecto. Esta experiencia me ha servido para comprender mejor las **necesidades** reales que puede tener un cliente.

- Ha sido necesario aprender diferentes términos **financieros y empresariales** con los que nunca he trabajo o que he visto muy por encima a lo largo de la carrera: Cómo funciona un almacén con el uso de albaranes de venta/compra/transporte, elementos de una **cadena de suministro**, importancia del **balance de situación**, el meticuloso control de finanzas con el que cuentan las empresas, la importancia de un **CRM** capaz de encontrar oportunidades comerciales, etc.

- Como se ha mencionado a lo largo del documento, se han definido **4 categorías** diferentes de KPIs: **Interna, Clientes, Aprendizaje y Financiera**, contando cada categoría con un número determinado de KPIs que servirán para satisfacer las necesidades más **genéricas** de los clientes, siendo 20 (veinte) el total de KPIs implementados.

- El último objetivo era el **desarrollo** de un sistema de **Business Intelligence** para el ERP EON.BS. Para ello se han utilizado las herramientas de **desarrollo** proporcionadas por el director de proyectos de la empresa y se ha seguido la guía de estilo de código indicada por el mismo. Además se ha trabajado junto con la **diseñadora** gráfica de la oficina para el desarrollo de la interfaz de usuario, de esta forma se ha conseguido desarrollar una **interfaz amigable** para el usuario que cuenta con un correcto uso de los espacios en blanco y el contraste de colores. El desarrollo de este sistema se ha hecho en **Java**, lenguaje en el que está desarrollado todo el ERP, y se han utilizado herramientas empresariales como **JBoss o JavaEE** para integrar todos los componentes que se han programado.

Para acabar la conclusión me gustaría exponer los puntos en los que creo que el sistema podría **mejorarse**:

- Al tratarse de una capa de BI con KPIs genéricos, es altamente probable que haya que **añadir KPIs elegidos** de forma expresa por el cliente. Gracias a la arquitectura del sistema y a su **diseño por capas**, añadir nuevos KPIs no es una tarea ni mucho menos dramática, de hecho es bastante sencillo e intuitivo.

- Otro apartado a mejorar puede ser el añadir **comportamiento más dinámico** a los KPIs, de forma que al pulsar en la gráfica de un año se mostrara información detallada sobre, por ejemplo, el balance de ese año.

- Para terminar, el filtro de selección de clientes puede mostrar un comportamiento errático o confuso para el usuario inexperto, sería recomendable añadir expresiones regulares a la búsqueda para facilitar la tarea.

8. Bibliografía

Historia ERP:
http://www.xperimentos.com/2007/05/23/historia-del-surgimiento-de-los-sistemas-erp/

Definición ERP:
https://en.wikipedia.org/wiki/Enterprise_resource_planning

Definición BI:
http://data-warehouses.net/guide/introduction.html
http://searchdatamanagement.techtarget.com/definition/business-intelligence
Nota: La información se ha extraído del trabajo obligatorio (TP6) realizado para la asignatura de “Almacenes y Minería de Datos”.

Definición de KPI:
http://management.about.com/cs/generalmanagement/a/keyperfindic.htm
https://es.wikipedia.org/wiki/KPI

Anexo I: Uso de los indicadores

Todos los indicadores siguen el mismo patrón de uso:

- Los datos mostrados por defecto son:
    - Si se trata de una gráfica evolutiva muestra el año actual y 7 años en el pasado. En caso de mostrar datos mensuales, muestra los datos del mes actual.
    - Si se trata de una gráfica que muestra un solo año, muestra el año y mes actuales.
    - Si se trata de una tabla, aparecerá sin datos hasta que se elija un cliente. En este caso, los meses y años que aparecerán seleccionados por defecto en el filtro serán los meses y años actuales y los inmediatamente anteriores.
- Para cambiar los valores de un filtro:
    - En caso de tratarse de una tabla, deberá elegirse el cliente pulsando en el icono “+” y filtrando los clientes de la empresa introduciendo el deseado en el cuadro de texto que aparecerá. Después se debe pulsar el botón filtrar.
    - En el resto de indicadores basta con rellenar los campos y pulsar el botón filtrar.
- Tras ejecutar un indicador:
    - Los valores obtenidos no se borran al cambiar de indicador, se quedan guardados en la memoria del navegador para que no haya que volver a hacer la consulta otra vez.
    - El filtro se guarda para que los datos ocupen toda la pantalla. Para volver a mostrar el filtro hay que pulsar sobre el icono “^” que aparece en la esquina superior derecha de la pantalla.

Todas las gráficas muestran los datos que representar al dejar el puntero del ratón encima del valor que queramos ver. Además, si pulsamos sobre la leyenda de una serie, la representación gráfica desaparecerá hasta que se vuelva a pulsar en la leyenda, es decir:

- Se muestran valores del año 2012 y del año 2013.
- Queremos ver sólo los datos del año 2013.
- Pulsamos sobre la leyenda del año 2012 para que desaparezca dicho año de la gráfica.
- Pulsamos otra vez para que vuelva a aparecer.

Anexo II: Diagrama de Gantt del proyecto

A continuación se muestra el diagrama de Gantt correspondiente a las fases del proyecto mostradas en el punto 6.3 de este documento.

