Universidad de las Ciencias Informáticas

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Título:

Desarrollo del componente Análisis del Sistema de Planificación de Producciones.

Autor(es): Alejandro Hernández Rodríguez

Tutores:

Ing. Disnayle Jorge Chacón

Ing. Yunior Orosa Velázquez

Ing. Erio Gutiérrez Llorens

La Habana, 2018.

Año 60 de la Revolución.

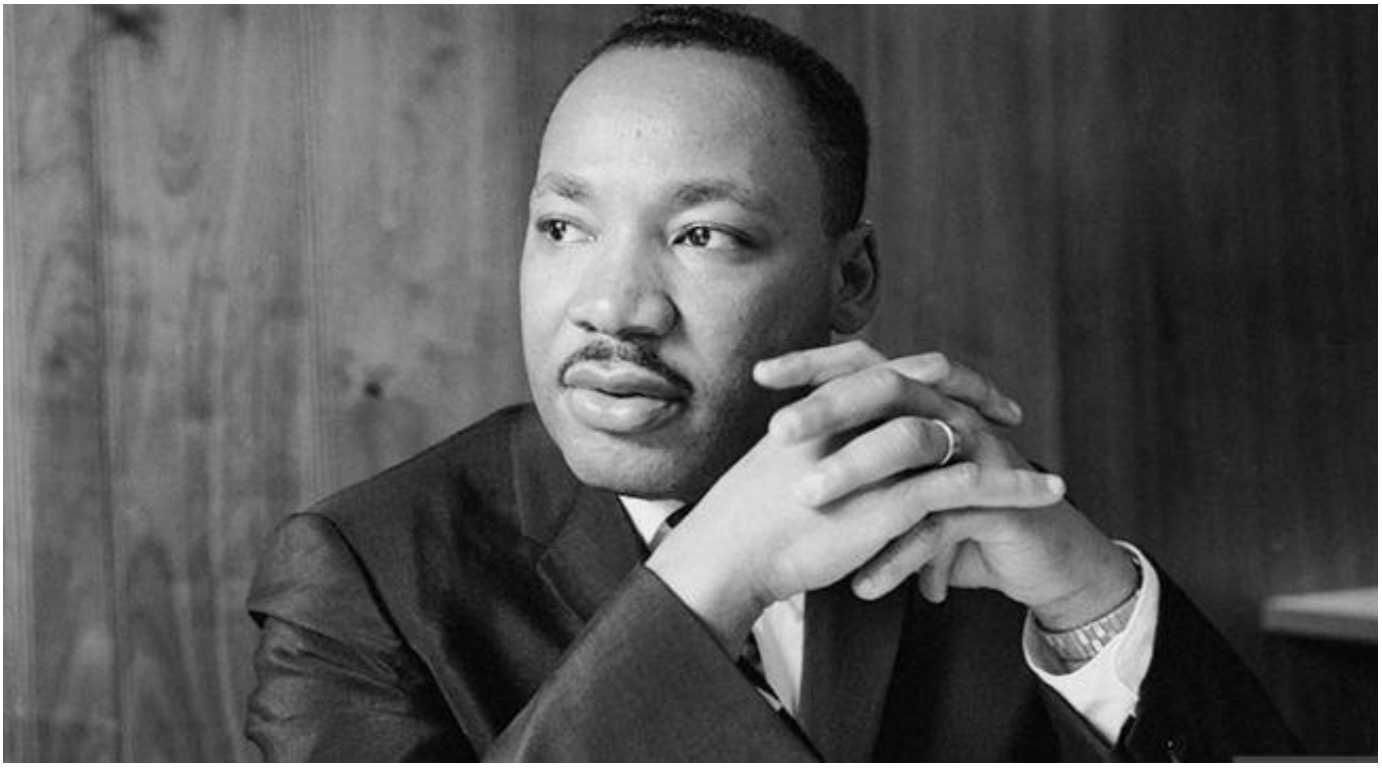
DECLARACIÓN DE AUTORÍA:

Declaramos ser autor(es) de la presente tesis y reconocemos a la XETID, Empresa de Tecnologías de la Información para la Defensa, los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los __ días del mes _____ del año _____.

Alejandro Hernández Rodríguez	Ing. Disnayle Jorge Chacón.
_____	_____
Firma de Autor	Firma de Tutor
Ing. Yunior Orosa Velázquez.	Ing. Erio Gutiérrez Llorens
_____	_____
Firma de Tutor	Firma de Tutor

Pensamiento:



“Nunca, nunca tengas miedo de hacer lo correcto, especialmente si el bienestar de una persona o animal está en juego. Los castigos de la sociedad son pequeños en comparación con las heridas que infligimos a nuestra alma cuando miramos para otro lado.”

-Martin Luther King-

Resumen

En la Empresa de Tecnologías de la Información para la Defensa (XETID), se ha desarrollado el Sistema de Gestión Empresarial Distra para llevar a cabo la gestión de los procesos de diferentes empresas. Para la planificación de producciones de la Unión Militar Agropecuaria se utiliza este sistema, el cual integra varios módulos. La utilización de los mismos por parte de los expertos se torna compleja, debido a la visualización de los datos y la cantidad de indicadores que influyen en el proceso de planificación, lo que trae consigo que entorpezca la toma de decisiones por parte de los expertos a la hora de definir el plan para cada granja agropecuaria a lo largo del país. El presente trabajo de diploma tiene como objetivo desarrollar el componente Análisis para contribuir a mejorar el proceso de toma de decisión de los expertos sobre la planificación de producciones dentro de la agricultura. La reciente investigación se desarrolló teniendo en cuenta el proceso de desarrollo PRODESOFT, y las tecnologías que contiene el marco de trabajo Zeolides (Ext.js, Zend, Doctrine), además se utilizó la librería Highcharts para el desarrollo de las gráficas. Como resultado se obtuvo un componente capaz de mostrar los datos de forma clara y oportuna, posibilitando a los expertos, facilidad en el proceso de toma de decisiones. La investigación finaliza con las pruebas funcionales del software, las cuales demostraron la correcta implementación de todas sus funcionalidades.

Palabras claves: *agropecuaria, componente, expertos, planificaciones, toma de decisiones.*

Índice

Introducción

El estudio de la Ciencia, la Tecnología y la Sociedad vincula el desarrollo científico y el avance tecnológico en pos de las necesidades de la sociedad. El uso de las Tecnologías de la Información y las Comunicaciones (TIC) permite realizar la gestión de la información eficientemente, eleva la productividad, la calidad, el control y favorece la comunicación en la gestión empresarial. Debido al impacto de las TIC en la gestión empresarial, el desarrollo de los sistemas automatizados sigue una línea ascendente haciendo el trabajo más eficiente y económico (SALINAS IBÁÑEZ, 2008).

En la actualidad el mundo ha experimentado profundos cambios debido a la creciente automatización de las empresas, los constantes cambios tecnológicos, la intensificación de la competencia internacional y a la gran diversidad de productos en el mercado. Esto ha conllevado a que las empresas empleen nuevas estrategias y métodos para realizar sus negocios (Romaní, et al., 2009).

La explotación de la web en las empresas es una notable ventaja en la informatización de dichas empresas debido a que se pueden crear sistemas con la capacidad de desarrollar procesos automatizados de sus subsistemas, facilitando así el trabajo y mayor rendimiento en las tareas y actividades, reduciendo la mano de obra y simplificando el trabajo; logrando que los procesos sean más rápidos y eficientes. Posibilita obtener una reducción de costes: racionalizando el trabajo y reduciendo el tiempo. Otra importante ventaja es la factibilidad de poder realizar procesos auditables que permitan la monitorización y evaluación de todos sus niveles en tiempo real (Perurena Cancio, et al., 2013).

La Empresa de Tecnología e Información para la Defensa (XETID), dedicada al sector del software, la automática y las comunicaciones, fomenta el avance tecnológico en nuestro país. Centra sus recursos a la evolución de sus productos informáticos bajo los principios de la seguridad, soberanía e independencia tecnológica. Sus soluciones velan por el cumplimiento de estándares nacionales e internacionales, en materia de brindar un servicio con calidad a sus clientes. Sus áreas temáticas abarcan procesos de la gestión del capital humano, la planificación, la gestión de suministros, la seguridad en las entidades y las comunicaciones entre otros temas que se pueden consultar en el sitio oficial de la entidad. La misma está llamada a respaldar la economía nacional mediante la producción de software en el mercado nacional (Ecured, 2017).

Dentro de la infraestructura productiva de la XETID, se encuentran los centros de producción especializados en distintos productos y servicios informáticos. Uno de ellos, es el Centro de Planificación, que es el

encargado de desarrollar sistemas de planificación material, financiera y estratégica para diferentes entidades y procesos.

Especial énfasis adquiere el sector agropecuario por su incidencia en la satisfacción de necesidades básicas de la población y el peso que tiene en función de garantizar la seguridad alimentaria y nutricional, tema que se ha declarado como prioridad de seguridad nacional.

El proceso de planificación de producciones agropecuarias tiene como objetivo realizar simulaciones de las producciones, utilizando el computador, para apoyar la toma de decisiones. El proceso de planificación de las producciones agropecuarias de cada granja agropecuaria del país, que se realiza por especialistas desde la Unión Agropecuaria Militar es una tarea muy compleja, debido a que, para una correcta planificación es necesario estudiar detalladamente cómo se comportan los factores e indicadores que intervienen en cada producción, para cada granja agropecuaria en específico.

El proceso de extracción de la información valiosa que se necesita para la toma de decisiones, se dificulta por la manera en, cómo se visualizan los datos, los cuales se muestran en tablas con gran cantidad de información en varias columnas. La mayoría de los datos son numéricos y se visualizan en diferentes unidades de medidas, lo que conlleva a que cause confusiones con mayor facilidad. Al ser la naturaleza de la información muy técnica en cuanto al negocio, puede causar el efecto contrario en los usuarios de ayudar a la toma de decisiones, así como dificultar la visualización del orden de cumplimiento, tiempo de duración, relaciones y dependencias entre las actividades de una producción.

Por las razones anteriores se plantea el siguiente **problema de investigación:**

¿Cómo contribuir a facilitar el proceso de toma de decisión de los expertos sobre la planificación de producciones dentro de la agricultura?

Para brindarle una solución al problema planteado se declara como **objeto de estudio** el proceso de análisis de la planificación de producciones. El **campo de acción** se centra en el proceso de análisis de la planificación de producciones dentro de la agricultura. Teniendo en cuenta la problemática planteada se define como **objetivo general** desarrollar el componente Análisis para contribuir a facilitar el proceso de toma de decisión de los expertos sobre la planificación de producciones dentro de la agricultura. El objetivo general planteado será desglosado en los siguientes **objetivos específicos**:

- Elaborar el marco teórico de la investigación a partir del estado del arte referente al proceso de análisis de la planificación de producciones.
- Identificar los aspectos funcionales y no funcionales que debe tener el sistema.
- Realizar las actividades y artefactos que tributen al diseño de la propuesta de solución, de acuerdo al proceso de desarrollo utilizado.
- Realizar las actividades y artefactos que tributen a la implementación de la propuesta de solución, de acuerdo al proceso de desarrollo utilizado.
- Realizar pruebas a las funcionalidades implementadas.

Idea a defender

El desarrollo del componente Análisis del Sistema de Planificación de Producciones permite facilitar el proceso de toma de decisiones en la planificación de producciones dentro de la agricultura.

Los **métodos científicos y técnicas de investigación** utilizados en la elaboración de este trabajo fueron los siguientes:

Métodos teóricos:

- Analítico-Sintético: Este método se utilizó para analizar las teorías, datos y documentos existentes permitiendo extraer los elementos más importantes que se relacionan con el objeto de estudio (GUIBOURG, et al., 1998), en este caso aplicado al proceso de planificación de producciones, usándose la información más adecuada para la realización del presente trabajo.
- Análisis histórico-lógico: Este método se utilizó para realizar un análisis en cuanto a la evolución del proceso de planificación de producciones a nivel mundial (GUIBOURG, et al., 1998).

Técnicas empíricas:

- Entrevistas: Se realizaron encuentros o entrevistas no formales a los especialistas de la entidad de planificación de XETID y especialistas de la Unión Agropecuaria Militar (UAM) para obtener información acerca del proceso de planificación de producciones.

El presente trabajo está estructurado en tres capítulos, los cuales se resumen a continuación:

Capítulo 1: Fundamentación teórica del Desarrollo del componente Análisis del Sistema de Planificación de Producciones. En este capítulo se hace un análisis de los principales conceptos relacionados con el dominio del objeto de estudio de la investigación y se realiza un estudio del estado del arte de herramientas para la creación del componente del Sistema de Planificación de Producciones. Además, se seleccionan las herramientas, tecnologías y proceso de desarrollo que guiará el ciclo de vida del producto.

Capítulo 2: Descripción del Desarrollo del componente Análisis del Sistema de Planificación de Producciones. En este capítulo se describe la solución de software propuesta. Abarca las fases de Exploración y Planificación del proceso de desarrollo en la solución de software. En este se generan Plan de entrega, donde queda definida la velocidad del proyecto y cuándo el cliente obtendrá cada entregable del producto.

Capítulo 3: Diseño, implementación y pruebas del Desarrollo del componente Análisis del Sistema de Planificación de Producciones. En este capítulo se describe la arquitectura del sistema, con los patrones aplicados a la solución. Además, se generan las tareas de ingeniería en la etapa de implementación y se realiza el proceso de pruebas generando los casos de pruebas necesarios.

Se incluyen además elementos complementarios en los **Anexos**.

Capítulo I: Fundamentación teórica del Desarrollo del componente Análisis del Sistema de Planificación de Producciones

El objetivo principal de este capítulo se centra en realizar un estudio de los conceptos necesarios para llevar a cabo la investigación, enfocándose principalmente en los Componentes y Módulos para el proceso de análisis de planificación de producciones, que se pueden implantar en empresas, para luego identificar el proceso de desarrollo o metodología, lenguajes y herramientas a utilizar para el desarrollo del componente Análisis del Sistema de Planificación de Producciones.

1.1 Componentes de Análisis de planificación de producciones

Un componente de análisis de planificación de producciones es un subsistema o subconjunto de funcionalidades de un sistema mayor. El mismo ofrece la exposición de los datos relevantes de la planificación de una o varias producciones con el propósito de obtener planes de trabajo, para determinar de forma anticipada los objetivos o escenarios que una organización pretende alcanzar y lo que hará para conseguirlo (R. Demey, et al., 1992).

Para el desarrollo de la presente investigación se exponen los siguientes conceptos del negocio para un mejor entendimiento de los procesos:

- **Planificación agropecuaria:** Es el acto de definir con antelación los objetivos que se prevén alcanzar en un momento determinado, qué acciones son necesarias acometer para alcanzarlos, quiénes van a realizarlas, dónde y cuándo, así como los recursos que se necesitan para realizarlas y el importe de dichos recursos, esto, aterrizado al sector agropecuario (Hernández Espino, et al., 2017).
- **Entidad:** Se refiere a estructura y/o subestructura organizativa (Hernández Espino, et al., 2017).
- **Carta tecnológica:** Se refiere a un documento que define las labores requeridas para cada cultivo, el orden de su realización, el tiempo y los aseguramientos que demanda cada una, lo cual permite determinar el volumen de las necesidades de insumos, equipos, implementos y fuerza de trabajo, estas labores se le conocen como actividades (Hernández Espino, et al., 2017).
- **Actividad:** Se refiere a un conjunto de acciones planificadas llevadas a cabo por personas en un momento dado, las actividades tienen asociada una norma (Hernández Espino, et al., 2017).

- **Norma:** Se refiere a la norma de consumo y de tiempo de una actividad (Hernández Espino, et al., 2017).
- **Producción:** Se refiere a la aplicación de una carta tecnológica con un consumo de infraestructura, en un período determinado (Hernández Espino, et al., 2017).
- **Infraestructura productiva:** Concepto que define el espacio de trabajo (área de campo cultivable, corrales o cuartones, naves, maquinaria moledora) (Hernández Espino, et al., 2017).
- **Ficha de costo:** Se refiere al documento donde se refleja la información relacionada con los costos, precios y gastos de la producción (Hernández Espino, et al., 2017).
- **Arrendamiento:** Se refiere a los arrendamientos de indicadores realizados en una entidad, en una carta tecnológica para un plazo determinado y con un costo asociado (Hernández Espino, et al., 2017).
- **Servicio:** Se refiere a cuando se alquila un servicio para el cumplimiento de una actividad (Hernández Espino, et al., 2017).
- **Depreciación:** Se refiere a la depreciación que pueda tener un equipo o un implemento. Este término de depreciación se refiere a una reducción del valor de un equipo o de un implemento por desgaste debido al uso, el paso del tiempo o por obsolescencia (Hernández Espino, et al., 2017).

1.2 Sistemas homólogos

Un sistema ERP es una solución informática integral que está formada por unidades interdependientes denominadas Módulos. Los primeros y fundamentales son los denominados Módulos Básicos, de adquisición obligatoria, y alrededor de los cuales se agregan los otros módulos opcionales, que no se adquieren obligatoriamente y se agregan para incorporar nuevas funciones al sistema ERP (DÍAZ, et al., 2005).

Algunos de los componentes y módulos que utilizan los principales sistemas ERP a nivel internacional que se encargan del análisis de las producciones se describen a continuación.

1.2.1 Visual Schedulig, de Netronic para Dynamics NAV

Visual Schedulig de Netronic, es un módulo de Dinamics NAV que permite planificar y programar las necesidades de los departamentos de Fabricación, Producción y Proyectos (Gutiérrez, 2017).

Licencia que utiliza:

- Propietario

Visual Schedule ofrece 4 beneficios principales para la planificación de producciones:

- Visualización rápida de las órdenes de producción, servicios y proyectos.
- Mejora de la capacidad de los recursos gracias a un marco de tiempo definido.
- Gráficos interactivos y dinámicos: el usuario puede modificar la planificación con un sólo clic.
- Notificaciones y alertas en caso de sobreproducción de alguno de los recursos.

1.2.2 Odoo

Odoo, un potente sistema de planificación y gestión de los recursos empresariales, preparado para trabajar en los procesos de negocio cubriendo las áreas de contabilidad, finanzas, ventas, RRHH, compras, proyectos y almacén entre otras, de manera fácil y sencilla. Odoo es un sistema completo basado en código abierto que garantiza la personalización y adaptación a las necesidades de cada negocio, de forma sencilla, fácil y sin costes de licencias (guadaltech, 2018).

Licencia que utiliza

Los módulos de Odoo, en su mayoría, están cubiertos por la licencia AGPL y algunas partes utilizan una derivada de la licencia Mozilla Public License.1 Como consecuencia directa, OpenERP no requiere ningún pago de licencias para ser utilizado, a diferencia de los softwares más usados del mercado. Esto también implica que, mientras que se respeten los términos de la licencia, la modificación directa del programa es posible.

Lenguajes en que esta implementado

- Base de datos: Postgresql.
- Lenguaje de programación: Python.

- Lenguaje de programación web: Html5, javascript y css.
- Está dotado de un entorno/framework de desarrollo rápido de aplicaciones (RAD) denominado Openobject.
- Desde su versión 8.0 (Odoo, anteriormente OpenERP) incluye también un entorno de desarrollo web que permite construir aplicaciones móviles y web a medida.

Su módulo de producciones posibilita:

- Generación de órdenes de fabricación en base a rutas y lista de materiales.
- Planificación temporal de las producciones, cargas de trabajo.
- Uso de códigos de barras.
- Trazabilidad completa.
- Vinculación ventas y almacén.
- Informes sobre las producciones, cumplimientos.

1.2.3 Doeet®

Doeet® es un sistema automatizado que realiza el control de productividad, debido a que monitoriza en tiempo real toda la actividad de las máquinas y crea un registro de las paradas, velocidad real, unidades fabricadas, parámetros de calidad y otras variables que intervienen en la Producción y los compara con sus valores objetivos (MES, 2018). Doeet® controla y registra todos los consumos, mermas, Scrap, retrabados, controles de calidad, controles de parámetros, asegurando la trazabilidad de la producción.

Con toda esta información elabora informes con gráficos y datos en los que se puede consultar y analizar el estado actual de la Producción, y tomar las medidas necesarias para su mejora (MES, 2018). El sistema cuenta con una batería de informes ya preinstalados, para usarse rápidamente.

Los **módulos doeet** cubren necesidades específicas de la Producción, ampliando las posibilidades del sistema y facilitando el tratamiento de las operaciones y la información en diferentes áreas como el Control de la Trazabilidad, la gestión del Mantenimiento, etc. Estos módulos están diseñados para adaptarse e incorporarse a la empresa en el momento que desee, bien en el momento de la implantación, o en una fase posterior, cuando la implantación del sistema esté consolidada (MES, 2018).

Los módulos han sido elaborados por expertos en Organización Industrial y *Lean Manufacturing* que durante mucho tiempo han trabajado en diferentes procesos industriales. Conscientes del poco tiempo que se dispone en el día a día, para analizar la información, los informes están diseñados para llegar desde un análisis general a un detalle particular en pocos segundos. Te permite personalizar los informes y pantallas con los logos y colores de su empresa (MES, 2018).

También permite crear causas de paros y guardar los filtros que utilices con más frecuencia. Además, el **Desarrollador de Informes** permite crear informes personalizados con los datos y gráficos que requiera el control de productividad. Permite diseñar el terminal del operario en planta, para que le resulte más sencillo y rápido introducir la información que se le requiere (MES, 2018).

1.2.4 Selenne ERP

Selenne ERP es un software de planificación empresarial que ofrece soluciones especializadas en sectores de fabricación, ingenierías, industria química, proyectos e inspecciones que gestiona todas las áreas de negocio en un único sistema flexible que evoluciona continuamente gracias al Ecosistema Selenne (Selenne, 2016).

Datos del producto:

- **Empresa/Fabricante**: Selenne ERP.
- **Tipo de licencia**: Compra.
- **Tipo de implantación**: Cliente Servidor, Entorno Web.
- **Sectores**: Industria Química, Ingeniería y Operaciones, Metal, Papel y Madera, Retail (Distribución Comercial), Servicios Profesionales (Synerplus, 2016).

Tecnología

Plataforma Selenne ERP cambia los paradigmas de los sistemas de software. Se basa en complejas arquitecturas de procesos, separando la funcionalidad de los procesos para servir de núcleo cohesionado y universal para la generación de la integridad absoluta entre todas las entidades del sistema. Está en constante crecimiento, ya que se trata de un sistema evolutivo preparado para el cambio, con una enorme reducción de códigos, que invocan procesos, altamente depurados y estables, también garantiza una integridad completa entre todas las entidades (Synerplus, 2016).

Selenne ERP reutiliza una gran cantidad de procesos y funciona como un único ente central que responde a todas las necesidades como sistemas complejos de fabricación industrial, movilidad de *tablets*, integración con maquinaria, integración con proveedores, planificadores para una central de compras, multi-localización fiscal internacional, distribución, integración con y hacia sistemas, así como muchas otras ventajas (Synerplus, 2016).

La profundidad tecnológica de Selenne, cubre una gran cantidad de procesos de negocio para las empresas de fabricación y vinculados a las actividades productivas. Tiene asociados ciclos completos que permiten generar ofertas y presupuestos con altísimos niveles de complejidad, también permite obtener una planificación de actividades productivas, solucionar definitivamente las compras de artículos que forman a su vez, parte de otros mayores, adecuar la puesta en marcha con órdenes de trabajo fijos y en *tablets*, gestiones de envasado, etiquetado y almacenes, entre muchas otras características (Synerplus, 2017).

Selenne Software ERP permite una gestión total de costes sobre cada fase de proyecto permite hacer el seguimiento en tiempo real, modificar el curso de la producción y conocer los cambios que se están produciendo para intentar corregirlos (Synerplus, 2017).

1.2.5 Sismagro

El software funciona online. Se puede usar la versión de Sismagro Freemium de manera gratuita por tiempo ilimitado. La limitación que tiene es de un solo usuario, un campo, 10 lotes y una determinada cantidad de superficie según la actividad. Si se precisa más, se debe adquirir versiones superiores a partir de 19$US por mes. No existe contratos. Se paga por lo que se usa. Sismagro trae incluido un paquete de tutoriales para mayor entendimiento y utilización (Sismagro, 2015). Las principales funcionalidades que implementa son:

Mapeo satelital

Los mapas satelitales, permiten agregar en menos de un minuto la geolocalización de los lotes asociados a nuestra cuenta y obtener la superficie cultivable. En pocos clicks y sin ninguna capacitación necesaria, gestiona automáticamente todo lo necesario para manejar el campo (Sismagro, 2015).

Seguimiento de la producción

A partir del mapa y de cultivos, permite gestionar producciones agrícolas guardando trazabilidad de todos los costos e insumos. También al calendario se agregan las labores, lo cual permite visualizar de manera simple todo lo que está pasando en el campo y en los lotes (Sismagro, 2015).

Planificación de la Producción agropecuaria

Esta funcionalidad permite planificar las labores y gestionar los costos dentro del calendario para toda la campaña. Anticipa el costo y la ganancia que se obtiene en cada lote antes de comenzar el trabajo para una mejor preparación. Escoge el mejor plan y lo ejecuta en producción (Sismagro, 2015).

Manejo de *Stock* de galpones

Agregar fácilmente los datos de los insumos y herramientas apoyándose en la base de datos para supervisar el *stock* de galpones y sus movimientos. Saber cuánto se tiene almacenado, su origen y ubicación (Sismagro, 2015).

1.2.6 Resultado del análisis de los sistemas homólogos

El análisis de los sistemas homólogos arrojó que no es viable su utilización en el desarrollo de la presente investigación, debido a que las herramientas estudiadas presentan indistintamente algunas de las siguientes causas:

- La instalación del sistema ERP es muy costosa.
- Los vendedores de los ERP pueden cobrar sumas de dinero para la renovación de sus licencias anuales, lo cual no está relacionado con el tamaño del ERP de la empresa en donde se implemente o las ganancias que tengan las mismas.
- Los ERP son muy rígidos, difíciles de adaptarse al flujo específico de los trabajadores y el proceso de negocios de algunas compañías, este punto se considera como una de las principales causas de falla.
- Los sistemas pueden ser difíciles de usar.
- Los sistemas pueden sufrir problemas de "cuello de botella": la ineficiencia en uno de los departamentos o en uno de los empleados puede afectar a otros participantes.
- No se ajustan a las producciones agropecuarias.

- La mayor parte de la documentación es restringida por políticas de seguridad.

Sin embargo, estas herramientas poseen un grupo de características que se tuvieron en cuenta para el desarrollo de la presente investigación las cuales son:

- Utilización de gráficas interactivas y dinámicas.
- El aprovechamiento óptimo y la mejor asignación de los recursos disponibles usando el modelo de programación lineal.
- Análisis de indicadores como horizonte de planificación, capacidad de producción instalada, cantidades a fabricar en cada período, nivel de los inventarios, Objetivo global.
- Notificaciones y alertas en caso de sobreproducción de alguno de los recursos.
- Análisis de fórmulas para el cálculo de costes de recursos materiales y de tiempo.
- Tolerancia a cambios de última hora.
- Crear informes personalizados con los datos y gráficos que requiera el control de productividad.
- Mapeo satelital de la infraestructura.

1.3 Proceso de Desarrollo y Gestión de Proyectos de Software PRODESOFT

Se decide utilizar PRODESOFT como proceso de desarrollo por ser el definido a utilizar por la dirección de la entidad de planificación de XETID. El uso del mismo propicia que la construcción y desarrollo de software sea un proceso menos complejo.

Este proceso de desarrollo cuenta con cinco fases secuenciales: Inicio, Modelación, Construcción, Explotación Experimental y Despliegue, conocidas como el ciclo de vida del proyecto (Curbelo Oliva, et al., 2008).

Inicio: En esta etapa se debe lograr una visión preliminar de la problemática a resolver y se definen los recursos relevantes para la ejecución del proyecto. Se estima de manera general las actividades a realizar durante todo el ciclo de desarrollo del proyecto (Curbelo Oliva, et al., 2008).

Modelación: Se capturan las partes esenciales del sistema, donde se identifican los procesos de negocio fundamentales y se aceptan los requisitos funcionales, obteniéndose la línea base de la arquitectura y una

estrategia de construcción de la aplicación aprobada por los implicados en el proyecto (Curbelo Oliva, et al., 2008).

Construcción: Se aclaran los requisitos restantes y se completa el desarrollo del sistema sobre una base estable de la arquitectura. En esta fase todas las características, componentes, y requisitos deben ser integrados, implementados, y probados en su totalidad, obteniendo una versión liberada del producto (Curbelo Oliva, et al., 2008).

Explotación Experimental: Se convierte la versión liberada del producto en una solución estable, donde se eliminan los errores que surgen durante las pruebas y se obtiene una certificación funcional y de seguridad del producto (Curbelo Oliva, et al., 2008).

Despliegue: Se instala y configura el sistema para un ambiente de producción real, se capacita al personal que usará la aplicación y se continúa dando soporte durante la explotación del sistema (Curbelo Oliva, et al., 2008).

1.3.1 Enfoques que utiliza el proceso de desarrollo PRODESOFT

Iterativo e incremental: Es un enfoque en el que el ciclo de vida está compuesto por iteraciones, estas son pequeños procesos compuestos de varias actividades cuyo objetivo es entregar una parte del sistema parcialmente completo, probado, integrado y estable. En cada iteración se obtiene como resultado un incremento (Curbelo Oliva, et al., 2008).

Basado en componentes: Lleva a alcanzar un mayor nivel de reutilización de software, aún en contextos distintos de aquellos para los que fue diseñado. Permite que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados. Cuando existe un débil acoplamiento entre componentes, el desarrollador es libre de actualizar y/o agregar componentes según sea necesario, sin afectar otras partes del sistema. Dado que un componente puede ser construido y luego mejorado continuamente, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo (Curbelo Oliva, et al., 2008).

Orientado a procesos: La modelación de proceso de negocio permite realizar una rápida y profunda exploración del dominio del problema, con el fin de lograr comprensión por parte del equipo de desarrollo de los procesos que se realizan actualmente en la entidad y la relación que existe entre estos (Curbelo Oliva, et al., 2008).

1.4 Lenguajes de modelado

1.4.1 UML (Unified Modeling Language)

Es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. Es un estándar en la industria del software, creado por Grady Booch, James Rumbaugh e Ivar Jacobson (RUMBAUGH, et al., 2017).

UML también intenta solucionar el problema de propiedad de código que se da con los desarrolladores, al implementar un lenguaje de modelado común para todos se crea una documentación que cualquier desarrollador con conocimientos de UML será capaz de entender. Su utilización es independiente del lenguaje de programación y de las características de los proyectos, ya que UML ha sido diseñado para modelar cualquier tipo de proyectos, tanto informáticos como de arquitectura, o de cualquier otra rama. Se estará haciendo uso del UML 2.0 (RUMBAUGH, et al., 2017).

1.4.2 BPMN (Business Process Modeling Notation)

La Notación para el Modelado de Procesos de Negocio por sus siglas en inglés (BPMN) es una notación gráfica estandarizada que permite el modelado de procesos de negocio. Su principal objetivo es proveer una notación estándar que sea fácilmente legible y comprensible tanto para los usuarios del negocio, los analistas de procesos de negocio, los desarrolladores encargados de la aplicación de la tecnología que llevará a cabo esos procesos como para los trabajadores del negocio que van a administrar y supervisar esos procesos. El modelado en BPMN se realiza mediante diagramas muy simples con un conjunto de elementos gráficos. Con esto se busca que para los usuarios del negocio y los desarrolladores técnicos sea fácil entender el flujo y el proceso. BPMN proporciona un medio sencillo de comunicar información de procesos a los usuarios de otros negocios, los ejecutores de procesos, clientes y proveedores (RUMBAUGH, et al., 2017).

1.5 Herramientas de Modelado

Herramienta CASE: Las Herramientas CASE (*Computer Aided Software Engineering* en español Ingeniería de Software Asistida por Ordenador) son aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas proporcionan ayuda a los analistas, diseñadores y desarrolladores en todos los aspectos del ciclo de vida de desarrollo del software (RUMBAUGH, et al., 2017).

1.5.1 Visual Paradigm

Visual Paradigm es una herramienta CASE multiplataforma que ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite importar y exportar ficheros XML, así como la generación de código para Java y exportación HTML. Es una herramienta fácil de usar que soporta ingeniería directa e inversa (código a modelo, código a diagrama), diagramas de flujo de datos y generación de código e informes (RUMBAUGH, et al., 2017).

Se decide utilizar como herramienta *CASE* a *Visual Paradigm* principalmente por permitir el modelado de diagramas para BPMN y por ser una tecnología multiplataforma. Es la herramienta definida por la dirección de la entidad de planificación de XETID aprobada por el cliente.

1.6 Lenguajes de programación

1.6.1 PHP 5

PHP es el acrónimo de Hipertext Preprocesor. Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación (Álvarez, 2015).

El principal objetivo de PHP5 ha sido mejorar los mecanismos de POO para solucionar las carencias de las anteriores versiones. Un paso necesario para conseguir que PHP sea un lenguaje apto para todo tipo de aplicaciones y entornos, incluso los más exigentes (Álvarez, 2016).

1.6.2 Javascript

JavaScript (abreviado comúnmente JS) es un lenguaje de programación interpretado, dialecto del estándar *ECMAScript*. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico (Ecma, 2017).

Se utiliza principalmente en su forma del lado del cliente (*client-side*), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, aunque existe una forma de JavaScript del lado del servidor (*Server-side JavaScript o SSJS*). Su uso en aplicaciones externas a la web, por ejemplo, en documentos PDF, aplicaciones de escritorio (mayoritariamente widgets) es también significativo (Ecma, 2017).

1.7 Framework's utilizados

Framework

Un *framework* (Marco de Trabajo) es una estructura de software compuesta por componentes personalizables e intercambiables para el desarrollo de una aplicación; está compuesto por arquitecturas definidas para un determinado dominio de la aplicación que contiene un conjunto de componentes implementados y sus interfaces bien definidas, estos componentes se pueden utilizar, redefinir y crear nuevos componentes (SOUZA, et al., 2009).

1.7.1 Ext 2.2

ExtJS es una librería construida con JavaScript que proporciona una interfaz a las librerías de *Yahoo!*, *jQuery* y *Prototype* + *Scriptaculous*, su potencia radica en la rica colección de componentes para el diseño del lado del cliente haciendo uso extensivo de Ajax. Es neutral al lenguaje que se use en el servidor. Siempre que el resultado se envíe a la página en el formato adecuado, *ExtJS* no se ocupa de lo que suceda en el servidor (ORCHARD, 2009).

Proporciona un selector de nodos *Document Objet Model* (DOM) extremadamente poderoso llamado *DomQuery* (puede usarse como una librería independiente, pero en el contexto de *ExtJS* se usará para seleccionar elementos y poder interactuar con ellos a través de la interfaz *Element,* contiene muchos de los métodos y propiedades de DOM) (ORCHARD, 2009).

1.7.2 Doctrine 1.2.1

Doctrine es un *Object Relational Mapper* (ORM) realizado en PHP, que es una potente capa de abstracción de la base de datos. Otra característica importante de Doctrine es la posibilidad de escribir consultas de base de datos utilizando un dialecto de *Structured Query Language* (SQL) denominado Doctrine *Query Language* (DQL) que está inspirado en *Hibernate* (Java) (WAGE, et al., 2010).

Un ORM es una técnica de programación que permite convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, es decir, las tablas de la base de datos pasan a ser clases y los registros objetos que se pueden manejar con facilidad (WAGE, et al., 2010). Las principales ventajas que presenta un ORM son:

1. Reutilización.

2. Encapsulación.
3. Portabilidad.
4. Seguridad.
5. Mantenimiento del código (WAGE, et al., 2010).

1.7.3 Zend Ext Framework (ZF) 1.9

Es un *framework* de código abierto para desarrollar aplicaciones web y servicios web con PHP 5. Zend Ext Framework es una implementación que usa código orientado a objetos. La estructura de los componentes de ZF es única; cada componente está construido con una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada permite a los desarrolladores utilizar los componentes por separado (SHASANKAR, 2013).

1.7.4 Highcharts v.4.2.3

Esta librería permite crear gráficas interactivas fácilmente para crear proyectos web. Usado por cientos de miles de desarrolladores y usado por más de 61 de las empresas más grandes del mundo, Highcharts es la solución más rápida, flexible y útil API de JavaScript en el mercado. Por la cantidad de funciones que ofrece, esta librería es paga, si el proyecto no usa la librería con fines comerciales con ámbito internacional, puede ser usada de manera gratuita bajo la licencia de Creative Commons Attribution-NonCommercial 3.0 (KUAN, 2012).

Highcharts tiene más componentes como Highmaps (mapas dinámicos con JavaScript) y *Highstocks* (contiene más gráficas) (KUAN, 2012).

- Usa SVG.
- Soporte para múltiples exploradores.
- Altamente personalizable.
- Muchos gráficos.

1.8 Sistemas gestores de base de datos

Se denomina Sistema Gestor de Base de Datos (siglas: SGBD) al conjunto de programas que permiten definir, construir y mantener una base de datos, asegurando su integridad, confidencialidad y seguridad (RAMÍREZ, 2006).

1.8.1 PostgreSQL

PostgreSQL es un sistema gestor de base de datos relacional y el gestor libre líder en el mundo. Se destaca en ejecutar consultas complejas, consultas sobre vistas, sub-consultas y *joins* de gran tamaño. Permite la definición de tipos de datos personalizados e incluye un modelo de seguridad completo (MOMJIAN, 2001).

Queda definido como Sistema gestor de base de datos PostgreSQL en su versión 9.4.1 definida por el marco de trabajo Zeolides. Está definida por la dirección de la entidad de planificación de XETID aprobada por el cliente.

1.9 Entorno de desarrollo Integrado (IDE)

Las herramientas de desarrollo son aplicaciones informáticas que tienen cierta importancia en el desarrollo de un producto de software. Puede ser un compilador, ensamblador, editor o un Entorno de Desarrollo Integrado (IDE). Un IDE es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien poder utilizarse para varios (FUENTES, et al., 2003).

1.9.1 NetBeans

NetBeans IDE es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. Este es un producto libre y gratuito sin restricciones de uso. NetBeans IDE proporciona soporte para C / C + + y los desarrolladores de PHP, proporcionando editores y herramientas integrales para sus marcos y tecnologías relacionadas. Además, el IDE tiene editores y herramientas para XML, HTML, PHP, Groovy, Javadoc, JavaScript y JSP (BOUDREAU, 2002).

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Debido a que los módulos pueden ser desarrollados

independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software (BOUDREAU, 2002).

Después de varios análisis fue seleccionado como herramienta de desarrollo a Netbeans, por ser un producto libre y gratuito sin restricciones de uso, además proporciona soporte para muchos lenguajes de programación entre los que se encuentra PHP.

1.10 Marco de trabajo del ERP

1.10.1 Zeolides 2.0

Es un conjunto de librerías, herramientas, tecnologías y componentes de software integrados en un marco de trabajo para desarrollar aplicaciones de múltiples propósitos, gran tamaño y grandes volúmenes de datos, que permiten el desarrollo ágil, basado en componentes, centrado en los requerimientos del usuario, las interfaces de usuario y la lógica del negocio de las aplicaciones que con el mismo se desarrollen, aunque puede utilizarse con múltiples propósitos, por ejemplo para aplicaciones de tiempo real, su objetivo fundamental es el desarrollo de soluciones web de gestión empresarial.

1.11 Servidor para aplicaciones web

Es un programa que permite crear un servidor http en un ordenador de una forma rápida y sencilla. Está diseñado para ser un servidor web potente y flexible que pueda funcionar en la más amplia variedad de plataformas y entornos. Es además un programa que implementa el protocolo HTTP el cual se encarga de transferir hipertextos, páginas web o páginas HTML: textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música (LUJÁN MORA, 2002).

1.11.1 Servidor web Apache

Es una tecnología gratuita de código fuente abierta, puede ser usado en varios sistemas operativos, lo que lo hace prácticamente universal. Es un servidor altamente configurable es decir se pueden elegir qué características van a ser incluidas en el servidor seleccionando que módulos se van a cargar, ya sea al compilar o al ejecutar el servidor (MONTOYA, et al., 2013).

Queda definido por la dirección de la entidad de planificación de XETID y aprobado por el cliente que se utilice como servidor para aplicaciones web Apache en su versión 2.0 o superior.

1.12 Herramientas complementarias para el desarrollo

1.12.1 Control de versiones

Capacidad de recordar todos los cambios que se hacen tanto en la estructura de directorios como en el contenido de los ficheros, cuando más de una persona trabaja con los mismos archivos y aun cuando es una sola persona resulta imprescindible mantener cierto control sobre los cambios que se realizan: quién, cuándo, qué. Si los cambios realizados por dos personas son incompatibles y es necesario tomar una decisión sobre la forma definitiva del archivo, una vez detectado los cambios realizados que no siguen el camino apropiado restablecerlos a una versión anterior (RUIZ-BERTOL, et al., 2007).

Subversion (SVN)

Subversion es un sistema de control de versiones multiplataforma de código abierto. Este maneja ficheros y directorios, y los cambios que se le hayan realizado a los mismos, a través del tiempo. Esto le permite recuperar versiones antiguas de sus datos o examinar el historial de cómo cambiaron estos. Subversion puede funcionar a través de redes, lo que permite que pueda ser utilizado por varios usuarios en diferentes equipos, esto brinda a los usuarios la capacidad de poder modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones (MORAL, 2008).

El sistema de control de versiones definido a utilizar es SVN ya que es un sistema multiplataforma, es un software libre, además es muy rápido y seguro en el control de versiones a través de la red. Está definido por la dirección de la entidad de planificación de XETID.

1.12.2 Navegador web

Un navegador web es el software o programa que permite ver la información que contiene una página web. Traduce el código HTML en el que está escrita la página y lo muestra en la pantalla, permitiendo interactuar con su contenido y navegar hacia otras páginas o sitios de la red, mediante enlaces o hipervínculos. El seguimiento de los enlaces de una página a otra se llama navegación, que es de donde se origina el nombre de navegador web (SANTIAGO, et al., 2007).

Mozilla Firefox

Mozilla Firefox, o simplemente Firefox es un navegador web libre y de código abierto desarrollado por Mozilla, una comunidad global que trabaja junta para mantener una Web Abierta, pública y accesible. Es un

navegador totalmente configurable, tanto su funcionamiento, configuración, aspecto, *add-ons* o complementos. En su sitio web Mozilla ofrece toda la información técnica necesaria a desarrolladores y usuarios en general, alto nivel de seguridad, efectiva la protección contra el spyware y otros tipos de malware, bloqueo asegurado contra pop-up y otras formas de publicidad comunes en la web. Sus desarrolladores aseguran una fuente casi infinita de extensiones hechas para todo tipo de propósito, permite crear y utilizar simultáneamente varios perfiles o preferencias en el mismo navegador, lo cual en la práctica es muy útil, es decir puedes tener una configuración diferente para usar Firefox en las tareas laborales o estudiantiles y otra para el uso privado o familiar, todo con el mismo navegador en la misma PC (KHOMH, 2012).

El navegador escogido es Mozilla Firefox (v 30) o superior por ser un navegador libre, multiplataforma y por la amplia comunidad que tiene por detrás para su desarrollo y mantenimiento. Es un navegador rápido con aplicaciones Web complejas y presenta una efectiva protección contra el spyware y otros tipos de malware, bloqueo asegurado contra pop-up y otras formas de publicidad comunes en la web.

1.13 Herramientas para pruebas

1.13.1 Jmeter

JMeter es una aplicación de escritorio de código abierto para realizar pruebas funcionales de software y medir el rendimiento. Inicialmente se diseñó para pruebas de aplicaciones web, aunque en sus versiones posteriores, ha aumentado su funcionalidad a otro tipo de pruebas (Caballero Redondo, 2012). Sus funcionalidades se pueden resumir en tres:

- Diseñar un testplan, esto es generar un archivo. jmx.
- Ejecutar un testplan.
- Ver de distintas formas los resultados de la ejecución de un testplan.

1.13.2 Firebug

Es un plug-in de Firefox: permite inspeccionar fácilmente el código HTML y XML devuelto por la aplicación objetivo, en el proceso de implementación del testplan (fichero. jmx) (McLean, 2018).

Conclusiones del capítulo

En este capítulo se realizó un análisis del estudio del estado del arte referente a módulos y componentes de análisis para la planificación de producciones existentes lo cual permitió identificar los elementos fundamentales para el desarrollo de la presente investigación. Se realizó un estudio del proceso de desarrollo (PRODESOFT), establecido en la XETID para la modelación y la implementación del componente el cual permitió definir las etapas en las que se desglosa el proceso de desarrollo del mismo.

Capítulo II: Descripción del Desarrollo del componente Análisis del Sistema de Planificación de Producciones

En el presente capítulo se describen las principales características que debe presentar la solución propuesta. Esta solución es basada en la definición de los requisitos y la arquitectura. Se presentan los principales artefactos propuestos en la fase de modelación en la metodología de desarrollo a aplicar (en nuestro caso proceso de desarrollo) y se fundamentan los patrones de diseño empleados en su elaboración.

2.1 Propuesta de solución

El componente Análisis del sistema automatizado para la Planificación Operativa en las Granjas de la UAM (SPO-UAM) permitirá que el sistema mediante la exposición de un conjunto de gráficas, muestre los resultados de las producciones, su comportamiento por ejercicio, establezca resúmenes de actividades y muestre toda esa información de una manera más sencilla y clara, lo cual es de gran importancia cuando se van a presentar los resultados a un público no experto (campesinos que trabajan la tierra, trabajadores de las cooperativas y granjas).

Algunas de las necesidades a resolver con la propuesta de solución que permiten conocer:

- Cuáles son las actividades que más ingresos y costos están generando.
- Los productos que pueden ser ajustados para disminuir los costos y aumentar los ingresos.
- La ocupación y disponibilidad de la infraestructura (campos, jaulas, paneles).
- Qué producto y en qué fecha se deben obtener según la planificación.
- Que productos deben estar disponibles por fecha para poder ejecutar las producciones planificadas.

Con las funcionalidades de la ficha de costo y los reportes de las producciones se analizarán las variaciones de los costos, las cuales permiten conocer cómo se han comportado o cómo han sido utilizadas las actividades directas o indirectas a la producción, lo cual constituye un valioso instrumento en el campo de la toma de decisiones.