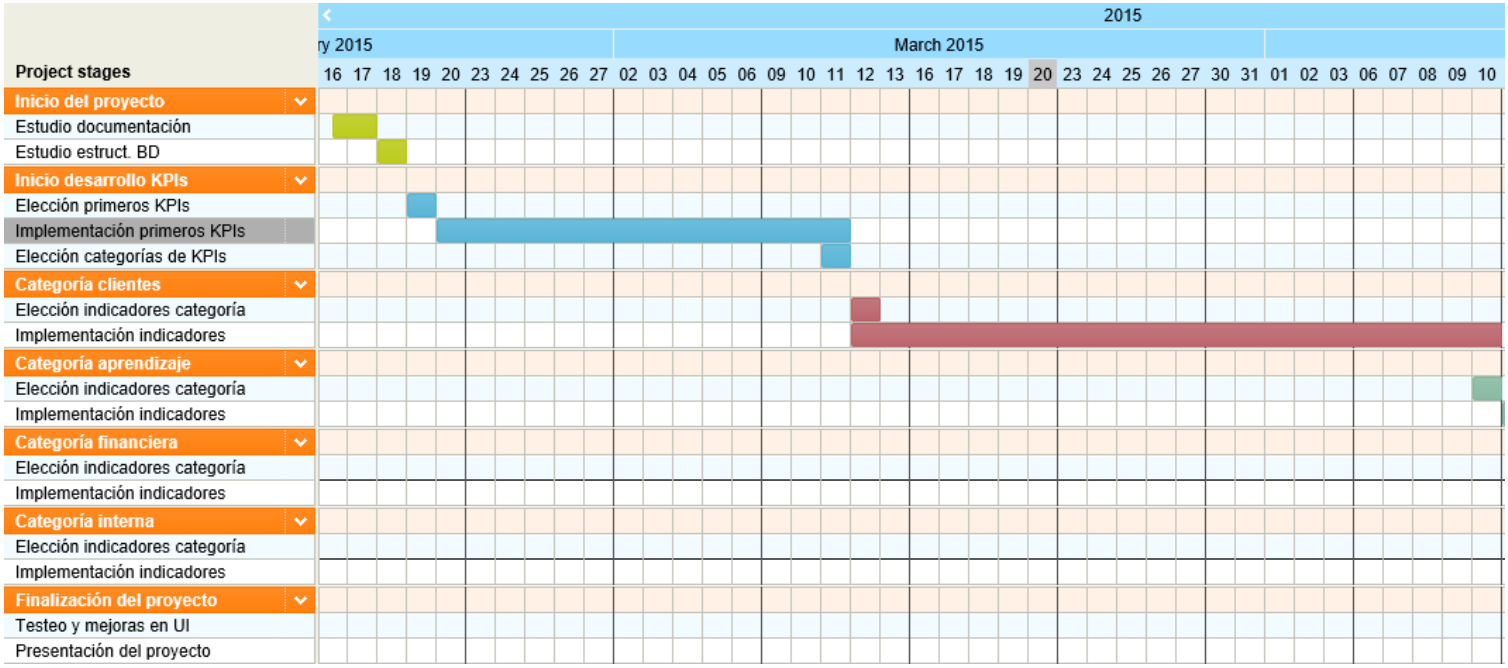


Figure 5: Diagrama de Gantt pt.1

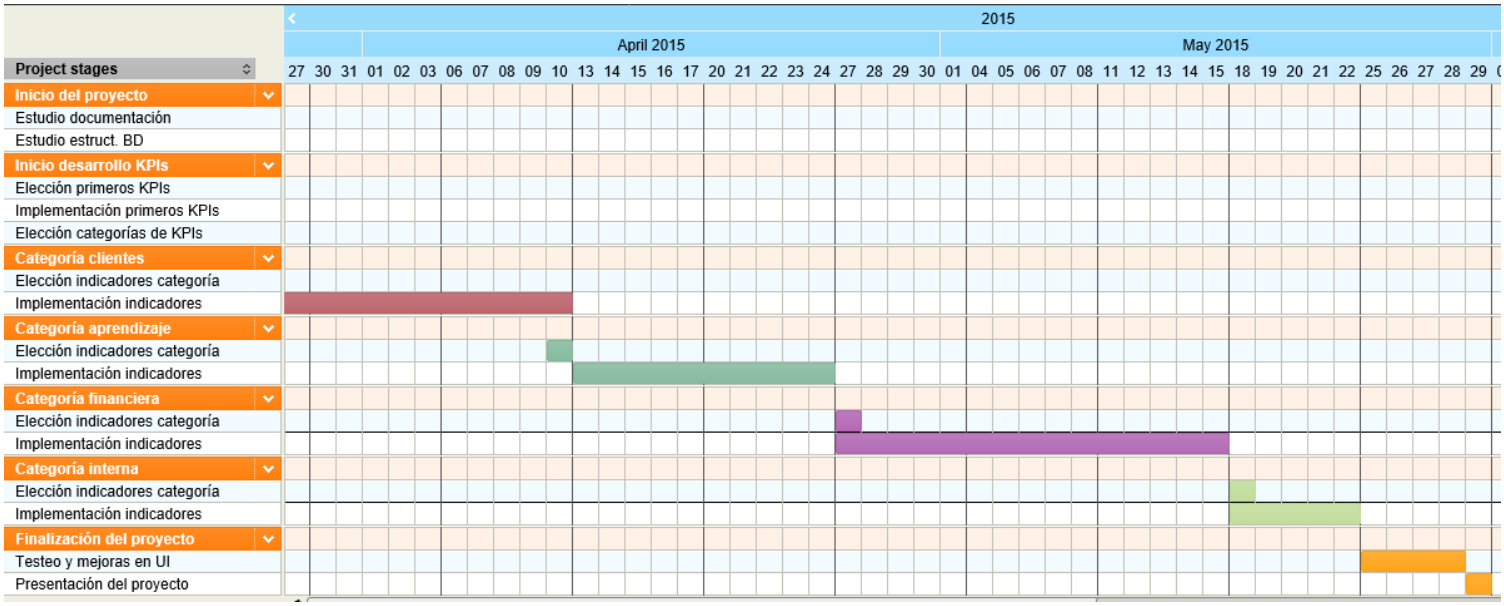


Figure 6: Diagrama de Gantt pt.2

Anexo III: Código fuente - Consultas con Criteria

La conexión con la base de datos Oracle se implementó con el uso del driver JDBC, el cual es ampliamente conocido y utilizado. Sin embargo, el enfoque que se ha dado a la interacción con la base de datos ha sido completamente diferente a lo que podría llamarse “habitual” al desarrollar un programa con Java y Oracle.

El enfoque dado a la interacción con la base de datos ha sido completamente orientado a objetos. Es decir, en ningún momento se han utilizado queries SQL relacionales ni se han tratado a las tablas como tales.

Para conseguir esto, se ha utilizado la tecnología “Eclipse Link”, que permite añadir una capa genérica de interacción con la base de datos. De esta forma, si en un futuro se decidiera cambiar la BD Oracle por una SQL Server, tan sólo habría que cambiar un par de líneas en el fichero de configuración.

En resumen, las interacciones con la BD se hacen utilizando las clases y métodos proporcionados por Eclipse Link y la librería “Persistence”. Los pasos a seguir son:

- Convertir una tabla de la BD a una clase Java. Cada columna de la BD será un atributo de la clase.

```
@Entity
@Table(name="A_CABE_ALBARANES_COMPRAS")
@NamedQuery(name="CabeceraAlbaranCompraEntity.findAll", query="SELECT c FROM CabeceraAlbaranCompraEntity c")
public class CabeceraAlbaranCompraEntity implements Serializable {
   private static final long serialVersionUID = 1L;

   @Id
   @Column(nullable = false, precision = 10)
   private long codigo;

   @Column(name="BIC_PROVEEDOR", length = 11)
   private String bicProveedor;

   @Column(nullable = false, precision = 15)
   private BigDecimal cambio;
```

Code 1: Tabla BD a clase Java

- Gestión de las claves ajenas.

```
@OneToMany(mappedBy = "ACabeceraAlbaranCompra")
private List<LineasAlbaranesComprasEntity> ALineasAlbaranesCompras;
```

Code 2: OneToMany

```
@ManyToOne
@JoinColumn(name = "FK_CABECERA_ALBARAN_COMPRA", nullable = false)
private CabeceraAlbaranCompraEntity ACabeceraAlbaranCompra;
```

Code 3: ManyToOne

- Crear un objeto “Criteria” al que se le enlaza una clase/tabla de la BD.

```
CriteriaBuilder cb = em.getCriteriaBuilder();