2.2 Modelo conceptual

Un modelo conceptual es una representación de conceptos en un dominio del problema. Muestra asociaciones entre conceptos y atributos de conceptos, además de descomponer el espacio del problema en unidades comprensibles (conceptos). La creación de este modelo contribuye a esclarecer la terminología o nomenclatura del dominio. Se puede ver como un modelo que comunica cuáles son los términos importantes y cómo se relacionan entre sí (Pressman, 2010).

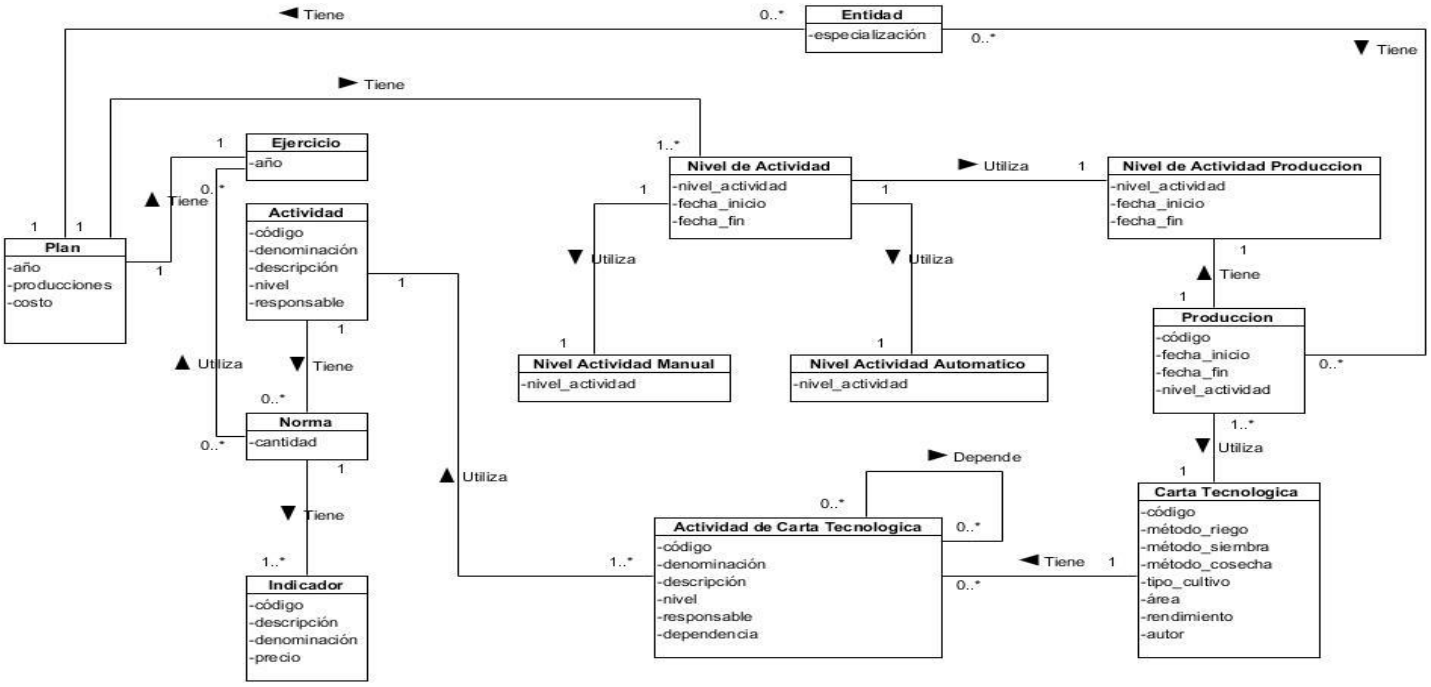


Figura 2.2-1Modelo conceptual.

Tabla 2.2-1Conceptos.

Concepto	**Descripción**
Entidad	Se refiere una entidad de la producción agropecuaria (empresas, granjas).
Carta tecnológica	Se refiere a un documento que define las labores requeridas para cada cultivo, el orden de su realización, el tiempo y los aseguramientos que demanda cada una, lo cual permite

	determinar el volumen de las necesidades de insumos, equipos, implementos y fuerza de trabajo.
Producciones	Se refiere a la utilización de una carta tecnológica como modelo pero enmarcándola en fecha de inicio y fecha fin
Actividad	Se refiere a un conjunto de acciones planificadas llevadas a cabo por personas en un momento dado.
Actividad de Carta tecnológica.	Se refiere a actividades pero asociadas a una carta tecnológica.
Indicador	Se refiere al conjunto de medios físicos existentes para el trabajo en las actividades y entidades.
Nivel de actividad	Se refiere a la cantidad de Actividades total.
Nivel de actividad de producción	Se refiere a la cantidad de Actividades de la carta tecnológica.
Nivel de actividad manual	Se refiere a la cantidad de Actividades que el usuario puede incluir a decisión propia.
Nivel de actividad automático	Se refiere a la cantidad de Actividades que se calculan a partir de la norma de las actividades.
Norma	Se refiere a la norma de consumo y de tiempo de una actividad.
Ejercicio	Se refiere al año para la planificación.
Plan	Se refiere a las producciones propuesta para una entidad a realizar en un ejercicio.

2.3 Técnicas empleadas para la captura de requisitos

Un requerimiento puede definirse como un atributo necesario dentro de un sistema, que puede representar una capacidad, una característica o un factor de calidad del sistema de tal manera que le sea útil a los clientes o a los usuarios finales.

El propósito de la definición de requisitos es especificar las condiciones o capacidades que el sistema debe cumplir y las restricciones bajo las cuales debe operar, logrando un entendimiento entre el equipo de desarrollo y el cliente, y especificándolas necesidades reales de forma que satisfaga sus expectativas (CARRIZO, 2013).

Ambas definiciones se aplican en la presente investigación. A continuación, se presentan las técnicas de captura de requisitos utilizadas para el desarrollo de la solución:

- **Entrevistas:** son realizadas a especialistas de la Unión Agropecuaria Militar (Cliente), en este proceso se realizaron preguntas con el objetivo de obtener toda la información posible sobre la visión que el entrevistado tiene y comprender los propósitos de la solución buscada. Esta técnica generó un documento que contiene la lista de requisitos funcionales con una breve descripción y su complejidad respectivamente (CARRIZO, 2013).
- **Tormenta de ideas:** Estas reuniones se realizaron con todos los involucrados en la solución del problema, donde cada uno expresó sus ideas y criterios respecto a los requisitos que debe poseer la solución propuesta. Tiene como objetivo fundamental dar una visión general de las necesidades del sistema (CARRIZO, 2013).

2.4 Requerimientos funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir una vez estudiado el problema y los posibles procesos a informatizar, de forma tal que se defina cómo el sistema o producto final se ajustará mejor a las necesidades del negocio y cómo va a ser utilizado éste por los usuarios. A continuación, se muestra un listado de los requisitos identificados y que el sistema deberá cumplir (PRESSMAN, et al., 1988).

Desde Carta Tecnológica

Tabla 2.4-1Requerimientos funcionales

ID	Nombre	Descripción	Prioridad	Complejidad
RF_1	Definir actividad como día cero	Se define actividad como día cero y se calcula el plazo de inicio para todas las actividades de la carta tecnológica	Alta	Alta
RF_2	Adicionar dependencia entre las actividades en la carta tecnológica	Se adicionan relaciones de dependencia entre una actividad con una u otras actividades de la carta tecnológica seleccionada	Alta	Alta
RF_3	Listar dependencia de las actividades en la carta tecnológica	Se muestran las dependencias entre la actividad seleccionada y las asociadas a la misma	Alta	Media
RF_4	Eliminar dependencia entre las actividades en la carta tecnológica	Se elimina una o varias dependencias entre la actividad seleccionada y las asociadas a la misma	Media	Baja
RF_5	Exportar listado de cartas tecnológicas	Se exporta un Excel con el registro de las cartas tecnológicas	Baja	Alta
RF_6	Exportar listado de cartas tecnológicas con sus actividades	Se exporta un Excel con el registro de todas las cartas tecnológicas con sus actividades asociadas	Baja	Alta

RF_7	Exportar ficha de costo de carta tecnológica	Se exporta un Excel o un PDF con el registro de la ficha de costo asociada a la carta tecnológica seleccionada	Baja	Alta
RF_8	Exportar carta tecnológica	Se exporta un Excel con el registro de los importes de las actividades asociadas a la carta tecnológica seleccionada	Baja	Alta
RF_9	Exportar indicadores de una carta tecnológica	Se exporta un Excel con el registro de los importes de los indicadores de las actividades asociadas a la carta tecnológica seleccionada	Baja	Alta

Desde Producciones

Tabla 2.44-2Requerimientos funcionales desde producciones

ID	Nombre	Descripción	Prioridad	Complejidad
RF_10	Exportar listado de producciones	Se exporta un Excel con el registro de las producciones	Baja	Alta
RF_11	Exportar actividades de una producción	Se exporta un Excel con el registro de los importes de las actividades asociadas a la producción seleccionada	Baja	Alta
RF_12	Exportar indicadores de una producción	Se exporta un Excel con el registro de los importes de los indicadores de las actividades asociadas a la producción seleccionada	Baja	Alta

Desde Análisis

Tabla 2.44-3Requerimientos funcionales desde análisis

ID	Nombre	Descripción	Prioridad	Complejidad
RF_13	Listar estructura	Se listan las estructuras	Alta	Alta
RF_14	Limpiar campo de búsqueda	Se limpia el campo de búsqueda y se recarga el árbol de estructuras	Media	Baja
RF_15	Buscar estructura	Se muestran las estructuras resultado de la búsqueda realizada por el criterio de búsqueda	Alta	Alta
RF_16	Contraer árbol de estructura	Se contrae el árbol de estructura	Media	Baja
RF_17	Expandir árbol de estructura	Se expande el árbol de estructura	Media	Baja
RF_18	Visualizar actividades-niveles de actividad por ejercicio	Se visualizan las gráficas que muestran la relación entre actividades y niveles de actividades por ejercicio de la entidad seleccionada	Media	Media
RF_19	Visualizar niveles de actividad por ejercicio	Se visualizan las gráficas que muestran los niveles de actividades por ejercicio de la entidad seleccionada	Media	Media
RF_20	Visualizar resumen de gráficas de niveles de actividad por ejercicio	Se visualizan las gráficas que muestran el resumen de niveles de actividades por ejercicio de la entidad seleccionada	Media	Alta
RF_21	Visualizar plan-normados y directos por ejercicio	Se visualizan las gráficas que muestran el plan normado y directo por ejercicio de la entidad seleccionada	Media	Alta

RF_22	Visualizar plan-ingresos-gastos por ejercicio	Se visualizan las gráficas que muestran la relación plan, ingresos, gastos por ejercicio de la entidad seleccionada	Media	Alta
RF_23	Visualizar plan-indicador-importe por ejercicio	Se visualizan las gráficas que muestran la relación plan, indicador, importe por ejercicio de la entidad seleccionada	Media	Alta
RF_24	Visualizar plan-resumen por centro de balance por ejercicio	Se visualizan las gráficas que muestran el plan resumen por centro de balance por ejercicio de la entidad seleccionada	Media	Alta

2.5 Requerimientos no funcionales

A continuación, se describen los requisitos no funcionales que establecen las características del sistema definido por la dirección de la entidad de planificación de la empresa XETID y aprobadas por la Unión Agropecuaria Militar, los cuales son listados a continuación:

Tabla 2.55-1Requerimientos no funcionales

ID	Categoría	Descripción
RNF_1	Usabilidad	El sistema debe ser fácil de utilizar para los usuarios que tengan niveles básicos de computación o hayan trabajado con la Web.
RNF_2	Usabilidad	Debe tener una opción de ayuda sobre las principales funcionalidades que brinda el sistema y sus iconos respectivos, para un mejor entendimiento.
RNF_3	Usabilidad	Las operaciones de la aplicación a informatizar serán lo más parecidas posible a los procesos que se realizan actualmente en la UAM, para así lograr el menor tiempo en cuanto a la comprensión y adaptación del sistema.
RNF_4	Rendimiento	La aplicación debe estar concebida para el consumo mínimo de recursos.
RNF_5	Rendimiento	La aplicación debe estar concebida para que se conecten 400 usuarios como máximo, simultáneamente conectados.
RNF_6	Rendimiento	Los tiempos de respuesta y velocidad de procesamiento de la información serán rápidos, no mayores de 5 segundos para las actualizaciones y 20 para las recuperaciones.

RNF_7	Soporte	Se necesita un servidor de bases de datos que soporte grandes volúmenes de datos.
RNF_8	Soporte	Debe elaborarse un paquete de instalación que abarque verificación de componentes ya instalados y la instalación de los nuevos.
RNF_9	Requerimiento de Ayuda y Documentación	Se propone que el sistema cuente con una ayuda general en la página principal, que guiará al usuario de cómo trabajar en el sistema, también estará disponible en cada una de las interfaces, de esta forma los usuarios tendrán conocimiento de las funcionalidades del mismo y hacer un mejor uso de estas.
RNF_10	Interfaz	La interfaz de la aplicación a desarrollar debe ser sencilla para reducir el tiempo de capacitación de los usuarios.
RNF_11	Interfaz	Por el uso diario y constante que tendrá el software, la interfaz debe ser agradable, que favorezca el estado de ánimo del cliente y que combine correctamente los colores, tipo de letra, tamaño y que los iconos estén en correspondencia con lo que representan.
RNF_12	Políticos Culturales	El producto no debe contener palabras en otros idiomas y debe respetar los términos empleados por los especialistas en el tema de la esfera que se automatiza.
RNF_13	Software	El servidor del debe tener como requerimientos mínimos de software Sistemas operativos Linux o Windows 2000 o superior.
RNF_14	Software	El servidor debe tener como requerimientos mínimos de

		software Servidor Web Apache 2.0 o superior, con módulo PHP 5.
RNF_15	Software	El servidor debe tener como requerimientos mínimos de software PostgreSQL8.3.9 como Sistema Gestor de Base de Datos.
RNF_16	Software	Para las PCs clientes se recomienda Mozilla Firefox 30 o superior, recomendado 48.
RNF_17	Software	Para las PCs clientes soporte para formato PDF**.**
RNF_18	Software	Para las PCs clientes debe tener soporte para hojas de Excel.
RNF_19	Hardware	Para las PC clientes Procesador: 3.0 GHz o superior.
RNF_20	Hardware	Para las PC clientes RAM: 512 o superior.
RNF_21	Hardware	Para los servidores Procesador: Doble Micro a 3.0 GHz o superior.
RNF_22	Hardware	Para los servidores RAM: 4GB o superior.
RNF_23	Hardware	Para los servidores Disco duro: 300 GB o superior.
RNF_24	Hardware	Para los servidores UPS: 1.
RNF_25	Hardware	Para los servidores Tarjeta de Red: 1 Gb.
RNF_26	Hardware	Del lado del servidor de Base de datos Procesador: Doble Micro a 3.0 GHz o superior.
RNF_27	Hardware	Del lado del servidor de Base de datos RAM: 4GB o superior.

RNF_28	Hardware	Del lado del servidor de Base de datos Disco duro: 1 TB o superior.
RNF_29	Hardware	Del lado del servidor de Base de datos UPS: 1.
RNF_30	Hardware	Del lado del servidor de Base de datos Tarjeta de Red: 1 Gb.

2.6 Arquitectura del sistema

La arquitectura de software conforma el esqueleto de cualquier sistema, y es la principal responsable de los atributos de calidad del sistema (parte de los requerimientos no funcionales del mismo) la misma identifica los elementos más importantes de un sistema, así como sus relaciones, es decir, da una visión global del sistema. (S. Pressma, 2010)

Dentro de la arquitectura del sistema se desarrollan varios procesos como son: diseño de la arquitectura tecnológica, diseño de la arquitectura de seguridad, diseño de la arquitectura de despliegue, diseño de la arquitectura de sistema y diseño de la arquitectura de datos, la misma es basada en componentes permitiendo una mayor reutilización de estos. A partir de estos procesos se definen términos fundamentales para el desarrollo de la solución propuesta, ya que se generan artefactos de gran valor como son: el diagrama de componentes que permite identificar dependencias entre varios componentes y las posibles integraciones a realizar y también se obtiene el modelo de datos que permite definir el tamaño y el correcto diseño de la base de datos, siendo este uno de los puntos claves para el futuro desarrollo de la aplicación (Pressman, 2010).

2.6.1 Patrón MVC

El patrón Modelo Vista Controlador (MVC) define responsabilidades y dependencias obedeciendo a objetivos específicos representados por tres paradigmas. Separa los datos de la aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos, permitiendo mayor independencia, mantenimiento y reutilización (LARMAN, 2003).

- **Modelo:** Representa la lógica del negocio de la aplicación. Su finalidad es servir de abstracción de la lógica relacionada con los datos o de algún proceso en el mundo real. Los datos conocidos por el modelo son manipulados por métodos que son totalmente independientes de las interfaces gráficas de usuarios (LARMAN, 2003). Se evidencia en las clases *analisisplanificacion, datactividadentidad*

y *datentidaddatosnomenclador,* las mismas manejan los datos que serán visualizados en las gráficas que propone la propuesta de solución.

- **Vista**: Es el objeto que maneja la información visual que es presentada al usuario, representa los datos proyectados por el Modelo y envía los datos y los eventos ejecutados por el usuario hacia el Controlador (LARMAN, 2003). Se evidencia con los ficheros *Analisisplanificacion.phtml* y *analisisplanificcion.view.js,* los mismos son los encargados de mostrar la vista de la propuesta de solución.
- **Controlador:** Es quien se encarga de verificar las operaciones que los usuarios solicitan realizar tanto para el modelo como para la vista. Interactúa con el Modelo a través de una referencia al mismo (LARMAN, 2003). Se evidencia en la clase *AnalisisplanificacionController*, la misma funciona de puente entre la vista y el modelo recogiendo las peticiones del usuario capturadas desde la vista enviándola al modelo, luego reenvía la respuesta del modelo hacia la vista.

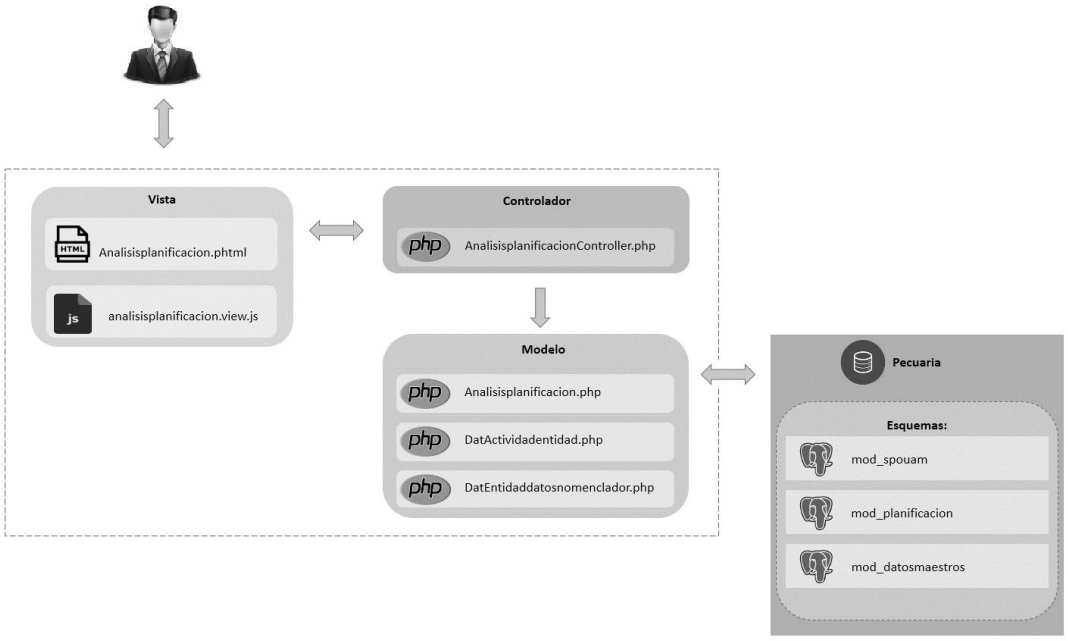


Figura 2.66.1Arquitectura modelo vista controlador (MVC) aplicada a la solución.

2.7 Diseño del sistema

En el diseño del sistema se realiza una modelación de los artefactos correspondientes, entre los cuales se encuentra el modelo de diseño, mediante diagramas de clases del diseño con estereotipos web (LARMAN, 2003). A continuación, se observa el diagrama de clases del diseño con estereotipos web: Exportar cartas. Los restantes se pueden consultar en el **Anexo 1**.

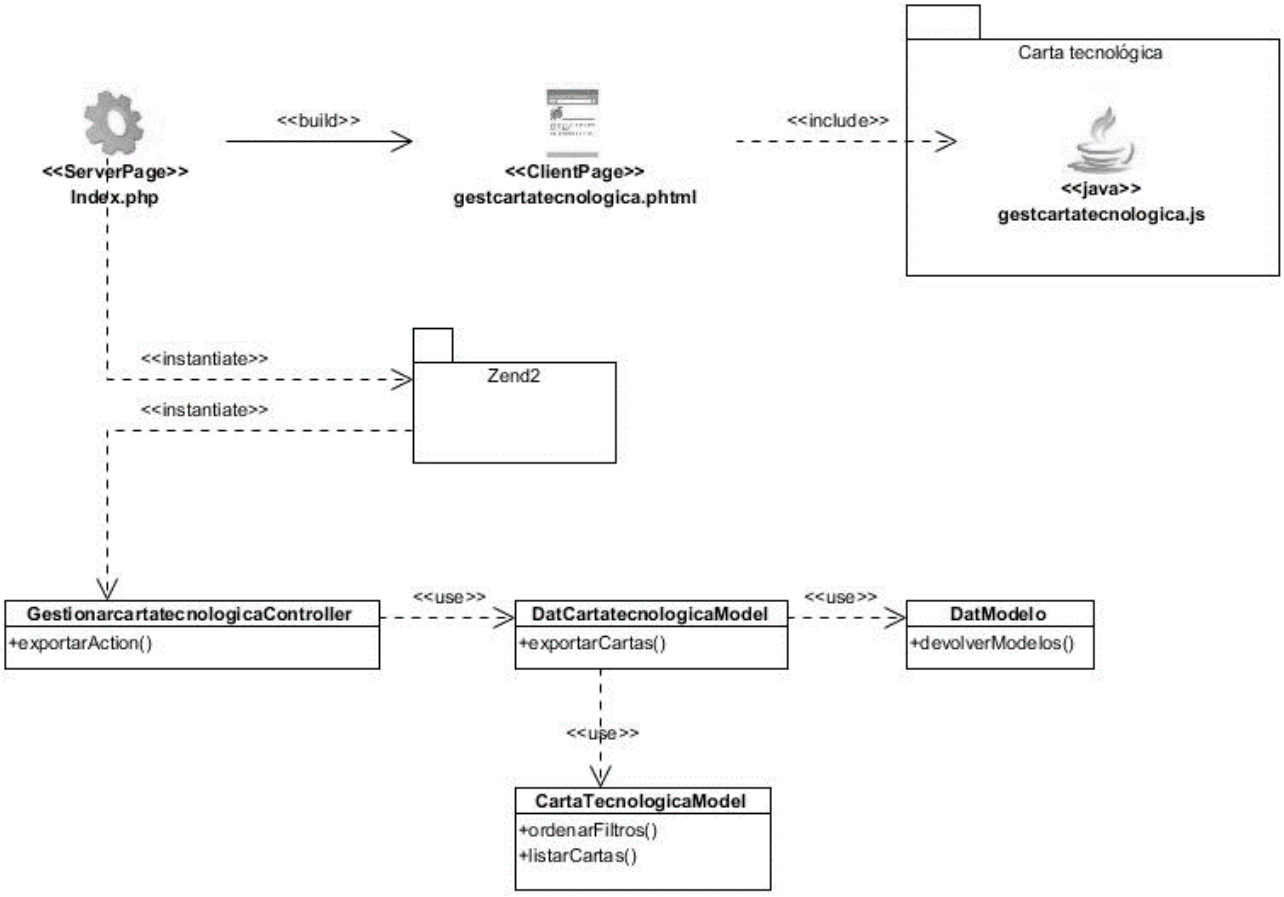


Figura 2.7-1Diagrama de clases del diseño con estereotipos web para Exportar cartas.