CriteriaQuery<Tuple> criteria = cb.createTupleQuery();
```

Code 4: Create criteria

```
Root<CabezaFacturaVentaEntity> cabFactRoot = criteria.from(CabezaFacturaVentaEntity.class);
```

Code 5: Criteria from

- Aplicar métodos sobre el objeto “Criteria”.

  - Elección de tabla principal y JOINS con otras tablas

```
Root<CabezaFacturaVentaEntity> cabFactRoot = criteria.from(CabezaFacturaVentaEntity.class);
Join<CabezaFacturaVentaEntity, DireccionFacturacionVentasEntity> joinCabDir = cabFactRoot.join(CabezaFacturaVentaEntity_.ADireccionFacturacion);
Join<DireccionFacturacionVentasEntity, ClientesEntity> joinDirCliente = joinCabDir.join(DireccionFacturacionVentasEntity_.ACliente);
Join<CabezaFacturaVentaEntity, FacturaVentaPorcentajeEntity> joinCabFactPor = cabFactRoot.join(CabezaFacturaVentaEntity_.APieFactura);
Join<CabezaFacturaVentaEntity, FacturacionVentasEntity> joinCabFactFact = cabFactRoot.join(CabezaFacturaVentaEntity_.AFacturacionVentas);
Join<FacturacionVentasEntity, CabeceraAlbaranVentaEntity> joinFactCabAlbaran = joinCabFactFact.join(FacturacionVentasEntity_.ACabeceraAlbaranesVenta);
Join<CabeceraAlbaranVentaEntity, LineasAlbaranesVentaEntity> joinCabAlbLineas = joinFactCabAlbaran.join(CabeceraAlbaranVentaEntity_.ALineasAlbaranesVentas);
Join<LineasAlbaranesVentaEntity, ArticuloEntity> joinLinArt = joinCabAlbLineas.join(LineasAlbaranesVentaEntity_.AArticulo, JoinType.LEFT);
```

Code 6: Root table y Joins

  - Restricciones

```
//desde principio de anio
Calendar calendar1 = UtilFecha.firstDayYear(anyoString);
restricciones.add(cb.greaterThanOrEqualTo(cabFactRoot.get(CabezaFacturaVentaEntity_.fechaFactura), calendar1));

//hasta final de anio
Calendar calendar2 = UtilFecha.firstDayYear(anyoString + 1);
restricciones.add(cb.lessThan(cabFactRoot.get(CabezaFacturaVentaEntity_.fechaFactura), calendar2));

restricciones.add(cb.equal(joinCabFactPor.get(FacturaVentaPorcentajeEntity_.ordenPorcentajeImporte), MapeoFacturaVentaPorcentaje.ORDEN_PORCENTAJE_IMPORTE_ZERO)); //0
restricciones.add(cb.equal(joinCabFactPor.get(FacturaVentaPorcentajeEntity_.porcentajeImporte), MapeoFacturaVentaPorcentaje.PORCENTAJE_IMPORTE_V)); //V
restricciones.add(cb.equal(joinCabFactPor.get(FacturaVentaPorcentajeEntity_.fkIdenContabilidad), mapeoSesionRecibo.getIdentificadorContabilidad()));
```

Code 7: Restricciones

```
if (restricciones.size() > 0)
  criteria.where(cb.and(restricciones.toArray(new Predicate[restricciones.size()])));
```

Code 8: Ejecución restricciones

- Operaciones de agregación

```
criteria.groupBy(joinCabDir.get(DireccionFacturacionVentasEntity_.nombreComercial));
criteria.orderBy(cb.desc(cb.sum(joinCabFactPor.get(FacturaVentaPorcentajeEntity_.base))));
```

Code 9: Operaciones de agregación

- Ejecutar el select

```
criteria.multiselect(joinCabDir.get(DireccionFacturacionVentasEntity_.nombreComercial).alias("cliente"),
    cb.sum(joinCabFactPor.get(FacturaVentaPorcentajeEntity_.base)).alias("total")).distinct(true);
```

Code 10: Ejecutar select

Un intérprete se encargará de traducir el código Java a una query SQL del SGBD que hayamos elegido. Esta capa genérica tiene un precio: Las operaciones con la BD son menos eficientes, ya que el intérprete debe traducir el código Java, además no se pueden utilizar operaciones o características especiales de cada SGBD, por ejemplo, en Oracle no podría realizarse una operación “ROLLUP”.

Anexo IV: Código fuente - Consultas indicador “Ventas anuales”

El indicador “Ventas anuales” cuenta con una complejidad superior a la media, por esta razón es el ejemplo ideal para mostrar las consultas realizadas a la Base de Datos.

A diferencia del Anexo III, en esta sección no se explicará en detalle el uso de “Criteria” para ejecutar las queries, sino la lógica de éstas y los pasos necesarios para obtener los datos mostrados en las tablas.

La asignación del “Root” y los “Joins” necesarios son 7 en total. Es decir, se usan 8 tablas diferentes:

```
Root<CabezaFacturaVentaEntity> cabFactRoot = criteria.from(CabezaFacturaVentaEntity.class);
Join<CabezaFacturaVentaEntity, DireccionFacturacionVentasEntity> joinCabDir = cabFactRoot.join(CabezaFacturaVentaEntity_.ADireccionFacturacion);
Join<DireccionFacturacionVentasEntity, ClientesEntity> joinDirCliente = joinCabDir.join(DireccionFacturacionVentasEntity_.ACliente);
Join<CabezaFacturaVentaEntity, FacturaVentaPorcentajeEntity> joinCabFactPor = cabFactRoot.join(CabezaFacturaVentaEntity_.APieFactura);
Join<CabezaFacturaVentaEntity, FacturacionVentasEntity> joinCabFactFact = cabFactRoot.join(CabezaFacturaVentaEntity_.AFacturacionVentas);
Join<FacturacionVentasEntity, CabeceraAlbaranVentaEntity> joinFactCabAlbaran = joinCabFactFact.join(FacturacionVentasEntity_.ACabeceraAlbaranesVenta);
Join<CabeceraAlbaranVentaEntity, LineasAlbaranesVentaEntity> joinCabAlbLineas = joinFactCabAlbaran.join(CabeceraAlbaranVentaEntity_.ALineasAlbaranesVentas);
Join<LineasAlbaranesVentaEntity, ArticuloEntity> joinLinArt = joinCabAlbLineas.join(LineasAlbaranesVentaEntity_.AArticulo, JoinType.LEFT);
```

Code 11: Root y Joins de la consulta del indicador "Ventas anuales"

Cabe destacar la última línea: Se trata de un “Join LEFT”, es decir, hará join de los datos que existan en ambas tablas y si algún dato no existe en la tabla derecha, dichos datos persistirán en la tabla final con un “null” asociado.

La lista de restricciones también es bastante amplia:

```
//Iden. Empresa
restricciones.add(cb.equal(joinCabFactPor.get(FacturaVentaPorcentajeEntity_.ordenPorcentajeImporte), MapeoFacturaVentaPorcentaje.ORDEN_PORCENTAJE_IMPORTE_ZERO)); //0
restricciones.add(cb.equal(joinCabFactPor.get(FacturaVentaPorcentajeEntity_.porcentajeImporte), MapeoFacturaVentaPorcentaje.PORCENTAJE_IMPORTE_V)); //V
restricciones.add(cb.equal(joinCabFactPor.get(FacturaVentaPorcentajeEntity_.fkIdenContabilidad), mapeoSesionRecibo.getIdentificadorContabilidad()));

//Para el anyo filtrado
Calendar calendar1 = UtilFecha.firstDayYear(mapeoFiltroRecibo.getEntradaAnyoHasta());
restricciones.add(cb.greaterThanOrEqualTo(cabFactRoot.get(CabezaFacturaVentaEntity_.fechaFactura), calendar1));

Calendar calendar2 = UtilFecha.firstDayYear(mapeoFiltroRecibo.getEntradaAnyoHasta() + 1);
restricciones.add(cb.lessThan(cabFactRoot.get(CabezaFacturaVentaEntity_.fechaFactura), calendar2));

//Para los clientes filtrados
if (!mapeoFiltroRecibo.getCodigosClientes().isEmpty())
  restricciones.add(joinDirCliente.get(ClientesEntity_.codigo).in(mapeoFiltroRecibo.getCodigosClientes()));

if (restricciones.size() > 0)
  criteria.where(cb.and(restricciones.toArray(new Predicate[restricciones.size()])));
```

Code 12: Restricciones de la consulta del indicador "Ventas anuales"

Además se debe de tener en cuenta el resultado obtenido en la consulta previa del filtro. Este indicador es el único que necesita acceder a la base de datos para filtrar contenidos.