2.7.1 Diagramas de secuencia del diseño

Ilustran las interacciones en un tipo de formato con aspecto de una valla, en el que cada objeto nuevo se añade a la derecha, mostrando claramente la secuencia y ordenación en el tiempo de los mensajes con una notación simple (LARMAN, 2003). A continuación, se observa el diagrama de secuencia: Exportar cartas. Los restantes se pueden consultar en el **Anexo 3**.

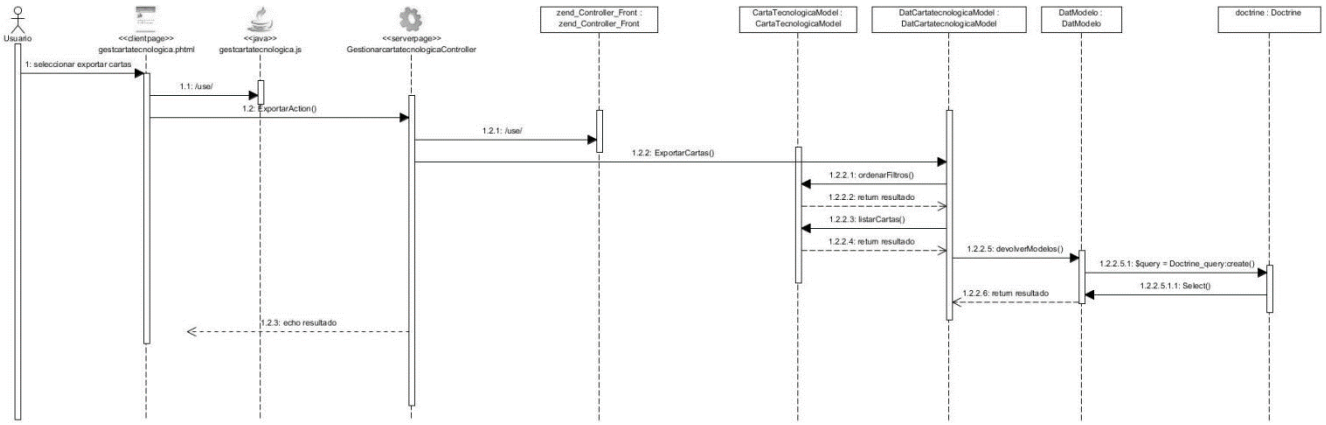


Figura 2.7-2Diagrama de secuencia del diseño para Exportar cartas.

2.7.2 Patrones de diseño

Los patrones de diseño ofrecen soluciones efectivas y reconocidas a problemas comunes de arquitectura/desarrollo de software. Estos se utilizan para estandarizar la comunicación entre otros colegas programadores, proporcionando un vocabulario común a usar para discutir soluciones a problemas simples o complejos (S. Pressma, 2010).

Patrones generales de software para asignar responsabilidades (GRASP)

Experto: Se encarga de asignar la responsabilidad al experto en la información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. Permite conservar el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide, lo que provee un bajo nivel de acoplamiento. Promueve clases sencillas y cohesivas que son más fáciles de mantener y comprender. Solo la clase DatCartatecnologicaactividad es encargada de gestionar las actividades por lo cual en ella se evidencia este patrón (S. Pressma, 2010).

Creador: Consiste en asignar a un Objeto la responsabilidad de crear otro Objeto. Un objeto es responsable de crear una nueva instancia de alguna clase si: agrega o contiene objetos de ella, registra las instancias de sus objetos o tiene los datos de inicialización que serán enviados a ella cuando el objeto sea creado. La utilización de este patrón se evidencia desde la clase AnalisisplanificacionModel en la cual se crea un objeto y se llama a una funcionalidad de ese objeto en la cual se crea otro objeto (S. Pressma, 2010).

```
class AnalisisplanificacionModel extends ZendExt_Model {
//Actividad - Niveles de actividad
    public function actNivelesActividad($params) {
```

```
        $analisisPlanif = new Analisisplanificacion();
        ...
        $tot = $analisisPlanif->diferenciaNormaSalidaTotal($ent,
$params['idejercicio'], $params);
        ...
    }
}
class Analisisplanificacion {
...
    public function diferenciaNormaSalidaTotal($identidad, $idejercicio,
$parametros) {
        $conn = Doctrine_Manager::connection();
        $result = $conn->execute($sqlQuery);
        $data = $result->fetchAll(PDO::FETCH_COLUMN);
        return $data;
    }
...
}
```

Fragmento de código 2.7.1: clase Analisisplanificacionmodel.

Controlador: El patrón ofrece una guía para tomar decisiones sobre los eventos de entrada, asignando la responsabilidad del manejo de mensajes de los eventos del sistema a una clase controladora, ya que los elementos de interfaz y sus controladores de eventos, no deben ser responsables de controlar los eventos del sistema. La utilización de este patrón se evidencia en la utilización de la clase controladora analisisPlanificacionController (S. Pressma, 2010).

Bajo acoplamiento: El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases. Este patrón da soporte a una mínima dependencia y a un aumento de la reutilización; una clase con bajo acoplamiento no depende de “muchas otras” clases para realizar sus tareas, permitiendo que se pueda reutilizar con mayor facilidad y flexibilidad. La utilización de este patrón se evidencia en la clase Analisisplanificacion que no extiende de otra clase y solo utiliza la clase NomActividad en su constructor para instanciar objetos de ella y no necesita de más relaciones para ejecutar sus funciones (S. Pressma, 2010).

```
class Analisisplanificacion {

    private $nivel;
```

```
    public function __construct() {
        $this->nivel = NomActividad::getNivel();
    }
    //Actividad - Niveles de actividad
    public function diferenciaNormaSalida($identidad, $idejercicio, $orden,
$start, $limit, $parametros) {
        //implementando filtros
        $filt = self::filtrarNA_addwhere($parametros);
        $filtros = $filt[0];
        ...
```

Fragmento de código 2.7.2: clase Analisisplanificacion.

Alta cohesión: La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realizan un trabajo enorme. Fomenta la reutilización, mejorando la claridad y facilidad del diseño. La utilización de este patrón se evidencia en la utilización de una sola clase controladora AnalisisplanificacionController que se encargue de asignar responsabilidades por funciones (S. Pressma, 2010).

```
class AnalisisplanificacionController extends ZendExt_Controller_Secure {
...
    public function analisisplanificacionAction() {
        $jsestructura = $this->integrator->metadatos->obtenerIUEstructura();
        $this->view->jsestructura = $jsestructura;
        $this->render();
    }
    //Datos para graficar el tab Plan - Más demandados
    public function graficarPlanMasdemandadosAction() {
        $modelAnalisisplanificacion = new AnalisisplanificacionModel();
        echo json_encode($modelAnalisisplanificacion->PlanMasdemandados($this-
>_request->getPost()));
    }
    //Datos para graficar el tab Plan - Más valiosos
    public function graficarPlanMasvaliososAction() {
        $modelAnalisisplanificacion = new AnalisisplanificacionModel();
        echo json_encode($modelAnalisisplanificacion->PlanMasvaliosos($this-
>_request->getPost()));
    }
...
```

Fragmento de código 2.7.3: clase AnalisisplanificacionController.

Patrones Gang of Four (GOF)

Puente(bridge): La capa de abstracción de base de datos se implementa de una forma similar al patrón bridge (Pressman, 2010), Ejemplo de estos se crean clases que garantizan la conexión con doctrine para posteriormente realizar consultas a la base de datos sin importar de que tipo sea esta. La utilización de este patrón se evidencia con la clase BaseDatEntidaddatosnomenclador.

```
abstract class BaseDatEntidaddatosnomenclador extends Doctrine_Record
{
    public function setTableDefinition()
    {
        $this->setTableName('dat_entidaddatosnomenclador');
        $this->hasColumn(
            'identidaddatosnomenclador', 'numeric', null,
            array('notnull' => true, 'primary' => true,
                'sequence' => 'mod_planificacion.sec_entidaddatosnomenclador')
        );
        $this->hasColumn(
            'identidad', 'numeric', null,
            array('notnull' => true, 'primary' => false)
        );
        $this->hasColumn(
            'iddatosnomenclador', 'numeric', null,
            array('notnull' => true, 'primary' => false)
        );
    }
    public function Setup()
    {
        parent::setUp();
    }
}
```

Fragmento de código 2.7.4: clase BaseDatEntidaddatosnomenclador.

2.8 Diseño de componentes

Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias que existen entre estos. En los mismos se ilustran las piezas del software, controladores embebidos, etc. que conformarán un sistema (S. Pressma, 2010).

A continuación, se muestran los diagramas de componentes para la solución propuesta del Sistema de Planificación Operativa de la Unión Agropecuaria Militar.

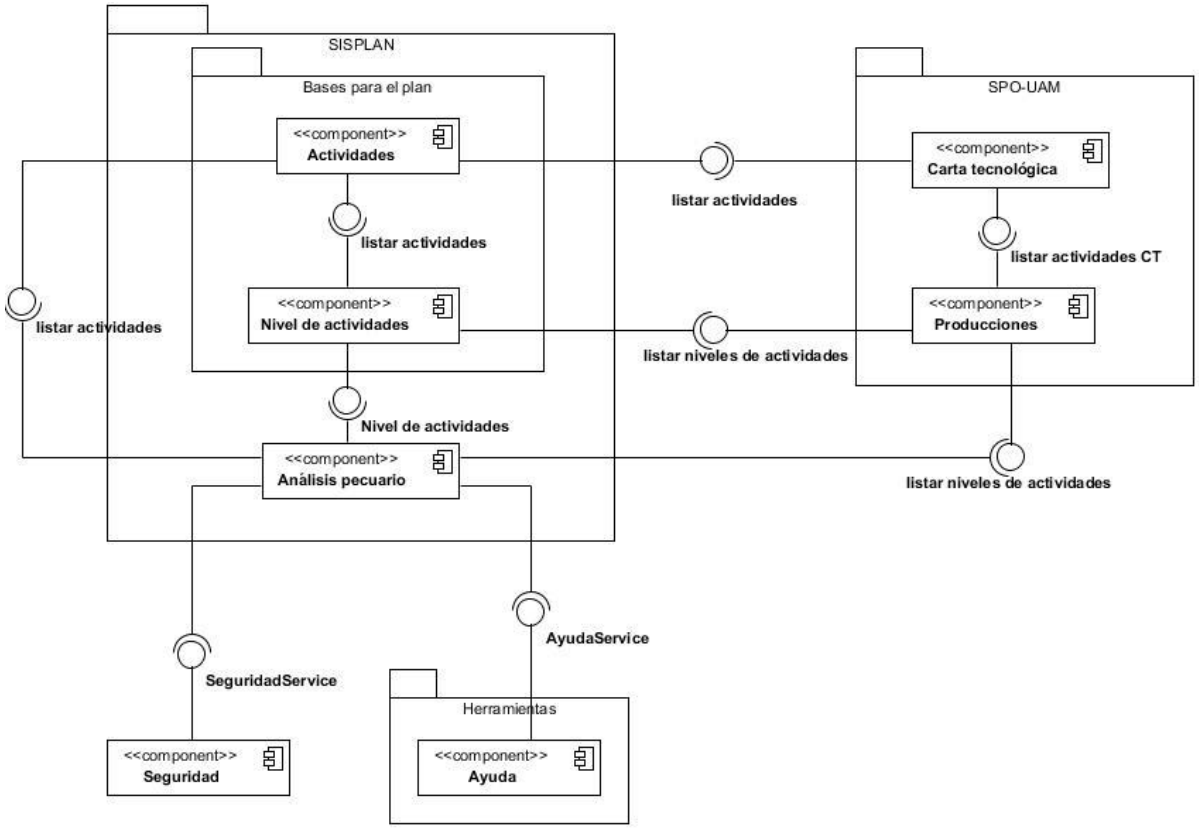


Figura 2.8-1Diagrama de diseño de componentes externo.

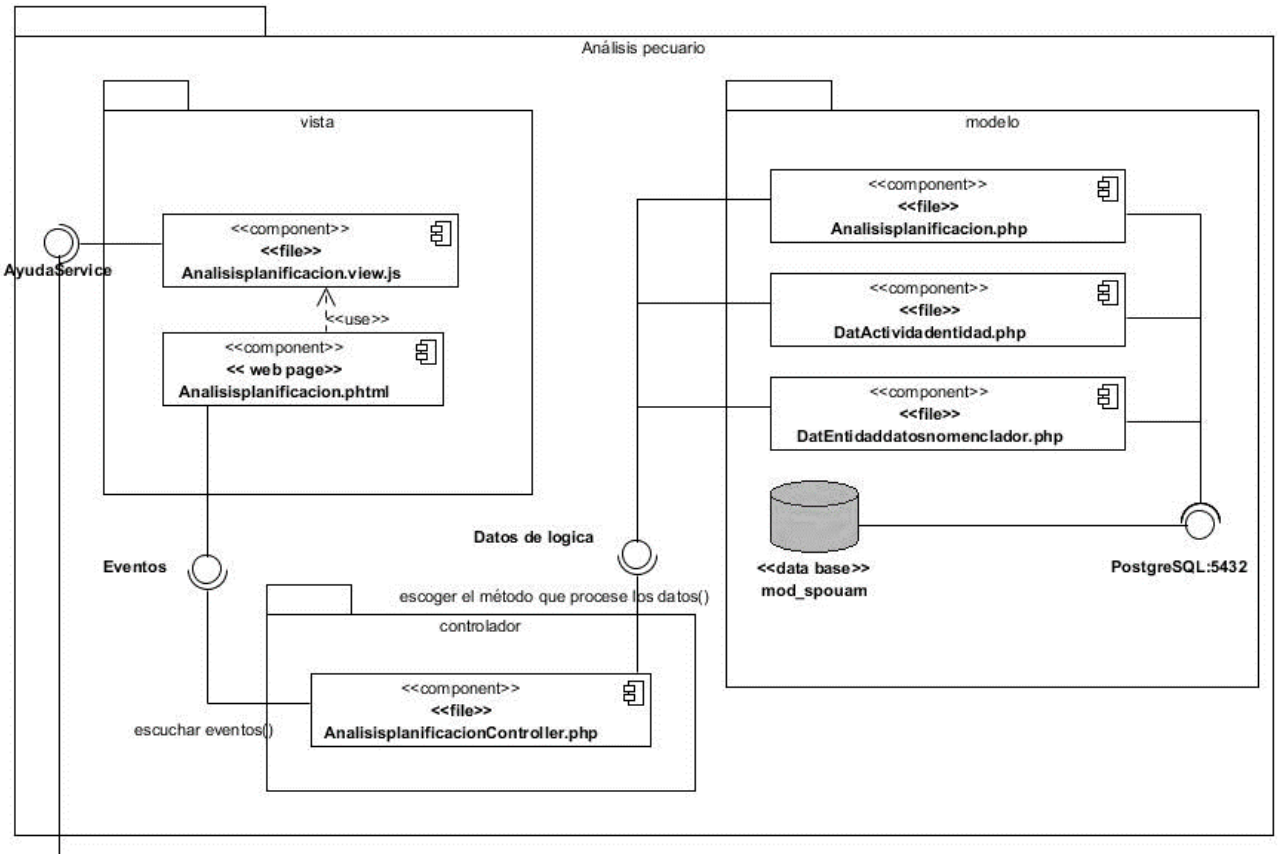


Figura 2.8-2Diagrama de diseño de componentes interno

Se listan todos los componentes definidos en el Sistema de Planificación Operativa de la Unión Agropecuaria Militar y de cada uno de ellos se especificará el nombre del componente, una descripción basada en el objetivo y principales características del componente, los servicios que provee ese componente con sus detalles.

Tabla 2.8-1Especificación de componentes.

Componentes	**Descripción**
Actividades	Se refiere a un conjunto de acciones planificadas llevadas a cabo por personas en un momento dado.

Nivel de Actividades	Permite cuantificar la cantidad de Actividades de la Producción.
Carta tecnológica	Permite gestionar un documento que define las labores requeridas para cada cultivo, el orden de su realización, el tiempo y los aseguramientos que demanda cada una, lo cual posibilita determinar el volumen de las necesidades de insumos, equipos, implementos y fuerza de trabajo.
Producciones	Permite Establecer producciones utilizando las cartas tecnológicas.
Análisis pecuario	Permite interpretar la información mediante gráficas del Sistema base.

2.9 Modelo de datos

Un modelo de datos es un lenguaje orientado a describir una Base de Datos. Típicamente un modelo de datos permite describir las estructuras de datos que representan a los elementos del dominio, el tipo de dato de los atributos y la forma en que se relacionan (S. Pressma, 2010).

El modelo de datos para la solución propuesta es el ya existente incorporando la tabla *dat_dependenciasactividad* para un total de 15 tablas, en la cual se consideran las tablas *dat_dependenciasactividad*, *dat_cartatecnologicaactividad* y *dat_cartatecnologica* como tablas críticas del sistema atendiendo a que la mismas, son las que más crecen y a su vez en las que más acciones de *UPDATE*, *INSERT*, *DELETE* y *SELECT* se realizan.

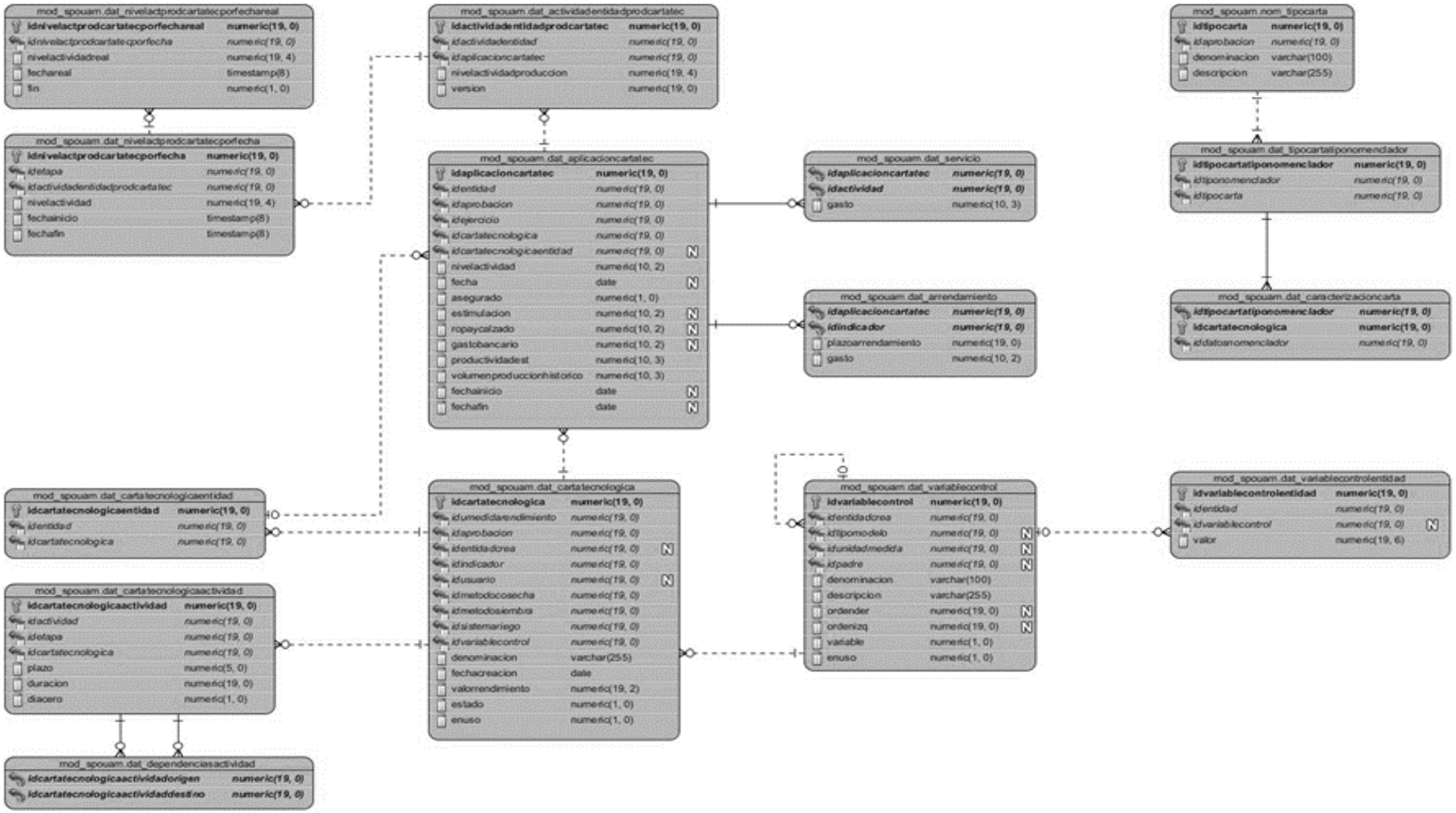


Figura 2.9-1Modelo de dato

2.10 Conclusiones del capítulo

La obtención de los elementos y artefactos de diseño y arquitectura, permitió proyectar una visión más acabada de cuáles son las prestaciones con las que se debe contar, cómo debe funcionar el componente y su integración con los restantes del sistema, las clases que lo componen y los métodos y propiedades que deben de implementarse.

Capítulo III: Implementación y prueba del Desarrollo del componente Análisis del Sistema de Planificación de Producciones.

En el presente capítulo se abordan los aspectos fundamentales de la fase de Construcción, dentro de la cual se destaca el proceso de implementación y el de prueba de software. La implementación del componente de análisis comprende el desarrollo e integración de artefactos que deberán ser sometidos a diferentes pruebas de calidad de software para garantizar que estos funcionen correctamente.

3.1 Diagrama de despliegue

Un diagrama de despliegue muestra cómo y dónde se desplegará el sistema. Las máquinas físicas y los procesadores se representan como nodos, y la construcción interna puede ser representada por nodos o artefactos embebidos. Como los artefactos se ubican en los nodos para modelar el despliegue del sistema, la ubicación es guiada por el uso de las especificaciones de despliegue (ZARAGOZA , et al., 2005).

Los Diagramas de Despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. Existen diferentes tipos de despliegues entre ellos se encuentra el distribuido, el centralizado y el mixto (ZARAGOZA , et al., 2005).

Despliegue distribuido: La aplicación se instala en los diferentes servidores teniendo en cuenta el principio de territorialidad. Cada lugar puede acceder a su sistema y si hay un problema en un servidor fuera de su área, no le afecta (ZARAGOZA , et al., 2005).

Despliegue centralizado: Toda la aplicación está instalada en un servidor único y a ella acceden todos los usuarios (ZARAGOZA , et al., 2005). El componente de Análisis del Sistema de Planificación de Producciones es centralizado ya que se ha desplegado en los servidores con que cuenta la Unión Agropecuaria Militar al que se accederá desde diferentes granjas agropecuarias a lo largo del país. El presente diagrama de despliegue muestra una PC cliente que se conecta a un servidor de aplicaciones (mediante el protocolo de hipertexto HTTP o HTTPS) y éste establece una conexión con el servidor de base de datos (mediante TCP/IP) ver **Figura 3.2-1.**

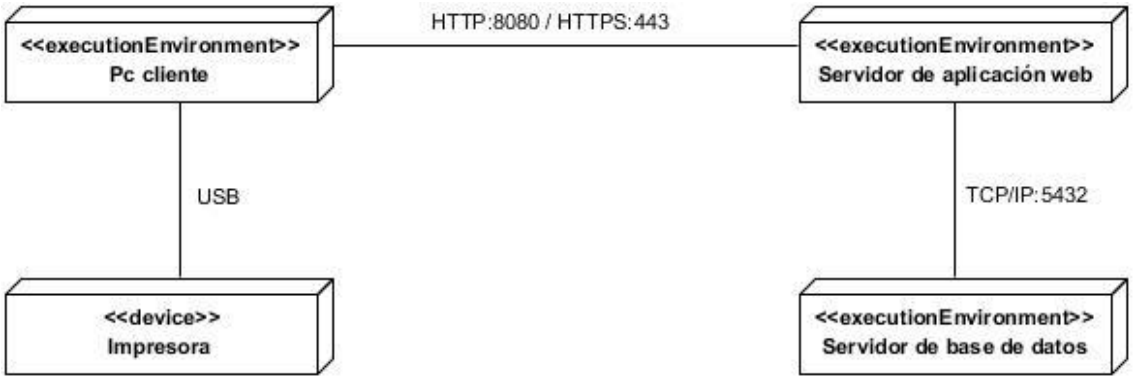


Figura 3.1-1Diagrama de despliegue.

3.2 Implementación

La implementación del sistema es un proceso que comienza con el resultado obtenido del diseño detallado. Esta tiene como objetivo principal desarrollar la arquitectura y el sistema como un todo (Curbelo Oliva, et al., 2008). Dentro de los propósitos más específicos de la implementación se encuentran:

- ✓ La planificación de las integraciones del sistema necesarias para cada iteración.
- ✓ Implementar las clases, componentes y subsistemas encontrados durante el diseño.
- ✓ Integrar los componentes.

En el proceso de implementación se conserva la estandarización y reutilización de códigos y componentes. Este proceso obtendrá como salida los ficheros de código fuente y el ejecutable, los cuales formarán parte de la Versión beta del producto (Curbelo Oliva, et al., 2008).

3.2.1 Estándares de codificación

Los estándares de codificación tienen como objetivo establecer estilos de códigos sólidos con vistas de que un proyecto de software se convierta en un producto fácil de comprender y de mantener (Rodríguez López, 2013).

En la implementación de la propuesta de solución se utilizaron los estándares definidos en el documento (Estándar de codificación para PHP Versión 2.0.0), elaborado por el Comité de rol de Lógica de Negocio de la XETID. Entre los estándares explicados se encuentran los relacionados con:

- Formato de los archivos PHP.
- Convenciones de nombre.

- Estilo de código.

Entre los estándares más significativos utilizados en los proyectos de software desarrollados por XETID se encuentran los siguientes:

Comentarios

Se incluirán como apoyo a variables y llamadas a funciones.

Nombre de identificadores

Se considera como identificador a los nombres de variables, funciones, así como cualquier tipo de dato definido por el usuario (estructura, clase). Dichos identificadores deberán seguir las siguientes normas:

- Tener un nombre significativo para que, por su simple lectura, pueda conocerse su función, sin tener que consultar manuales o hacer demasiados comentarios.
- Los nombres deben estar en el idioma español y no deben contener caracteres extraños, excluyendo el guion bajo.
- Para nombres que se usen con frecuencia o para términos largos, se recomienda usar abreviaturas estándar para que estos tengan una longitud razonable.

Identificadores de variables

Para distinguir palabras dentro del nombre deberá emplearse un guion bajo (_). Preferiblemente no usar más de 3 palabras por identificador siempre y cuando en el nombre quede implícita la acción. Ejemplo: nueva_variable.

Identificadores de clases y métodos

- Los identificadores de clases comenzarán escritos con letra mayúscula. En caso de tener más de una palabra, la segunda comenzará de nuevo con mayúscula.
- Los identificadores de los métodos comenzarán en letra minúscula. En caso de tener más de una palabra, la segunda comenzará con mayúscula.

Ejemplo clase: NuevaClase, **ejemplo método**: nuevaFuncion.

Líneas, sangrías y espacios en blanco

- No manejar en los programas más de una instrucción por línea.
- Declarar variables del mismo tipo en una línea.

- Evitar líneas de más de 100 caracteres.
- La unidad de la sangría es de cuatro espacios
- Insertar una línea en blanco antes y después de una declaración de datos que aparezca entre instrucciones ejecutables.
- Las declaraciones de datos dentro de una función, deberán ir al inicio y separadas de las instrucciones ejecutables de la función por medio de una línea en blanco.

Las vistas

En la **Figura 3.2.1** se muestra la estructura de carpetas de la interfaz de usuario del componente Análisis del Sistema de Planificación de Producciones. Para este componente los archivos más importantes son los que se encuentran en el directorio producciones/views/js, los cuales se ajustan a un estilo Modelo – Vista – Controlador.

Clases controladoras

En la **Figura 3.2-2** se muestra la estructura de carpetas donde se encuentra la clase controladora del componente Análisis del Sistema de Planificación de Producciones, específicamente en el directorio *controllers*, tomando como archivo:

AnalisisplanificacionsisplanproduccionesController.php.

3.3 Pruebas

Las pruebas de software se definen como una actividad en la cual un sistema o uno de sus componentes se ejecutan en circunstancias previamente especificadas, los resultados se observan y registran, y se realiza una evaluación de algún aspecto. La prueba es un elemento crítico para la calidad del software. La importancia de los costos asociados a los errores promueve la definición y aplicación de un proceso de pruebas minuciosas y bien planificadas. Las pruebas permiten validar y verificar el software, entendiendo como validación del software el proceso que determina si el software satisface los requisitos, y como verificación, el proceso que determina si los productos de una fase satisfacen las condiciones de dicha fase (PRESSMAN, et al., 1988).

Tal y como lo define el Proceso de desarrollo PRODESOFT, las pruebas tienen el objetivo principal de asegurar que el software cumpla con las especificaciones requeridas y eliminar los posibles defectos que este pudiera tener. De forma más específica los propósitos de las pruebas son:

- ✓ Realizar la validación del software desarrollado.

- ✓ Realizar la verificación del software desarrollado.

De las pruebas de software surgen las no conformidades (NC), las cuales deberán de ser analizadas por el equipo de desarrollo de la aplicación para darle la mejor solución.

Descripción de las pruebas utilizadas

Las pruebas aplicadas al componente Análisis son: **la prueba de unidad, la prueba de integración y la prueba de aceptación.**

3.3.1 Prueba de unidad

La prueba de unidad se enfoca a los elementos testeables más pequeño del software, estos pueden ser clases, métodos, propiedades, componentes, etc. Es aplicable a componentes representados en el modelo de implementación para verificar que los flujos de control y de datos están cubiertos, y que ellos funcionen como se espera. La prueba de unidad siempre está orientada a caja blanca. Antes de iniciar cualquier otra prueba es preciso probar el flujo de datos de la interfaz del componente. Si los datos no entran correctamente, todas las demás pruebas no tienen sentido. El diseño de casos de prueba de una unidad comienza una vez que se ha desarrollado, revisado y verificado en su sintaxis el código a nivel fuente. A este nivel uno de los métodos de prueba más importantes es el método de caja blanca (Pressman, 2010).

Métodos de pruebas

Pruebas de caja blanca

Las pruebas de caja blanca, en ocasiones llamadas pruebas de cristal, es un método de diseño que usa la estructura de control descrita como parte del diseño al nivel de componentes para derivar los casos de pruebas. La prueba de caja blanca del software se basa en un examen cercano al detalle procedimental. Se prueban las vistas lógicas del software y la colaboración entre componentes, al proporcionar casos de pruebas que ejercen conjuntos específicos de condiciones, bucles o ambos (Pressman, 2010).

En el caso de la técnica de las pruebas de caja blanca se le realizaron al componente Análisis. Estas pruebas están dirigidas a las funcionalidades internas del sistema. Se ejecutan probando caminos lógicos por los cuales deberá transitar el sistema una vez puesto en funcionamiento y se observan los resultados arrojados en diferentes puntos; lo que permitió constatar que los resultados son los esperados. Estas pruebas se desarrollan de forma que se asegure que la operación interna se ajusta a las especificaciones, y que todos los componentes internos se han probado de forma adecuada. Dentro

de las pruebas de caja blanca se encuentra la prueba del camino básico. Este método permite que el diseñador de casos de pruebas obtenga una medida de complejidad de un diseño procedimental y que use esta medida como guía para definir un conjunto básico de rutas de ejecución. Los casos de pruebas derivados para ejercitar el conjunto básico deben garantizar que se ejecuta cada instrucción del programa por lo menos durante la prueba (Pressman, 2010).

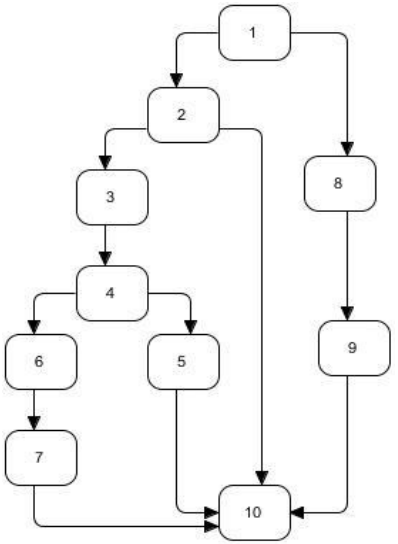
Esta prueba se basa en la obtención de la complejidad lógica de un diseño. El método consiste en la representación mediante un grafo de flujo asociado de todos los posibles caminos que se puedan iniciar en una determinada porción de código y aplicarle a cada uno de estos caminos un caso de prueba (Pressman, 2010). A la representación realizada por el grafo se le calculará la complejidad ciclomática. Los pasos a seguir para la aplicación del método son los siguientes:

- Dada una selección del código fuente se representa un grafo asociado.
- Se calcula la complejidad ciclomática del grafo.
- Se determina un conjunto básico de caminos independientes.
- Se elaboran los casos de pruebas que obligan a la ejecución de cada camino del conjunto básico.

Tabla 3.3-1Diseño del caso de prueba (Técnica del Camino Básico).

<table>
<tr><th colspan="2">Prueba de caja blanca</th></tr>
<tr><td>Código al que se aplica:

public function graficarCalculoPlanesPorEntidades($params) {
if ($params['global'] == 'true') {//1
if ($params['identidad'] == null) {//2
$entid = array();//3
$datos=array();//3
$entid = DatActividadentidad::auxObtenerEntidadesHijasComun($params['identidad']);//3
if ($params[0]['identidaddatosnomenclador'] != null) {//4
$datos[0]['identidad'] = $params[0]['identidaddatosnomenclador'];//5
$entid = $this->integrator->metadatos->DameEstructura($params['idEstructura']);//5
$datos[0]['denominacion'] = $entid[0];//5
} else{//6
$datos = null;//7
}
}
}else{//8
$analisisPlanif = new Analisisplanificacion();//9
$data = $analisisPlanif->calculoPlanesPorEntidad($params);//9
}
return array('data' => $data, 'entidades' => $entid);//10
}</td><td>Representación en grafo de flujo:

 
</td></tr>
<tr><td>Complejidad ciclomática:

[V (G)] = Cantidad de Aristas [A] – Cantidad de nodos [N] + 2.</td><td>Caminos independientes:

1. 1-2-10</td></tr>
</table>

• V (G) = 12 – 10 + 2 • V (G) = 4	2. 1-8-9-10 3. 1-2-3-4-5-10 4. 1-2-3-4-6-7-10

Dando continuidad al método y contando con el grafo y los caminos identificados previamente en la prueba de caja blanca, se pasa a la creación de casos de pruebas para cada uno de estos caminos. La tabla muestra el caso de prueba para el camino 1-2-10, los restantes casos de pruebas se encuentran en los anexos.

Tabla 3.3-2 Caso de prueba para el camino No 1.

Caso de prueba para el camino No 1.	
Camino	1-2-10
Descripción.	• Verifica que el global este en true. • Verifica que la entidad no este vacía.
Entrada.	Chequea cada la variable global recogida por parámetros.
Resultado esperado.	Retorna los datos asociados a las entidades.

Prueba de caja negra

Las pruebas de caja negra son las que no toman en cuenta el código, el que lo prueba no sabe cómo está estructurado por dentro el programa o bien no necesita saber nada de programación, solo necesita saber cuáles pueden ser las posibles entradas sin necesidad de entender cómo se deben obtener las salidas, donde se trata de encontrar errores en la interfaz mientras se está usando (LÓPEZ, 2005).

Las pruebas de caja negra se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiene a su especificación. Por ello se denominan pruebas funcionales, y el probador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo por dentro. Las pruebas de caja negra están especialmente indicadas en aquellos módulos que van a ser interfaz con el usuario (en sentido general: teclado, pantalla, ficheros, canales de comunicaciones, etc.) (LÓPEZ, 2005)

Descripción del caso de prueba Adicionar dependencia entre las actividades en la carta tecnológica.

Se realizó un caso de prueba para cada requisito implementado. A continuación, se muestra el caso de prueba realizado al requisito Adicionar dependencia entre las actividades en la carta tecnológica que describe cada uno de los escenarios que pueden existir ante las posibles acciones realizas por el usuario. De manera similar se realizaron los casos de prueba a los requisitos restantes. Para ver los requisitos restantes dirigirse al **anexo 4**.

Condiciones de ejecución:

- ✓ El usuario debe estar autenticado.
- ✓ Debe estar definido al menos una carta tecnológica con actividades asociadas a ella.
- ✓ Se debe seleccionar la opción **Inicio/ Planificación/ Planificación agropecuaria/ Producción/ Carta tecnológica/ Listar Cartas/ Listar actividades de la carta/ Adicionar dependencia entre las actividades en la carta tecnológica.**

Tabla 3.3-3Diseño de caso de prueba Adicionar dependencia entre las actividades en la carta tecnológica.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Adicionar dependencia entre las actividades en la carta tecnológica.	Con la realización de este requisito se pretende adicionar dependencias entre actividades de una carta tecnológica	EP 1.1: Adicionar dependencia correctamente.	1. Se muestra la interfaz de Listar actividades de una carta tecnológica. 2. Se selecciona el botón **Adicionar**. 3. Se muestra la interfaz Adicionar dependencias. 4. Se seleccionan las actividades a adicionar. 5. Se presiona el botón **Aceptar**. 6. El sistema muestra el mensaje de información: "*La(s) dependencia(s) ha(n) sido adicionada satisfactoriamente*" y vuelve a la interfaz Principal.
		EP 1.2: Adicionar dependencia incorrectamente.	1. Se muestra la interfaz de Listar actividades de una carta tecnológica. 2. Se selecciona el botón **Adicionar**. 3. Se muestra la interfaz Adicionar dependencias.

			4. No se selecciona ninguna actividad para adicionar. 5. Al presionar Aceptar o Aplicar no permite realizar ninguna operación.
		EP 1.3: Adicionar dependencia correctamente presionando el botón **Aplicar.**	1. Se muestra la interfaz de Listar actividades de una carta tecnológica. 2. Se selecciona el botón **Adicionar**. 3. Se muestra la interfaz Adicionar dependencias. 4. Se seleccionan las actividades a adicionar. 5. Se presiona el botón **Aplicar.** 6. El sistema muestra el mensaje de información: "*La(s) dependencia(s) ha(n) sido adicionada satisfactoriamente*" y se limpia la interfaz.
		EP 1.4: Cancelar operación.	1. Se muestra la interfaz de Listar actividades de una carta tecnológica. 2. Se selecciona el botón **Adicionar**. 3. Se muestra la interfaz Adicionar dependencias.

			4. Se introducen todos los datos. 5. Se presiona el botón **Cancelar**. 6. Se cancela la acción, se cierra la ventana y se regresa a la interfaz principal.

Descripción de variables

Tabla 3.3-4Descripción de variables del RF Adicionar dependencia entre las actividades en la carta tecnológica.

No	Nombre del campo	Clasificación	Puede ser nulo	Descripción
1	actividad	CheckBox	Si	Permite seleccionar una actividad para adicionarla como dependencia de la anterior seleccionada en el listar actividades de la carta seleccionada.

Juego de datos a probar

Tabla 3.3-5Juego de datos a probar del RF Adicionar dependencia entre las actividades en la carta tecnológica.

Id del escenario	Escenario	Variable 1	Respuesta del sistema
		actividad	

EP 1.1	Adicionar dependencia correctamente.	V (true)	El sistema muestra el mensaje de información: "*La(s) dependencia(s) ha(n) sido adicionada satisfactoriamente*" y vuelve a la interfaz Principal.
EP 1.2	Adicionar dependencia incorrectamente.	V (false)	Al presionar Aceptar o Aplicar no permite realizar ninguna operación.
EP 1.3	Adicionar dependencia correctamente presionando el botón **Aplicar.**	V(false)	El sistema muestra el mensaje de información: "*La(s) dependencia(s) ha(n) sido adicionada satisfactoriamente*" y se limpia la interfaz.
EP 1.4	Cancelar operación.	NA	El sistema cancela las acciones y cierra la ventana Adicionar dependencias.

En la primera iteración de las pruebas de software de caja negra realizadas se detectaron un total de 10 no conformidades en el componente de Análisis de ellas 2 de funcionalidad y 8 de ortografía, las cuales fueron resueltas en su totalidad. En una segunda iteración de pruebas se encontraron 2 no conformidades que no se habían identificado con anterioridad, las mismas de concordancia y fueron resueltas; ya para la tercera iteración de pruebas al sistema, no se detectaron no conformidades. Para finalmente poder comprobar que la aplicación cumple con los requisitos planteados al inicio del desarrollo y que responde exitosamente las peticiones de los usuarios, teniendo como resultado una versión estable del componente.

Pruebas de rendimiento

Las pruebas de rendimiento permiten conocer cómo responde el sistema ante diversas situaciones para detectar fallas que puedan atentar contra la calidad del mismo. Se efectúan para determinar lo rápido que un sistema realiza una tarea en circunstancias particulares de trabajo. Es de vital importancia que las condiciones en las que se desarrollen las pruebas sean similares a las esperadas en la realidad tratando de garantizar que los resultados sean lo más precisos posibles (Pressman, 2010). Esta contiene las pruebas de carga y de estrés brevemente explicadas a continuación.

Pruebas de carga

Las pruebas de cargas son utilizadas para ver cómo se comporta la aplicación ante la conexión de un gran número de usuarios sometiendo la misma al límite de su funcionamiento con el objetivo de verificar la robustez de la aplicación bajo una carga intensiva, permitiendo así determinar el límite real de usuarios concurrentes (Pressman, 2010).

Pruebas de estrés

Las pruebas de estrés son utilizadas para analizar las respuestas de la aplicación ante las peticiones de múltiples números de usuarios. Este tipo de prueba permite conocer la solidez del sistema en los momentos de carga extrema y permite determinar si rendirá lo suficiente en caso de que la carga real supere a la esperada (Pressman, 2010).

Selección de la muestra de las funcionalidades del sistema para la prueba de rendimiento.

Se realiza la selección de las funcionalidades del sistema a ser analizadas. A continuación, se relacionan las funcionalidades seleccionadas para ser parte de la muestra a estudiar con la herramienta JMeter:

- Visualizar plan-normados y directos por ejercicio.
- Visualizar resumen de gráficas de niveles de actividad por ejercicio.
- Listar dependencia de las actividades en la carta tecnológica.

Las características de la estación de trabajo en las que se realizarán las pruebas de rendimiento son las siguientes:

- Memoria de 6 GB RAM.
- Microprocesador Intel™Core i5.
- Disco duro de 500GB.
- Servidor de aplicaciones Apache 2.2.
- Lenguaje de programación de lado del servidor PHP 5.4.4.

A continuación, en la **Tabla 3.4-6** se muestran los resultados obtenidos, que fueron extraídos de las pruebas realizadas al sistema en la herramienta JMeter.

Tabla 3.3-6Resultados de las pruebas de rendimiento.

Aplicado a	**Usuarios (Muestras)**	**Tiempos de ejecución(ms)**					**Rendimiento**	
		Mín.	**Máx.**	**Media**	**Mediana**	**%Error**	**Pet/seg**	**Kb/seg**
Visualizar plan-normados y directos por ejercicio.	50 200 400	1 1 1	3410 3410 4020	732 263 850	22 7 27	0.00 0.00 6.3	40.6 263 365	170 1363 1943
Visualizar resumen de gráficas de niveles de actividad por ejercicio.	50 200 400	1 1 1	3854 3482 4863	607 588 597	5 18 13	0.00 0.03 0.05	114 198.9 206.6	147.9 272 282.5
Listar dependencia de las actividades en la carta tecnológica.	50 200 400	1 1 1	3426 3905 4603	283 306 365	8 10 11	0.00 0.00 0.01	135.2 223.8 325.7	613.8 1012.6 1547.1

Los valores que se reflejan en la tabla 3.4-6 se pueden interpretar de la siguiente forma:

La columna Mínimo (Mín.) indica el mínimo de tiempo de ejecución invertido para una petición con n (columna Muestras) usuarios haciendo peticiones de manera concurrente. De manera similar se puede interpretar el resultado de la columna Máximo (Máx.) con la diferencia de que esta última muestra el mayor tiempo de ejecución para una petición. Como su nombre lo indica, la Media representa el tiempo de ejecución promedio de una petición con n usuarios. En el caso de la Mediana, su valor significa que el 50% de las peticiones realizadas por n usuarios tardaron menos del valor reflejado. El Error indica la relación entre el total de peticiones y el número de peticiones que originaron errores. Por último, el Rendimiento (Pet/seg) o (Pet/min) hace referencia al número de peticiones que el servidor puede procesar en un segundo o en un minuto, mientras que el Rendimiento (Kb. /seg) hace referencia a la cantidad de datos que el servidor puede procesar en un segundo.

Los datos obtenidos reflejan que la aplicación responde en períodos de tiempo adecuados definidos en los requerimientos no funcionales **RNF_5** y **RF_6** según lo establecido, a las peticiones realizadas por un número considerable de usuarios concurrentes, lo cual demuestra que el sistema es escalable.