Por último, se ejecuta el select con el group by incluido:

```
criteria.multiselect(
    joinLinArt.get(ArticuloEntity_.articulo).alias("codigoArticulo"),
    joinCabAlbLineas.get(LineasAlbaranesVentaEntity_.codigoArticuloSinCodificar).alias("codigoManual"),
    joinCabAlbLineas.get(LineasAlbaranesVentaEntity_.descripcion).alias("producto"),
    cb.sum(joinCabAlbLineas.get(LineasAlbaranesVentaEntity_.importeDivisa)).alias("importe"));

criteria.groupBy(joinLinArt.get(ArticuloEntity_.articulo),
    joinCabAlbLineas.get(LineasAlbaranesVentaEntity_.codigoArticuloSinCodificar),
    joinCabAlbLineas.get(LineasAlbaranesVentaEntity_.descripcion));
```

Code 13: Select de la consulta del indicador "Ventas anuales"

El resultado se almacena en un “array” y los datos almacenados en dicho array pasan a tratarse. Se debe tener en cuenta que el array puede estar vacio, que puede contener resultados nulos o que la consulta ha sido errónea y no devuelve los datos deseados.

Para tratar estos casos se relega en métodos privados, manteniendo un diseño de código descendente:

```
if (result.size() != 0)
{
  MapeoIndicadorVentasAnuales mapeoIndicadorVentasAnualesAuxVarios = new MapeoIndicadorVentasAnuales();

  mapeoIndicadorVentasAnualesAuxVarios.setNombreImporte("PRODUCTOS VARIOS");
  mapeoIndicadorVentasAnualesAuxVarios.setCantidadImporte(BigDecimal.ZERO);
  mapeoIndicadorVentasAnualesAuxVarios.setCodigoArticulo("N/A");
  mapeoIndicadorVentasAnualesAuxVarios.setAnyo(mapeoFiltroRecibo.getEntradaAnyoHasta());

  mapAux.put(this.getKeyGrupo(mapeoFiltroRecibo.getEntradaAnyoHasta(), "PRODUCTOS VARIOS"), mapeoIndicadorVentasAnualesAuxVarios);

}

this.ordenarResultados(mapAux, mapDatosVentasAnuales);
```

Code 14: Diseño descendente pt.1

```java
/**
 * Dado un mapAux, mete sus datos en el map principal
 * @param mapAuxRecibo
 * @param mapDatosVentasAnualesRecibo
 */
private void ordenarResultados(Map<String, MapeoIndicadorVentasAnuales> mapAuxRecibo,
                               Map<Integer, MapeoIndicadorVentasAnuales> mapDatosVentasAnualesRecibo)
{
  //Obtener y ordenar los mapeos
  List<MapeoIndicadorVentasAnuales> listaOrdenada = new ArrayList<MapeoIndicadorVentasAnuales>(mapAuxRecibo.values());
  Collections.sort(listaOrdenada, (o1, o2) -> {
    if (o1.getPorcetanjeImporte().compareTo(o2.getPorcetanjeImporte()) < 0)
      return 1;
    else if(o1.getPorcetanjeImporte().compareTo(o2.getPorcetanjeImporte()) > 0)
      return -1;
    else
      return 0;
  });

  int contador = 0;
  for (MapeoIndicadorVentasAnuales mapeo : listaOrdenada)
  {
    mapDatosVentasAnualesRecibo.put(Integer.valueOf(contador++), mapeo);
  }
}
```

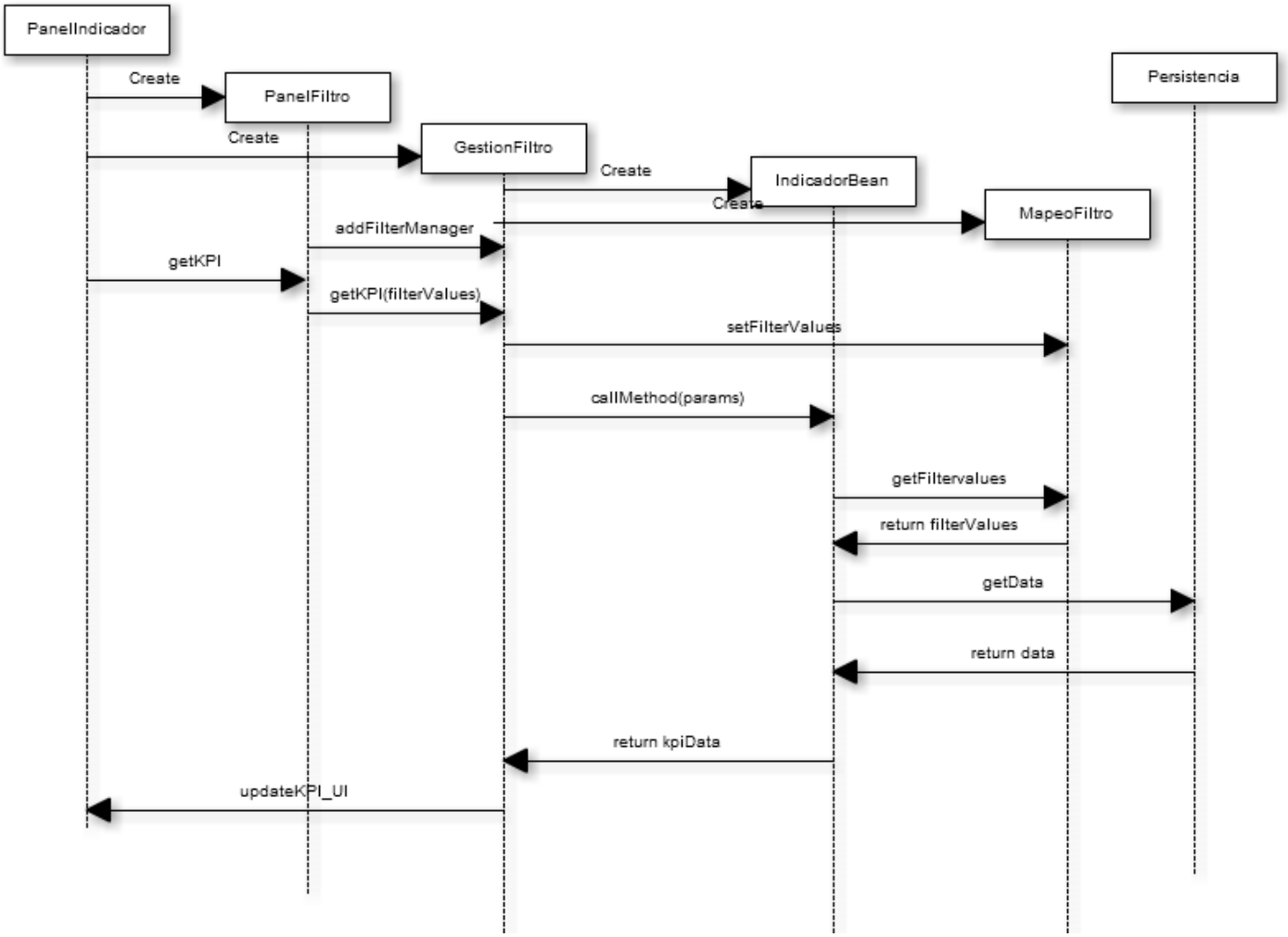
Code 15: Diseño descendente pt.2

Estos fragmentos de código tan solo muestran la gestión de las consultas a la base de datos y el tratamiento a los datos obtenidos para que sean útiles y fácilmente representables en la interfaz. Por supuesto, habría que sumar las partes de creación de la interfaz gráfica, la gestión del mapeo de las tablas de la BD, la implementación del filtro y la obtención de los clientes que en éste se muestran, etc…

Anexo V: Código fuente - Clases empleadas en un indicador

A lo largo del documento se ha ido explicando la arquitectura del sistema. Este Anexo tiene como propósito el mostrar las líneas de código en las que se muestra la interacción entre objetos descrita en el punto 5.3 y explicar de forma breve qué clases se utilizan, dónde se utilizan y por qué.

El diagrama de secuencia se trataba del siguiente:



Al seleccionar un KPI se crea un objeto “PanelIndicador”, el cual genera instantáneamente los objetos “PanelFiltro” y “GestorFiltro”.

```
public PanelIndicadorRatioLiquidez(AbstractEscritorio abstractEscritorioRecibido)
{
  super(abstractEscritorioRecibido);
  this.gestionEventosFiltroRatioLiquidez = new GestionEventosFiltroRatioLiquidez(this);
  this.init();
}

private void init()
{
  //Filtro
  this.panelFiltroRatioLiquidez = new PanelFiltroRatioLiquidez(this.gestionEventosFiltroRatioLiquidez, this);
  this.addFiltro(this.panelFiltroRatioLiquidez, 150);
```

Code 16: Creación de PanelFiltro y GestorFiltro

Obviamente, el objeto “PanelFiltro” se encargará de gestionar el comportamiento del filtro del indicador y de recoger los datos introducidos. “GestorFiltro” detectará cuando se pulsa el botón “Filtrar” y usará los datos recogidos por el filtro (tras comprobar que son correctos en formato) como parámetros de entrada al llamar a la clase indicada para obtener la información que se mostrará en ese indicador.

```
@Override
public void botonFiltrarPulsado()
{
  if (this.verificarCamposObligatorios())
  {
    this.rellenarMapeo();
    this.panelIndicadorRatioLiquidez.panelFiltroRatioLiquidez.plegar();

    this.updateLabelRatioLiquidez();
    this.updateChartRatioLiquidez();
  }
}
```

Code 17: Compropación de campos pt.1

```
private boolean verificarCamposObligatorios()
{
  boolean ok = true;

  if (this.panelIndicadorRatioLiquidez.panelFiltroRatioLiquidez.entradaNumeroAnyoDesde.getValueInteger() == null)
  {
    this.panelIndicadorRatioLiquidez.panelFiltroRatioLiquidez.entradaNumeroAnyoDesde.setComponentError(new UserError("El campo desde es obligatorio"));
    ok = false;
  }
  else
    this.panelIndicadorRatioLiquidez.panelFiltroRatioLiquidez.entradaNumeroAnyoDesde.setComponentError(null);
```