3.3.2 Pruebas de integración

Las pruebas de integración constituyen el mecanismo para comprobar el correcto ensamblaje del sistema completo. Estas pruebas se realizan al terminar las pruebas unitarias con el objetivo de verificar que los módulos o componentes que conforman un sistema funcionan correctamente una vez que han sido integrados. Para la realización de las mismas debe tenerse en consideración un conjunto de aspectos tales como, el grado de complejidad de los módulos o componentes, su importancia funcional con respecto a las especificaciones del sistema, la interrelación entre los mismos y las estadísticas de error en las pruebas unitarias de forma tal que se prueben primeramente los más críticos. Para ejecutar las mismas se seleccionan los casos de pruebas que serán diseñados teniendo en cuenta el grado de dependencia entre los componentes para realizar una funcionalidad específica (Pressman, 2010).

3.3.3 Prueba de aceptación

El uso de cualquier producto de software tiene que estar justificado por las ventajas que ofrece. Sin embargo, antes de empezar a usarlo es muy difícil determinar si sus ventajas realmente justifican su uso. El mejor instrumento para esta determinación es la llamada “prueba de aceptación” (Pressman, 2010).

La persona adecuada (o el equipo adecuado) para llevar a cabo la prueba de aceptación dispone de estos conocimientos y además es capaz de interpretar los requerimientos especificados por los futuros usuarios del sistema de software en cuestión (Pressman, 2010).

El objetivo de las pruebas de aceptación es validar que el sistema cumple con el funcionamiento esperado y permitir al usuario de dicho sistema determine su aceptación, desde el punto de vista de su funcionalidad y rendimiento. Las pruebas de aceptación son definidas por el usuario del sistema y preparadas por el equipo de desarrollo, aunque la ejecución y aprobación final corresponden al usuario (Pressman, 2010).

Los principales especialistas trabajaron con el software y validaron la aceptación del mismo, ya que cumple con la estrategia de planificación llevada a cabo en las diferentes granjas agropecuarias del país cumpliéndose así los requisitos capturados en la primera etapa.

3.4 Validación de usuarios

Para la validación de la idea a defender propuesta en la investigación se utiliza el método criterio de expertos en su variante Delphi (Sánchez, 2015) empleando los siguientes pasos:

- Identificación de los posibles expertos.
- Selección de los expertos.
- Realización de la consulta a los expertos y procesamiento y valoración de la información obtenida.

Para identificar los posibles expertos se tuvieron en cuenta, la experiencia profesional en relación con sistemas de planificación para diferentes entidades y procesos desarrollados con el uso del marco de trabajo Zeolides.

En la siguiente tabla se muestran los expertos seleccionados.

Tabla 3.4-1Expertos utilizados en la validación de la propuesta de solución.

No	Experto	Entidad	Años de experiencia
1	Olennys Carcasés Durán	División gestión de entidad, centro de Planificación.	5
2	Yunior Orosa Velázquez	División gestión de entidad, centro de Planificación.	2
3	Igniris Valdivia Báez	División gestión de entidad, centro de Planificación.	8
4	Roberto Yasel García Quintana	División gestión de entidad, centro de Planificación.	4
5	Liván Valdés Yero	División gestión de entidad, centro de Planificación.	7

Luego de seleccionados los expertos, se sometió a su consideración un instrumento para la validación del desarrollo del componente Análisis del Sistema de Planificación de Producciones. El instrumento se compone de 5 sentencias, así como 5 categorías evaluativas que permitan conocer la opinión de los expertos. Las categorías evaluativas empleadas fueron: muy adecuado (MA), bastante adecuado (BA), adecuado (A), poco adecuado (PA) e inadecuado (I).

Tabla 3.4-2Sentencias a evaluar por los expertos para validar la solución

No.	Sentencias plasmadas en la consulta realizada a los expertos
1	Veracidad de los datos
2	Relevancia de los datos
3	Interfaz intuitiva
4	Interactividad de las gráficas

5	Rapidez de respuesta

Se calcula el coeficiente de Kendall que permite analizar la concordancia en las valoraciones realizadas por los expertos (Sampieri, 2014). El coeficiente de concordancia (W) será un índice de la divergencia del acuerdo efectivo entre los expertos. El coeficiente de concordancia de Kendall se obtiene de la expresión:

W = 12S/K2 (N3-N)

Donde S representa el cuadrado de las desviaciones medias, K el número de expertos y N el número total de aspectos a evaluar. El valor de W oscila entre 0 y 1. El valor de 1 significa una concordancia de acuerdos total y el valor de 0 un desacuerdo total.

Se aplica además la Prueba de Significación de la idea a defender para comprobar el grado de significación de Kendall, planteándose la idea a defender nula y la alternativa de la siguiente forma: donde H0: no existe concordancia entre los expertos y H1: existe concordancia entre los expertos.

X2=K(N-1) W

X2=0.352

El X2 calculado se compara con el tabulado en la tabla del percentil de la distribución X2. Para tener un 95% de confianza se utilizará α=0.05. Si se cumple que X2 calculada < X2(α, N-1), se obtiene que 0.352<9.4877 entonces es válida la idea a defender alternativa H1 de que existe concordancia entre los expertos.

Los criterios aportados por los expertos se someten a una prueba estadística no paramétrica que permite concluir qué valoración final tiene cada uno de los aspectos a evaluar. Para los datos anteriores se debe confeccionar una distribución de frecuencia a partir de los datos primarios para cada uno de los aspectos sometidos a consulta (Castro, 2014).

Tabla 3.4-3Distribución de frecuencia para los datos primarios obtenidos

Categorías evaluativas	**Frecuencia absoluta**	**Frecuencia relativa**
Muy adecuado	23	0.92
Bastante adecuado	1	0.04

Adecuado	1	0.04
Poco adecuado	0	0
Inadecuado	0	0

Los resultados obtenidos en la validación pueden observarse en la **Figura 3.5-1**.

Figura 3.4-1Comportamiento de la valoración de los expertos por categorías evaluadas

De acuerdo con los datos el 92% de los aspectos analizados fueron valorados de muy adecuado, el 4% de bastante adecuado y un 4% de adecuado. El análisis de los resultados obtenidos de la consulta de expertos permitió identificar que existe una coincidencia en las valoraciones realizadas sobre el alto valor que presenta el componente Análisis del Sistema de Planificación de Producciones. Además, todos los indicadores fueron evaluados satisfactoriamente evidenciando la calidad de la propuesta presentada.

3.5 Conclusiones del capítulo

Con la realización de este capítulo, se le dio cumplimiento a la fase de Construcción del componente propuesto. Se realizó un bosquejo de distintos aspectos con gran significado para las pruebas de software, resaltándose la importancia de las mismas en el desarrollo de un proyecto.

Después de realizadas las pruebas de caja negra, de aceptación, de carga y estrés, se obtuvo una versión estable del producto, obteniendo como resultado que el componente desarrollado cumple con el funcionamiento esperado y satisface todos los requisitos funcionales.

Conclusiones

- El estudio de diferentes módulos y componentes de análisis de sistemas de planificación determinó que ninguno de estos módulos y componentes cumple con los requisitos necesarios planteados por el usuario, pues no planifican producciones agropecuarias. Se hizo necesario el desarrollo de un componente que cumpla con estas funcionalidades.
- Para el desarrollo del componente se identificaron y aprobaron un grupo de requisitos funcionales y no funcionales con los que debe cumplir el producto final, los mismos fueron determinados en conjunto con los especialistas de la Unión Agropecuaria Militar lo que permitió definir las características del producto final.
- Para la realización del diseño se definieron un conjunto artefactos especificados el proceso de desarrollo PRODESOFT. Estos permitieron reducir los esfuerzos en el desarrollo, eficiencia y consistencia de los diseños del producto final.
- Para la implementación del producto final se utilizó un conjunto de patrones y estándares de codificación definidos en el documento “Estándar de codificación para PHP Versión 2.0.0”, elaborado por el Comité de rol de Lógica de Negocio de la XETID.
- Para mejorar la calidad del producto realizado se efectuó un conjunto de pruebas; estas se aplicaron atendiendo a los requisitos, como se define en la segunda fase propuesta por el proceso de desarrollo PRODESOFT. Las pruebas se aplicaron en varias iteraciones hasta tener un producto cien por ciento funcional.
- En sentido general se puede afirmar que se cumplieron los objetivos trazados para el desarrollo del trabajo. Se logró implementar un componente Análisis que permite interpretar los datos valiosos resultado de las producciones agropecuarias y mostrarlos de una forma clara y sencilla para contribuir a mejorar el proceso de toma de decisión de los expertos sobre la planificación de producciones dentro de la agricultura.

Recomendaciones:

- Continuar el desarrollo del componente Diagrama de Gantt para mostrar la planificación de las actividades asociadas a las producciones.
- Desarrollar el componente Calendario para desplegar la realización de las actividades asociadas a la producción en una fecha dada y velar por su cumplimiento.

Referencias bibliografías

Álvarez, Miguel Ángel. 2015. DesarrolloWeb.com. *DesarrolloWeb.com.* [Online] 2015. [Cited: 5 20, 2018.] https://www.desarrolloweb.com/articulos/392.php.

Perurena Cancio, Dra. Lilliam and Morágu, Ing. Mercedes. 2013. *Usabilidad de los sitios Web, los métodos y las técnicas para la evaluación.* La Habana: s.n., 2013.

R. Demey, J., Adams, M. and Freites, H. 1992. USO DEL METODO DE ANALISIS DE COMPONENTES PRINCIPALES PARA LA CARACTERIZACION DE FINCAS AGROPECUARIAS. *USO DEL METODO DE ANALISIS DE COMPONENTES PRINCIPALES PARA LA CARACTERIZACION DE FINCAS AGROPECUARIAS.* [Online] 1992. http://www.sian.inia.gob.ve/revistas_ci/Agronomia%20Tropical/at4403/Arti/demey_j.htm.

BOUDREAU, Tim, et al. 2002. *NetBeans: The Definitive Guide: Developing, Debugging, and Deploying Java Code. " O'Reilly Media, Inc.".* 2002.

Caballero Redondo, Jose. 2012. *Anális de la aplicación JMeter.* 2012. 1.

CARRIZO, Dante. 2013. *Contextual dynamic of the software requirements elicitation. Revista Facultad de Ingeniería Universidad de Antioquia.* 2013. 69.

Castro, L. 2014. *Guía de gestión del riesgo tecnológico para el tratamiento de la seguridad durante el proceso de desarrollo de software.* 2014.

Curbelo Oliva, Lissa, et al. 2008. *Proceso de Desarrollo y Gestión de Proyectos de Software.* 2008. 1.5.

DÍAZ, Alecxys, GONZÁLEZ, J. C. and RUIZ, M. 2005. *Implantación de un sistema ERP en una organización.* s.l. : RISI , vol. 2, 2005. 3.

Ecma, Internacional. 2017. *ECMAScript-262.* 2017. 8.

Ecured. 2017. Ecured. *Ecured.* [Online] 2017. [Cited: 4 13, 2018.] https://www.ecured.cu/Empresa_de_Tecnolog%C3%ADas_de_la_Informaci%C3%B3n_para_la_Defensa.

FUENTES, Lidia, JIMÉNEZ, Daniel and PINTO, Mónica. 2003. *Hacia un entorno de desarrollo integrado basado en componentes y aspectos. Desarrollo de Software Orientado a Aspectos.* 2003.

guadaltech. 2018. guadaltech. *guadaltech.* [Online] 2018. https://www.guadaltech.es/odoo.

GUIBOURG, Ricardo A., GHIGLIANI, Alejandro M. and GUARINONI, Ricardo V. 1998. *Introducción al conocimiento científico.* Eudeba: s.n., 1998.

Gutiérrez, JR Vejo. 2017. *Desarrollo de una aplicación para la planificación de horarios.* 2017.

KHOMH, Foutse, et al. 2012. *Do faster releases improve software quality?: an empirical case study of Mozilla Firefox. En Proceedings of the 9th IEEE Working Conference on Mining Software Repositories. .* s.l.: IEEE Press, 2012.

KUAN, Joseph. 2012. *Learning Highcharts.* s.l. Packt Publishing Ltd, 2012.

LARMAN, C. 2003. *UML y Patrones.* 2003. vol 2.

LÓPEZ, Carlos. 2005. *Pruebas de caja negra: una experiencia real en laboratorio.* 2005.

LUJÁN MORA, Sergio. 2002. *Programación de aplicaciones web: historia, principios básicos y clientes web.* s.l.: Editorial Club Universitario, 2002.

McLean, Lorena. 2018. Scribd. *Scribd.* [Online] 2018. https://es.scribd.com/document/241603657/Descripcion-de-Jmeter.

MES, Sistema OEE. 2018. Sistema OEE MES para el Control de Producción y Productividad. *Sistema OEE MES para el Control de Producción y Productividad.* [Online] 2018. http://doeet.es/control-de-productividad.

MOMJIAN, Bruce. 2001. *PostgreSQL: introduction and concepts.* New York: Addison-Wesley, 2001.

MONTOYA, Carlos Eduardo Gómez, URIBE, Christian Andrés Candela and RODRÍGUEZ, Luis Eduardo Sepúlveda. 2013. *Seguridad en la configuración del servidor web Apache.* s.l: INGE CUC, 2013.

MORAL, Juan Antonio Breña. 2008. *Develop lejos programs step by step. .* 2008. 0.4.

ORCHARD, Leslie M., et al. 2009. *Professional JavaScript Frameworks: Prototype, YUI, ExtJS, Dojo and MooTools.* s.l: Wrox Press Ltd., 2009.

Pressman, Roger S. 2010. *Ingeniería de software. Un enfoque práctico.* 2010.

PRESSMAN, Roger S. and TROYA, Jose Maria. 1988. *Ingeniería del software.* 1988.

RAMÍREZ, RAQUEL ZAMBRANO. 2006. *MEDIO, CICLO FORMATIVO GRADO. SISTEMAS GESTORES DE BASE DE DATOS.* 2006.

Rodríguez López, Javier. 2013. *Estándar de codificación para PHP versión 1.0.* La Habana: s.n., 2013.

RUIZ-BERTOL, Fran J. and ZARAZAGA-SORIA, Francisco Javier. 2007. *El Control de Versiones en el aprendizaje de la Ingeniería Informática: Un enfoque práctico.* 2007.

RUMBAUGH, James, BOOCH, Grady and JACOBSON, Ivar. 2017. *The unified modeling language reference manual.* s.l.: Addison Wesley, 2017.

SALINAS IBÁÑEZ, Jesús, et al. 2008. *Innovación educativa y uso de las TIC.* s.l.: Universidad Internacional de Andalucía, 2008.

Sampieri, R. 2014. *Metodologia de la investigacion. Mc Graw-Hill Education.* 2014.

Sánchez, S. 2015. *Estrategia de soporte técnico para el proceso de migración a código abierto de los organismos de la Administración Central del Estado.* Universidad de las Ciencias Informáticas: s.n., 2015.

SANTIAGO, Fernando Martínez, RÁEZ, Arturo Montejo and CUMBRERAS, Miguel Ángel García. 2007. *Representación formal de la estructura lógica de sitios web, y su aplicación a un navegador web multilinge basado en diálogo. Procesamiento del lenguaje natural.* 2007.

Selenne, ERP. 2016. comunicae.es. *comunicae.es.* [Online] 1 12, 2016. [Cited: 5 21, 2018.] https://www.comunicae.es/nota/selenne-erp-un-software-de-planificacion-1194278.

SHASANKAR, Krishna. 2013. *Zend Framework 2.0 by Example: Beginner's Guide.* s.l.: Packt Publishing Ltd, 2013.

Sismagro. 2015. Sismagro Software Agropecuario. [Online] 3 2015. [Cited: 5 23, 2018.] https://www.sismagro.com/.

SOUZA, Maria Helena do Nascimento, SOUZA, Ivis Emília de Oliveira and TOCANTINS, Florence

Synerplus. 2016. softwareseleccion.com. *softwareseleccion.com.* [Online] 3 7, 2016. [Cited: 5 21, 2018.] http://www.softwareseleccion.com/selenne+erp-p-2672.

—. 2017. synerplus.es. *synerplus.es.* [Online] 9 12, 2017. [Cited: 5 21, 2018.] http://www.synerplus.es/Selenne-ERP/ERP-Software-Gestion-de-Fabricacion-y-Produccion/12.html.

WAGE, Jonathan H., BORSCHEL, Roman and BLANCO, Guilherme. 2010. *Doctrine ORM for PHP (1.2).* s.l.: SensioLabs, 2010.

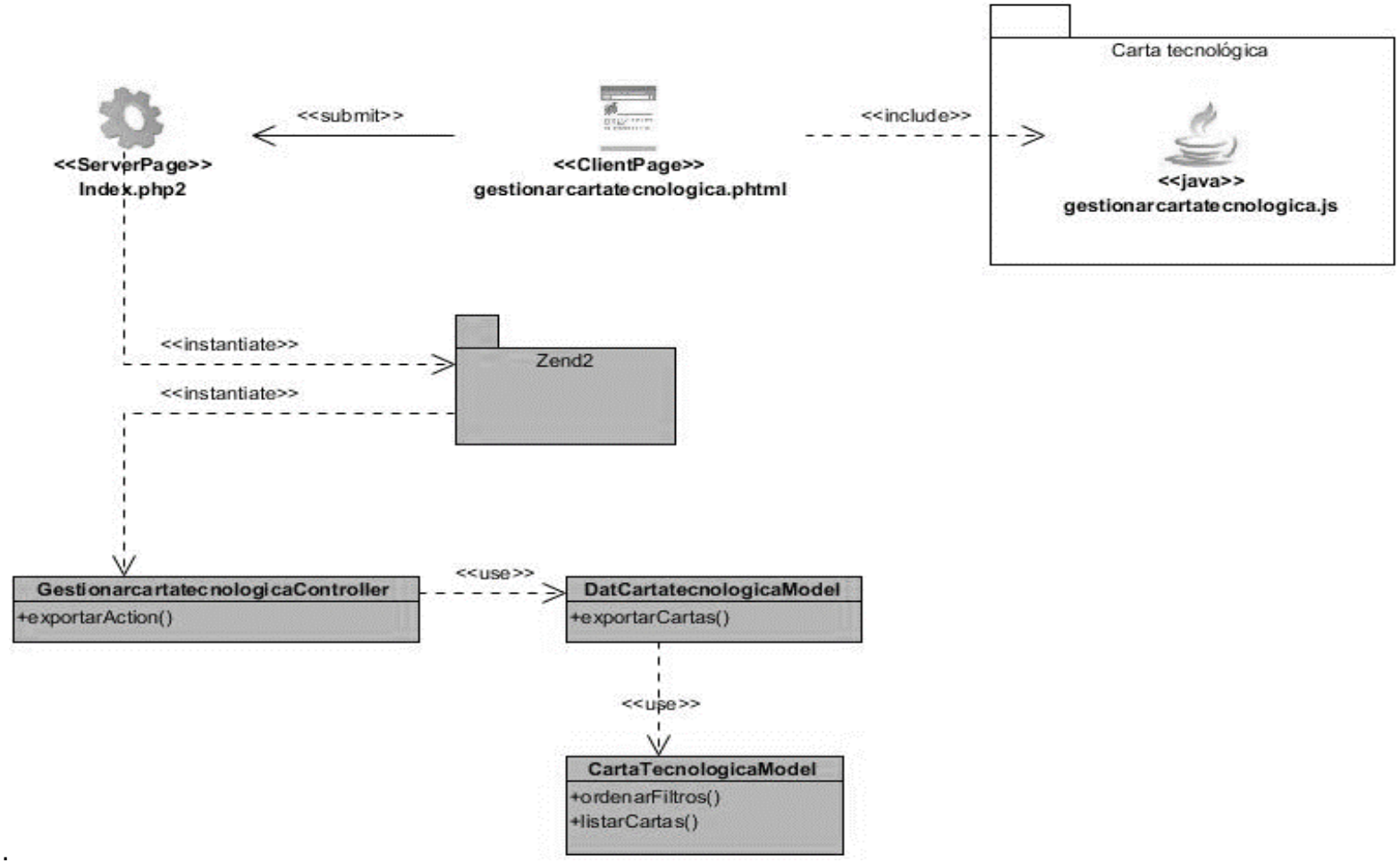
ZARAGOZA, Francisco, Ortiz, José and TORRES. 2005. *Arquitectura de referencia para unidades de control de robots de servicio teleoperados. El Autor,* 2005.