Code 18: Comprobación de campos pt.2

Para que los datos del filtro sean más fáciles de utilizar (puesto que puede haber múltiples datos introducidos por el usuario), se utiliza una clase “mapeo” llamada “MapeoFiltro”. El objeto creado de esta clase será el que se pase como parámetro de entrada:

```
private void rellenarMapeo()
{
  this.mapeoFiltroRatioLiquidez.setAnyoDesde(this.panelIndicadorRatioLiquidez.panelFiltroRatioLiquidez.entradaNumeroAnyoDesde.getValueInteger());
  this.mapeoFiltroRatioLiquidez.setAnyoHasta(this.panelIndicadorRatioLiquidez.panelFiltroRatioLiquidez.entradaNumeroAnyoHasta.getValueInteger());
}
```

Code 19: Mapeo de campos

La llamada a la clase que devolverá la información del indicador se hace llamando a su interfaz, de esta forma sólo la clase que implemente dicha interfaz será la que reciba la llamada. Por otra parte, es el servidor JBoss junto con la configuración del sistema la que mapea en tiempo de ejecución la vista con el controlador.

```
private void crearConexionEJB()
{
  try
  {
    this.indicadorRatioLiquidezBeanRemote = EJBLocator.getInstance().
    <IndicadorRatioLiquidezBeanRemote> locateEJB(IndicadorRatioLiquidezBeanRemote.JNDI);
  }
  catch (NamingException e)
  {
    ExcepcionDelegate.getInstance().controlarExcepciones(e);
  }
}
```

Code 20: Mapeo Vista-Controlador

```
private void updateLabelRatioLiquidez()
{
  try
  {
    BigDecimal ratioLiquidez = BigDecimal.ZERO;
    ratioLiquidez = indicadorRatioLiquidezBeanRemote.
    obtenerRatioLiquidez(this.panelIndicadorRatioLiquidez.panelFiltroRatioLiquidez.
    entradaNumeroAnyoHasta.getValueInteger(), AbstractUi.getCurrent().getMapeoSesion());
```

Code 21: Llamada a interfaz del indicador

Una vez recogidos los datos tratados devueltos por la respuesta, se actualiza el KPI con la nueva información.

```
panelIndicadorRatioLiquidez.getTextoVentasLabel().setValue("VALOR RATIO de LIQUIDEZ: " + ratioLiquidez);
```

Code 22: Actualización de la UI del indicador

Todos los KPIs del sistema funcionan de forma idéntica a la aquí resumida. Incluso aquellos con un filtro más complejo mantienen la misma secuencia de interacción.

Anexo VI: Vaadin Framework

Para el desarrollo de la UI de los indicadores se ha utilizado Vaadin. En su propia web oficial, Vaadin se define como:

Vaadin Framework te permite construir webapps en sistemas que cuenten con back-end Java u otro lenguaje que utilice la JVM. Vaadin se encarga de automatizar la comunicación front-end <-> back-end y mantiene el estado de la webapp en el server, usando HTML5 para la representación gráfica en los navegadores.

Es decir: Vaadin es un framework que transforma código Java en JavaScript, permitiendo la visualización de los indicadores mediante interfaz web y sin necesidad de instalar ningún programa en el cliente: El navegador de internet será lo único que se necesite. Además, automatiza la comunicación entre navegador y servidor, facilitando la tarea de los programadores.

A la hora de programar utilizando Vaadin hay que tener en cuenta dos aspectos que, si bien pueden parecer intrínsecamente unidos en programación Java, se gestionan de forma totalmente separada en programación Web: El contenido y el estilo. Aunque Vaadin presume de su capacidad para transformar el código Java en JavaScript, necesita una “pequeña ayuda” por parte del programador. Por esta razón, aunque se programe en Java es necesario programar el contenido y el estilo de forma separada:

Programación del contenido del indicador

Como se ha mencionado anteriormente, la programación del contenido del indicador se hace en Java. Al hablar de contenido hablamos de: Tipo de gráfica, filtros a incluir, gestión de los datos y de las consultas, etc.

Para especificar qué tipo de gráfica deseamos creamos un objeto “ChartFactory”. Como el nombre del objeto indica, se trata de un objeto “Factory” el cual generará la gráfica que le indiquemos.

```
ChartFactory chartFactoryLine;

ChartFactory chartFactoryColumn;
```

Code 23: Atributo ChartFactory

```
this.chartFactoryLine = new ChartFactory();

ChartConfiguration chartConfiguration = chartFactoryLine.getChartConfiguration();

chartConfiguration.setTitulo("NUMERO CLIENTES");
chartConfiguration.setSubtitulo("Mostrando evolución anual del número de Clientes");
chartConfiguration.setTipo(Types.LINE);
```

Code 24: Iniciación de ChartFactory y atributos

En caso de querer otro tipo de componentes, como un “Label”, tan solo tenemos que crear el objeto.

```
LabelValorTotal labelTotal;
```

Code 25: Atributo LabelValorTotal

```
labelTotal = new LabelValorTotal();
labelTotal.getTextoVentas().setValue("NUM. CLIENTES: " + "0");
```

Code 26: Iniciación de LabelTotalVentas y atributos

Una de las mayores ventajas que proporciona Vaadin, es la posibilidad de crear componentes personalizados. Como se puede apreciar en el fragmento de código anterior, el objeto “Label” que hemos creado se trata un objeto “LabelValorTotal”. Esto se debe a que dicho objeto es una extensión de la clase “CustomComponent”, clase que permite crear nuevos tipos de componentes utilizando otros ya existentes como base.

```
public class LabelValorTotal extends CustomComponent
{
```

Code 27: Herencia de CustomComponent

```
Label textoVentas;

HorizontalLayout hLayout;
```

Code 28: Uso de componentes ya existentes

Al crear un nuevo componente, se debe especificar su anchura, márgenes, etc.