Anexos

Anexo 1

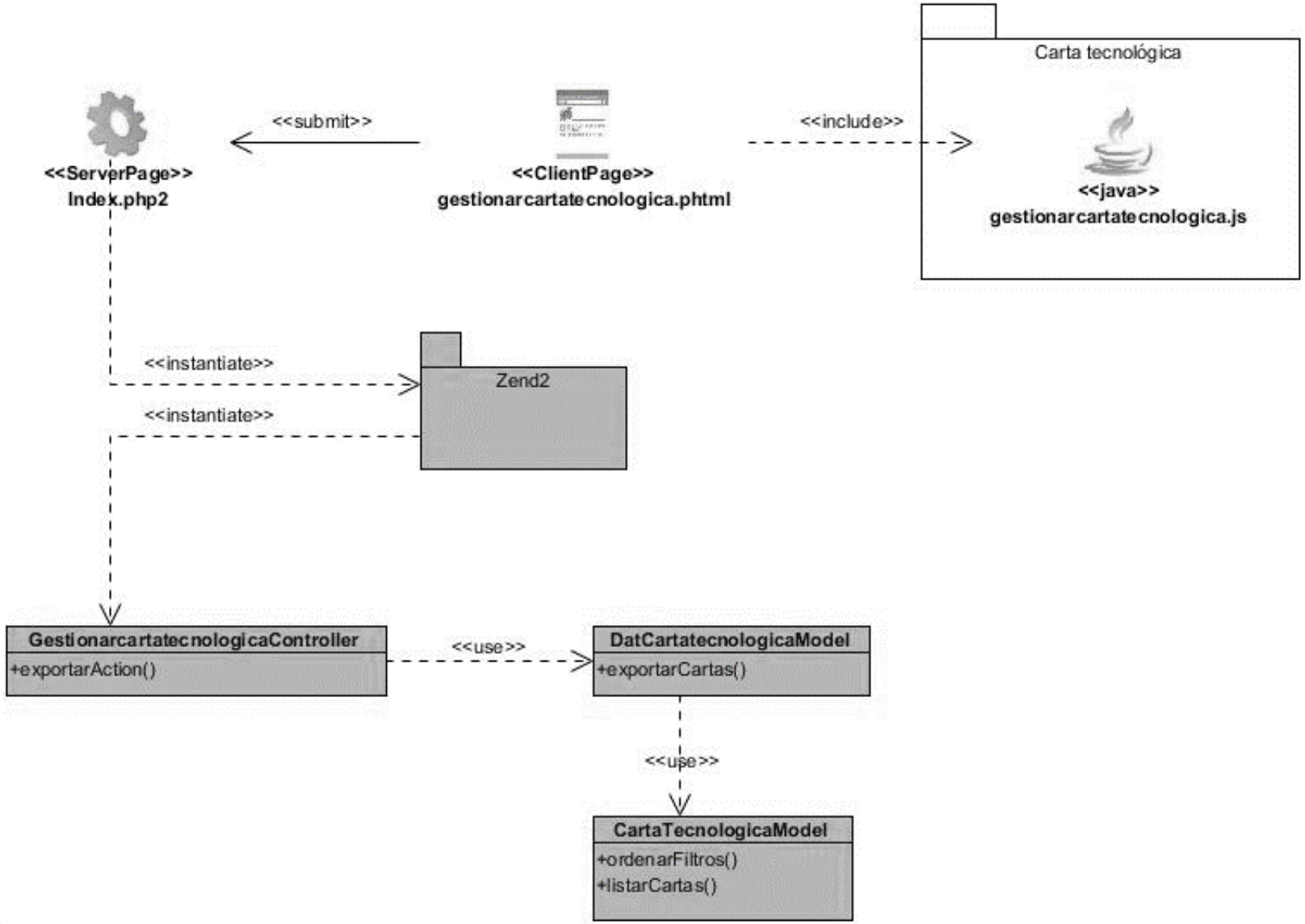
Diagrama de diseño con estereotipos web

RF_5: Exportar cartas



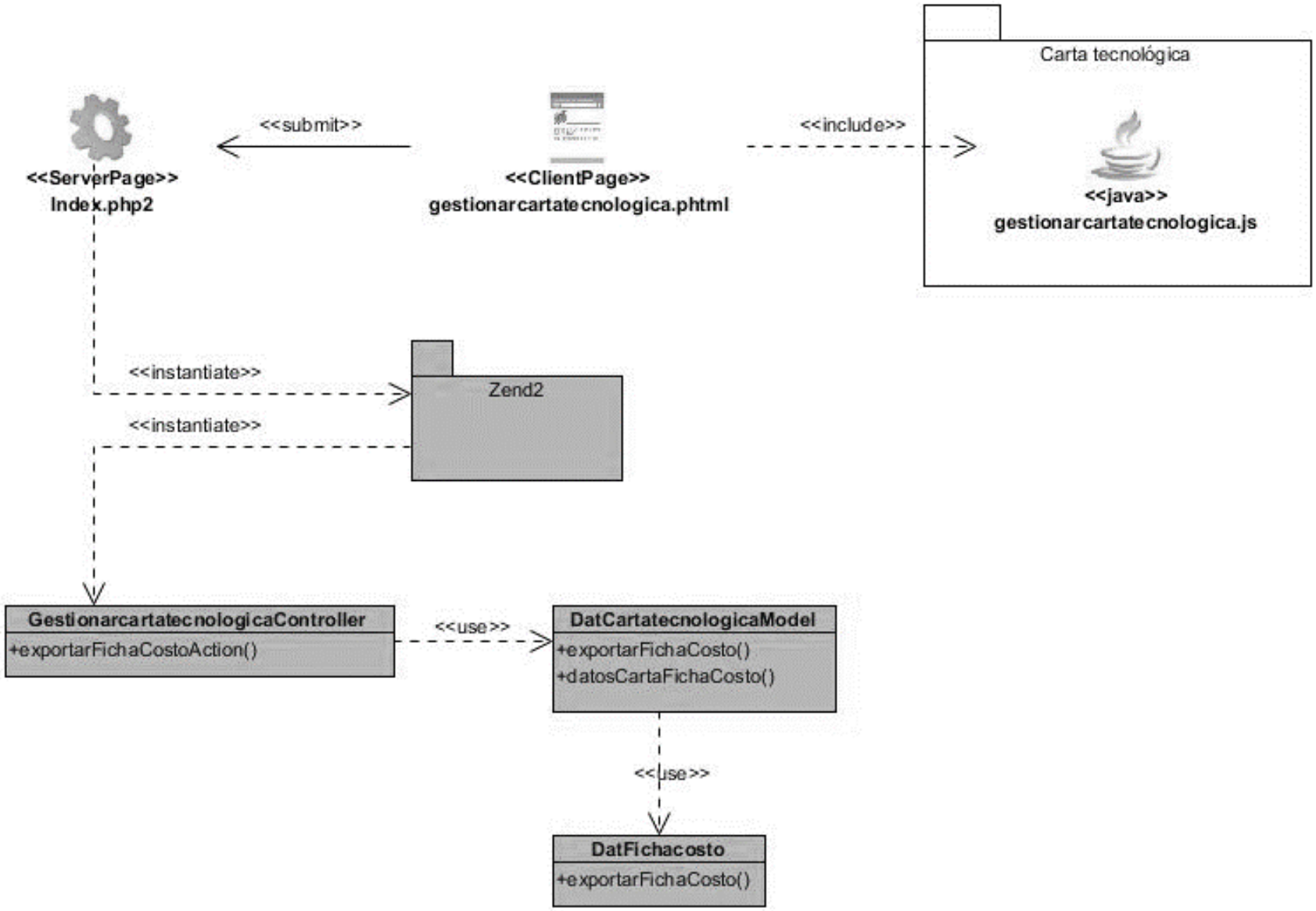
.

RF_6: Exportar cartas y actividades

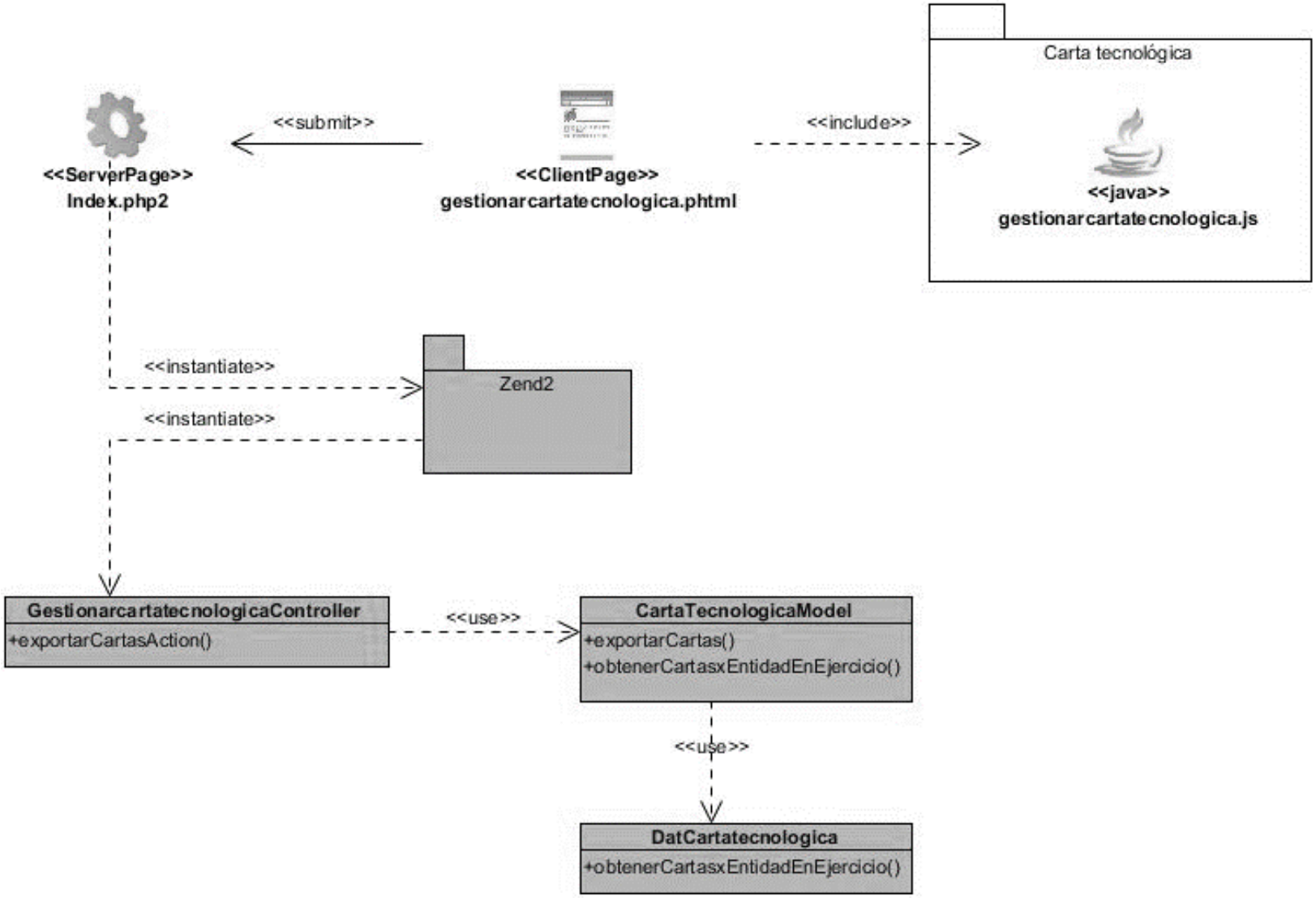


.

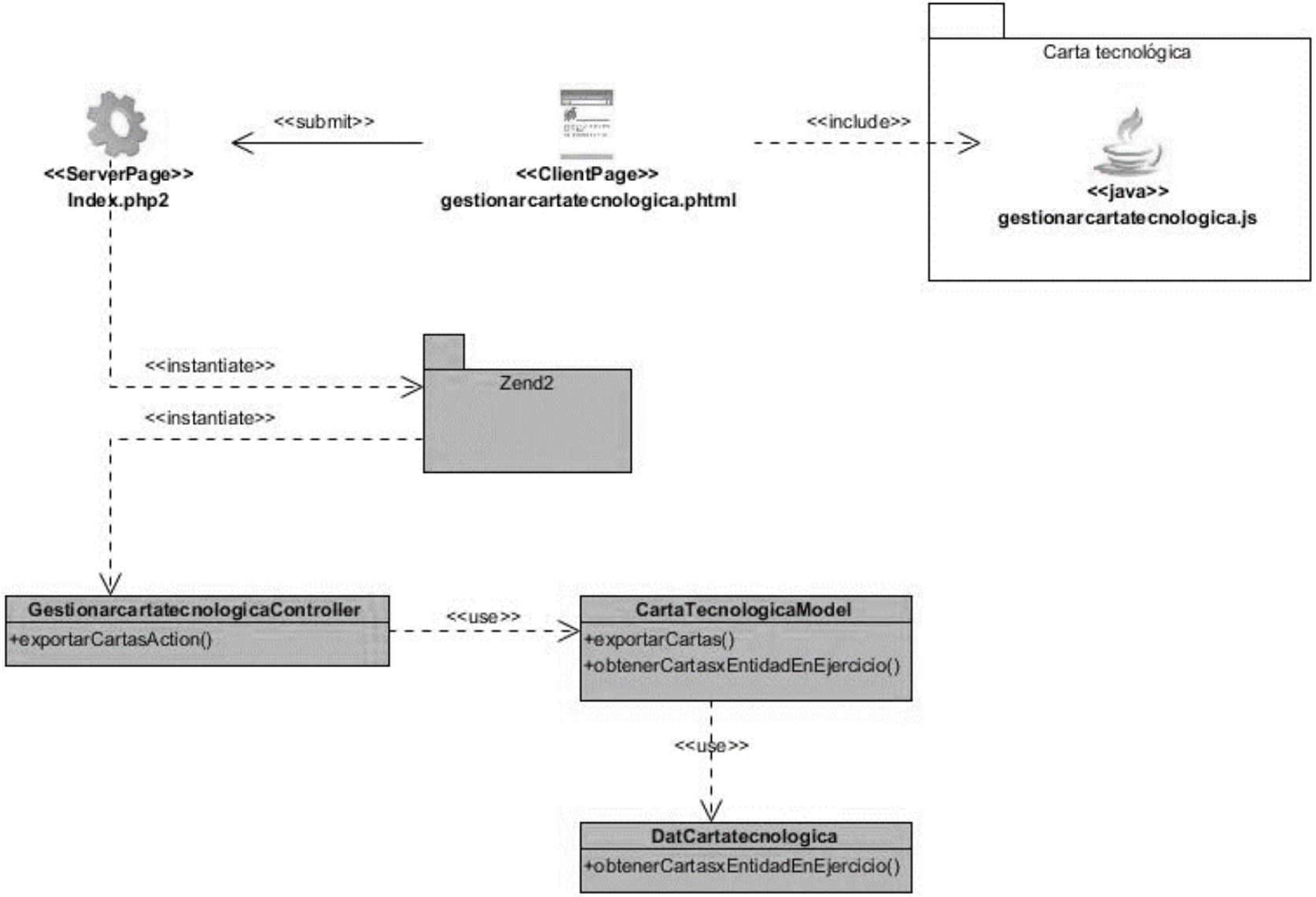
RF_7: Exportar ficha de costo de carta tecnológica.



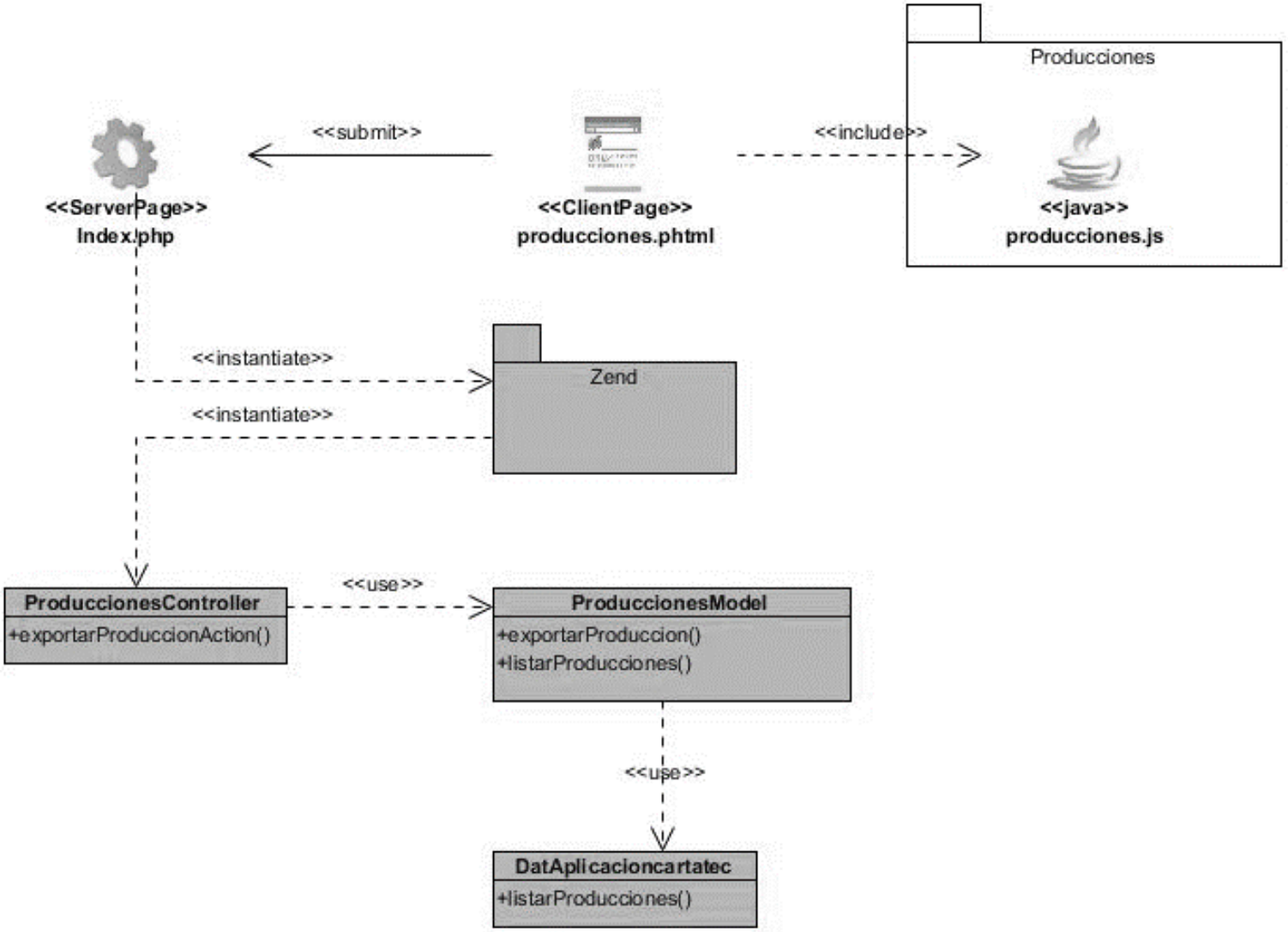
RF_8: Exportar importe de cartas



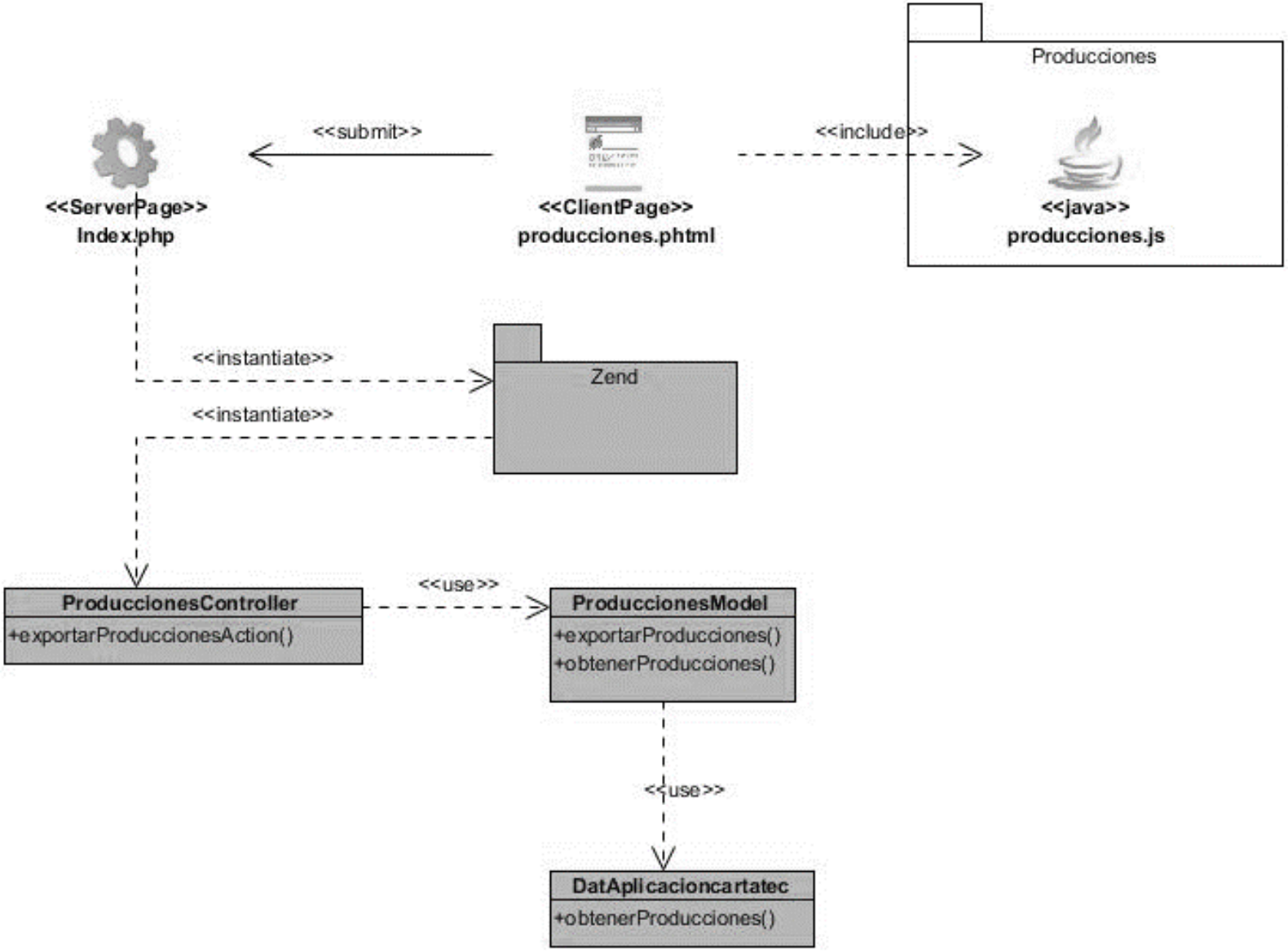
RF_9: Exportar importe de actividades de carta tecnológica.



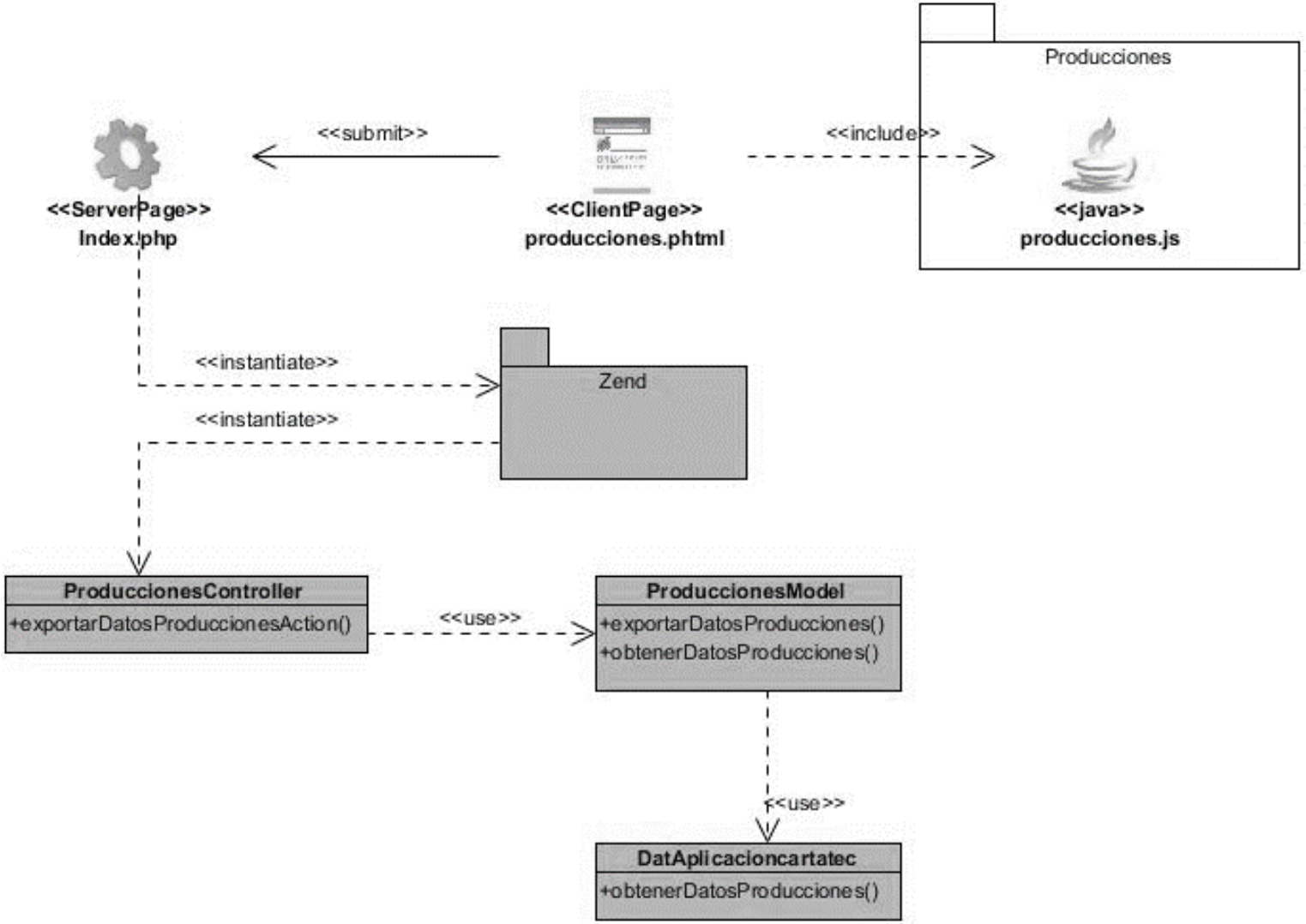
RF_10: Exportar producciones.