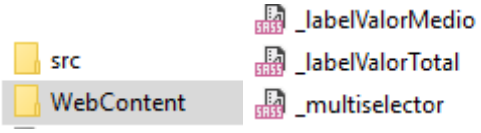
```
//Creamos panel horizontal
this.hLayout = new HorizontalLayout();
this.hLayout.setMargin(false);
this.hLayout.setSpacing(true);
this.hLayout.setSizeFull();

//Creamos label
this.textoVentas = new Label();
this.textoVentas.setSizeUndefined();

//Agregamos label al panel
this.hLayout.addComponent(textoVentas);

this.setCompositionRoot(this.hLayout);
```

Code 29: Especificación de márgenes, etc.

Sin embargo, esto no es suficiente para conseguir un indicador que siga una línea de diseño determinada. Mantiene márgenes y "padding", pero no se le ha especificado todavía ningún color o tipo de fuente.

Programación del estilo del indicador

Hasta ahora, no ha sido necesario tocar ningún lenguaje que no fuera Java o SQL, pero si queremos añadir un estilo acorde al resto del sistema a nuestro nuevo componente es necesario usar un lenguaje distinto. Al tratarse de una UI basada en Web, en la carpeta "WebContent" se encuentran los ficheros para aplicar estilos, sin embargo, no se trata de ".css" al uso.



Como se puede apreciar en la imagen superior, el estilo se aplica mediante ficheros ".SCSS". Estos ficheros deben nombrar de forma idéntica al componente al cual queremos aplicar estilo, además deben empezar con "_" para que Vaadin los reconozca a la hora de mapearlos con su componente.

El contenido de un fichero ".SCSS" no dista demasiado del contenido de un fichero ".css" normal:

```
@mixin tapbi-label-valor-total{
    .tapbi-label-valor-total{
        .v-label{
            background-color: $tapbi-color;
            color: white;
            font-family: "Roboto-Regular";
            font-size: 16px;
            padding: 10px
        }
    }
}
```

Code 30: Fichero .SCSS

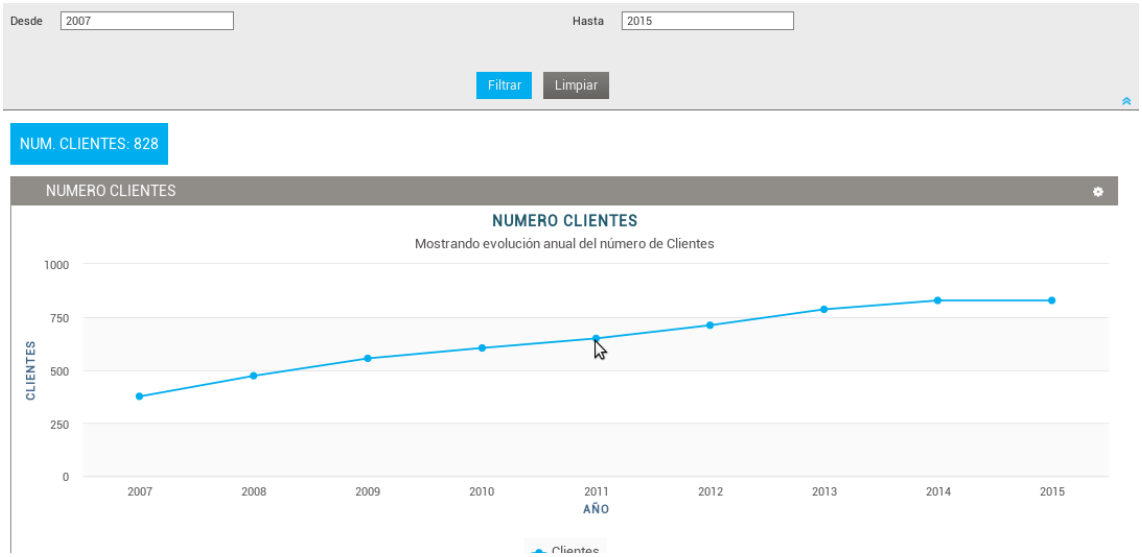
La jerarquía de clases de un fichero de estilo normal se aplica con normalidad y la sintaxis también es la misma. La única diferencia se puede apreciar en la primera línea, donde la orden @mixin se utiliza para que el fichero se pueda incluir en otros ficheros. De esta forma, cuando arranca el sistema puede se carga un único fichero que se encarga de llamar a todos los ".SCSS" con los estilos de los componentes.

```
@mixin TAPBI_theme {
  @include valo;
  @include tapbi-caption;
  @include tapbi-entrada-texto;
  @include tapbi-entrada-combo;
  @include tapbi-entrada-fecha;
  @include tapbi-entrada-color;
  @include tapbi-entrada-busqueda-fecha;
  @include tapbi-area-texto;
  @include tapbi-checkbox;
  @include tapbi-selector-fecha;
  @include tapbi-boton;
  @include tapbi-ventana-aceptar-cancelar;
  @include tapbi-ventana-seleccion;
  @include tapbi-barraBotones;
  @include tapbi-tabla;
  @include tapbi-multiselector;
  @include tapbi-label-valor-total;
  @include tapbi-label-valor-medio;
  @include tapbi-generador-calendario;
```

Code 31: Carga de ficheros .SCSS

Resultado

El resultado de esta unión entre Java y CSS es código JavaScript que permite la representación de gráficas como esta:



Lamentablemente, los pantallazos no permiten apreciar los efectos gráficos que se producen al elegir un indicador.