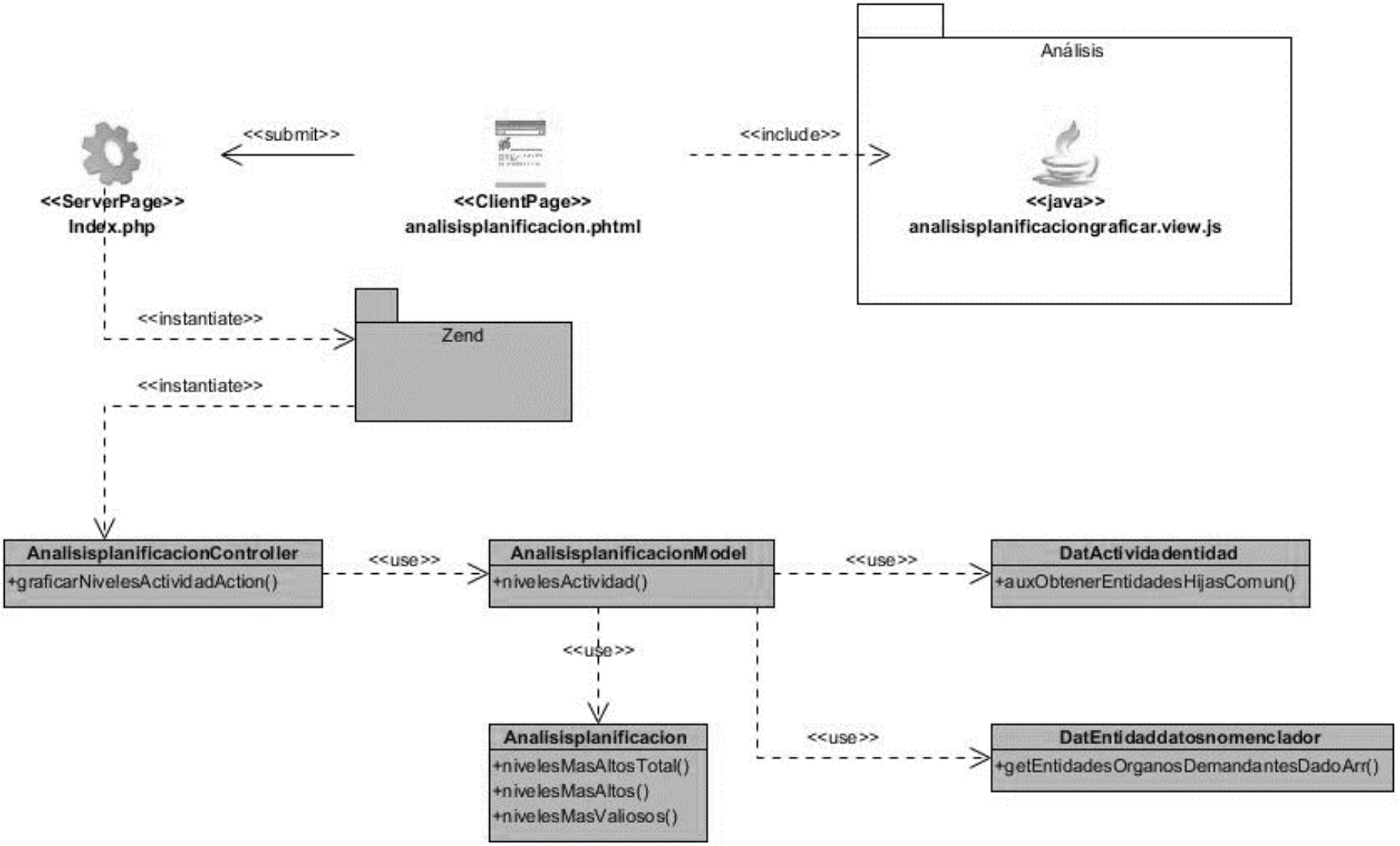
RF_11: Exportar importes de producciones.



RF_12: Exportar importes de actividades de producción.

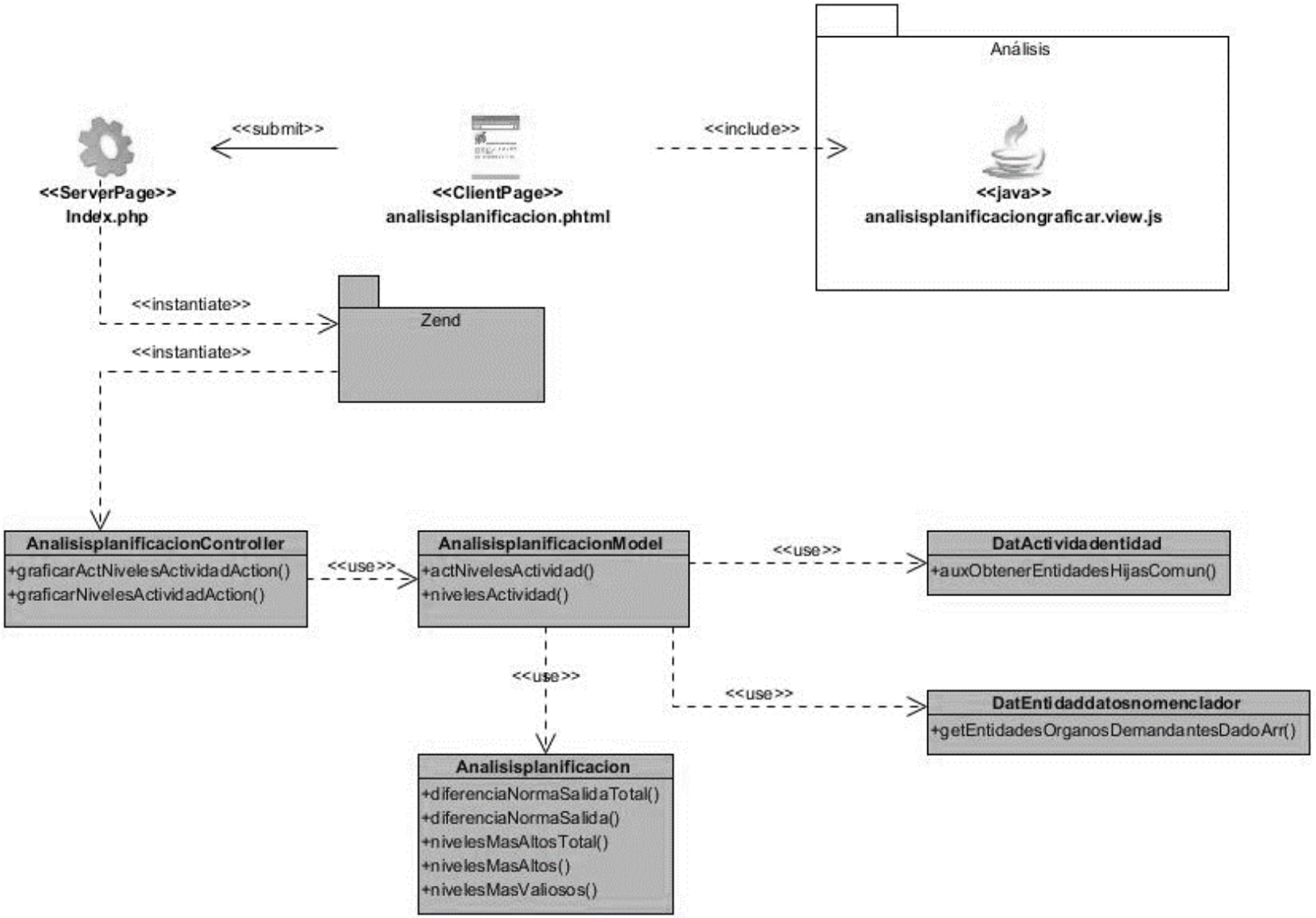


RF_18: Visualizar actividades-niveles de actividad por ejercicio.

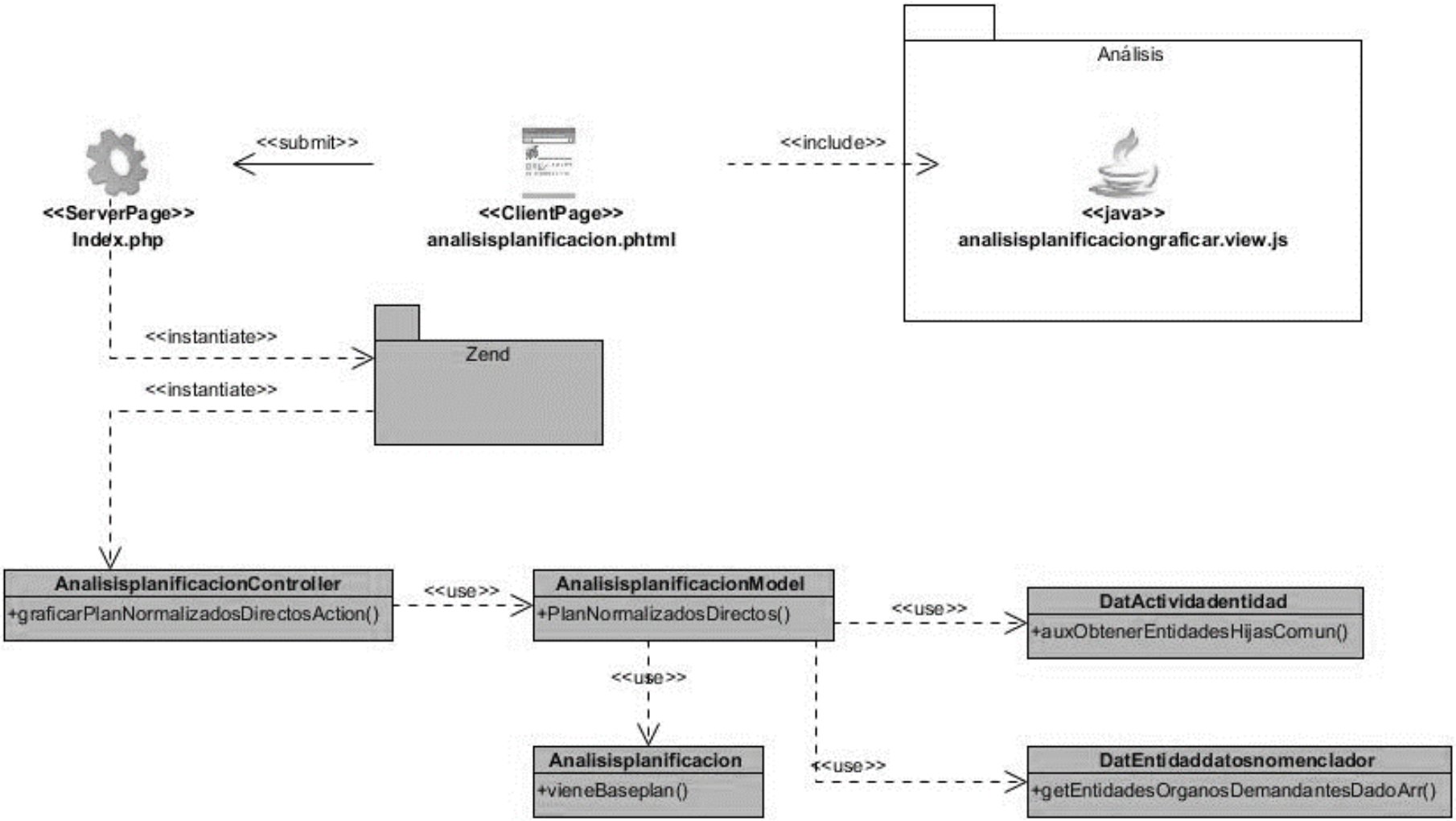


RF_19: Visualizar niveles de actividad por ejercicio.

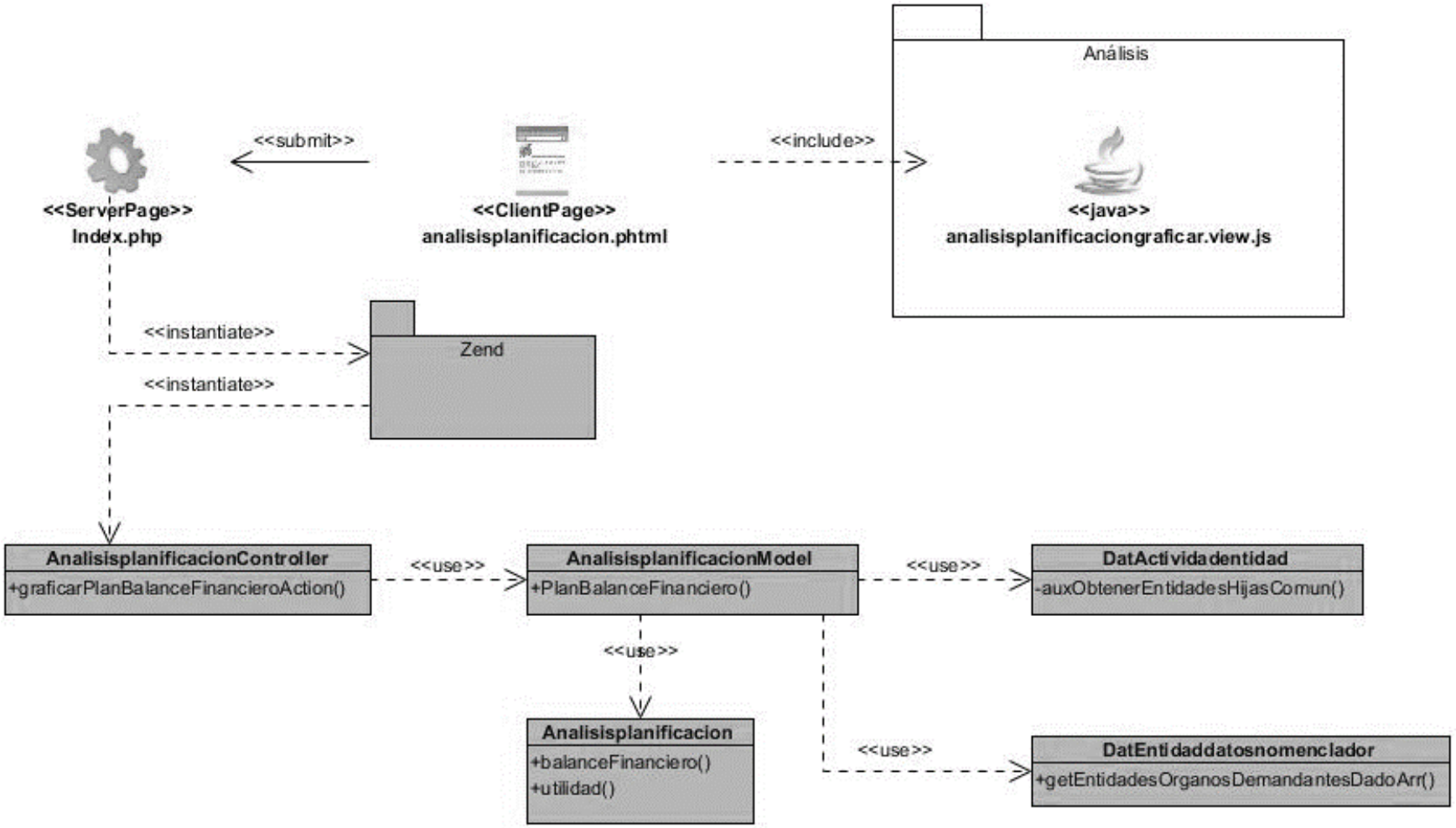
RF_20: Visualizar resumen de gráficas de niveles de actividad por ejercicio.



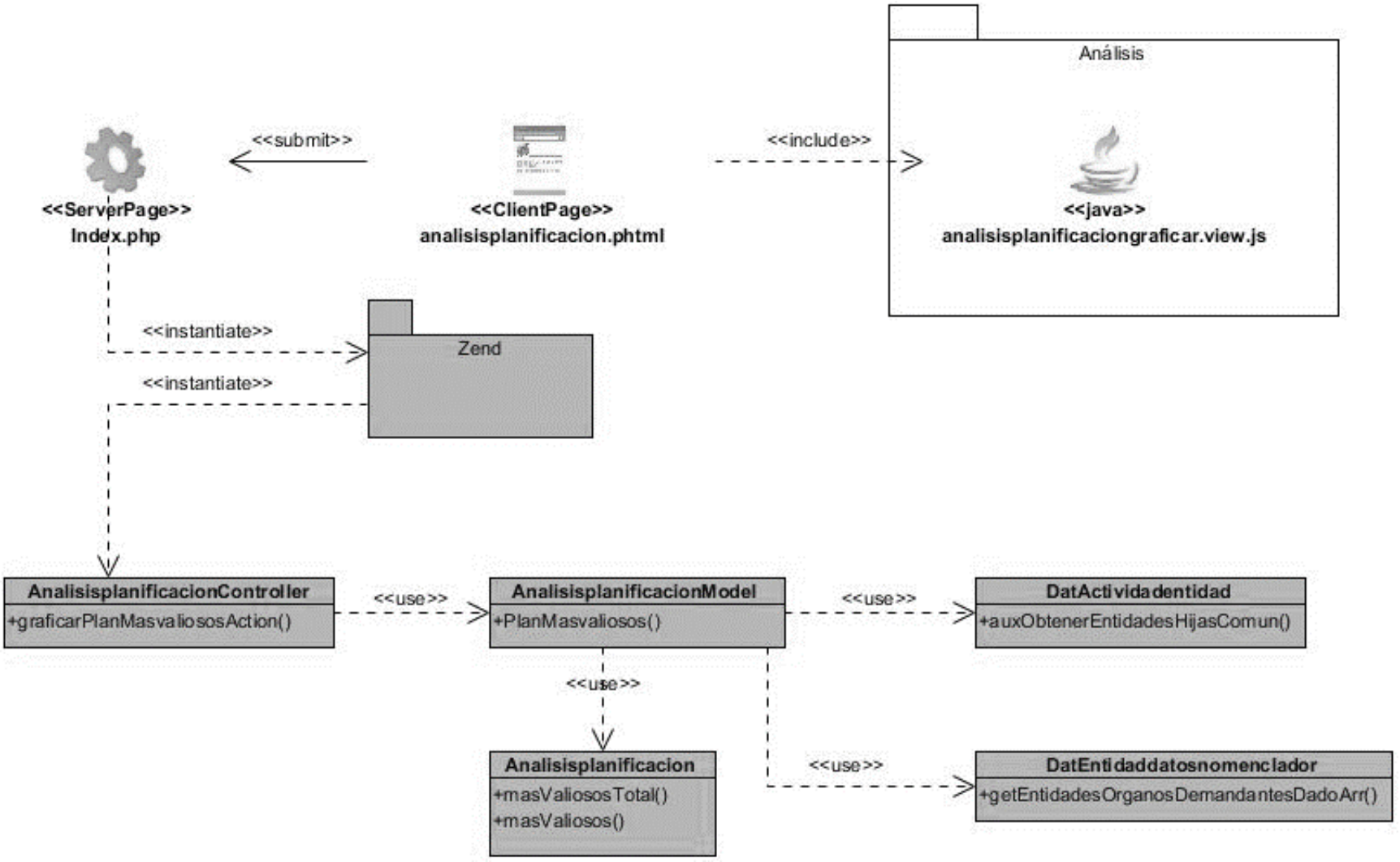
RF_21: Visualizar plan-normados y directos por ejercicio.



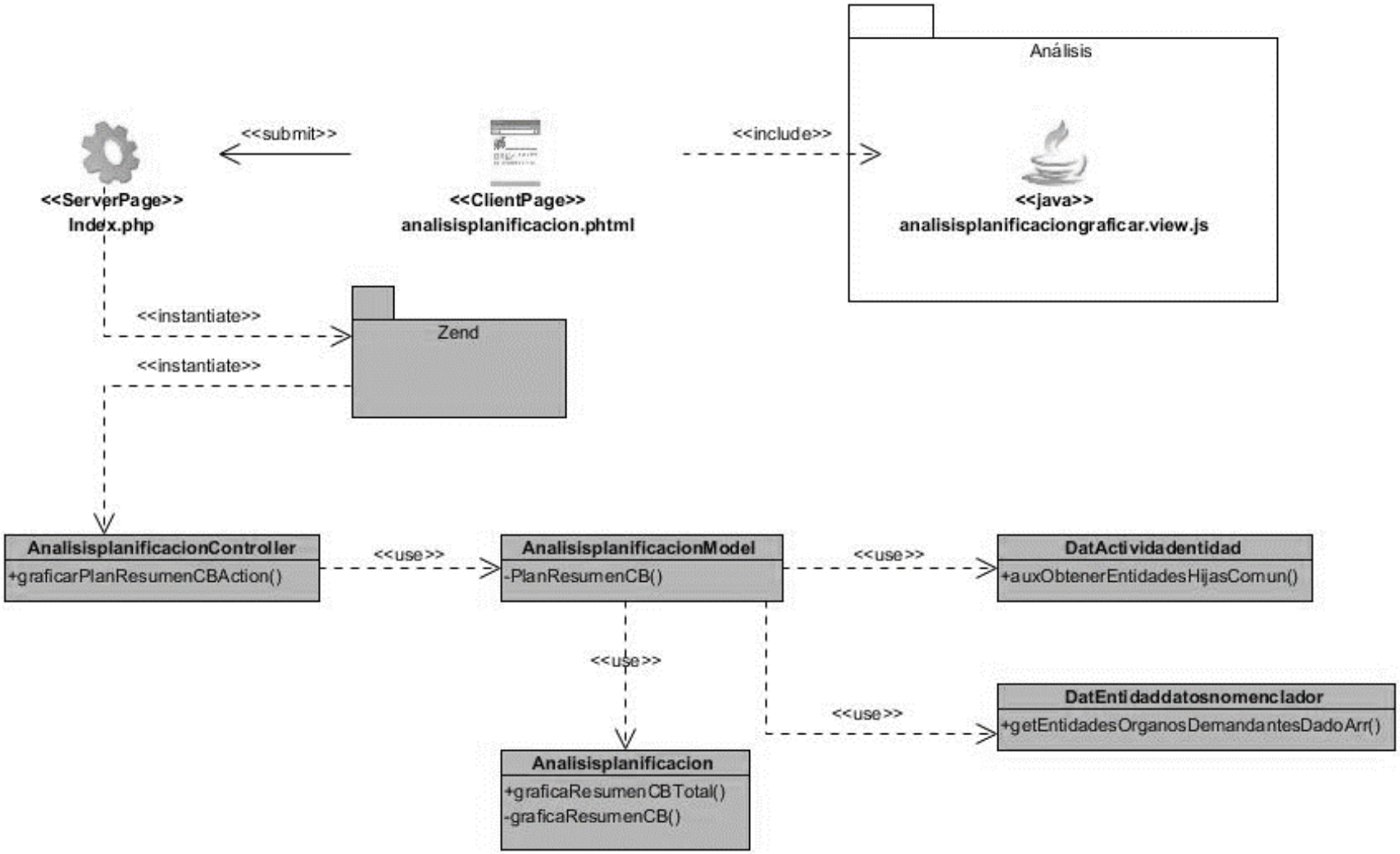
RF_22: Visualizar plan-ingresos-gastos por ejercicio.

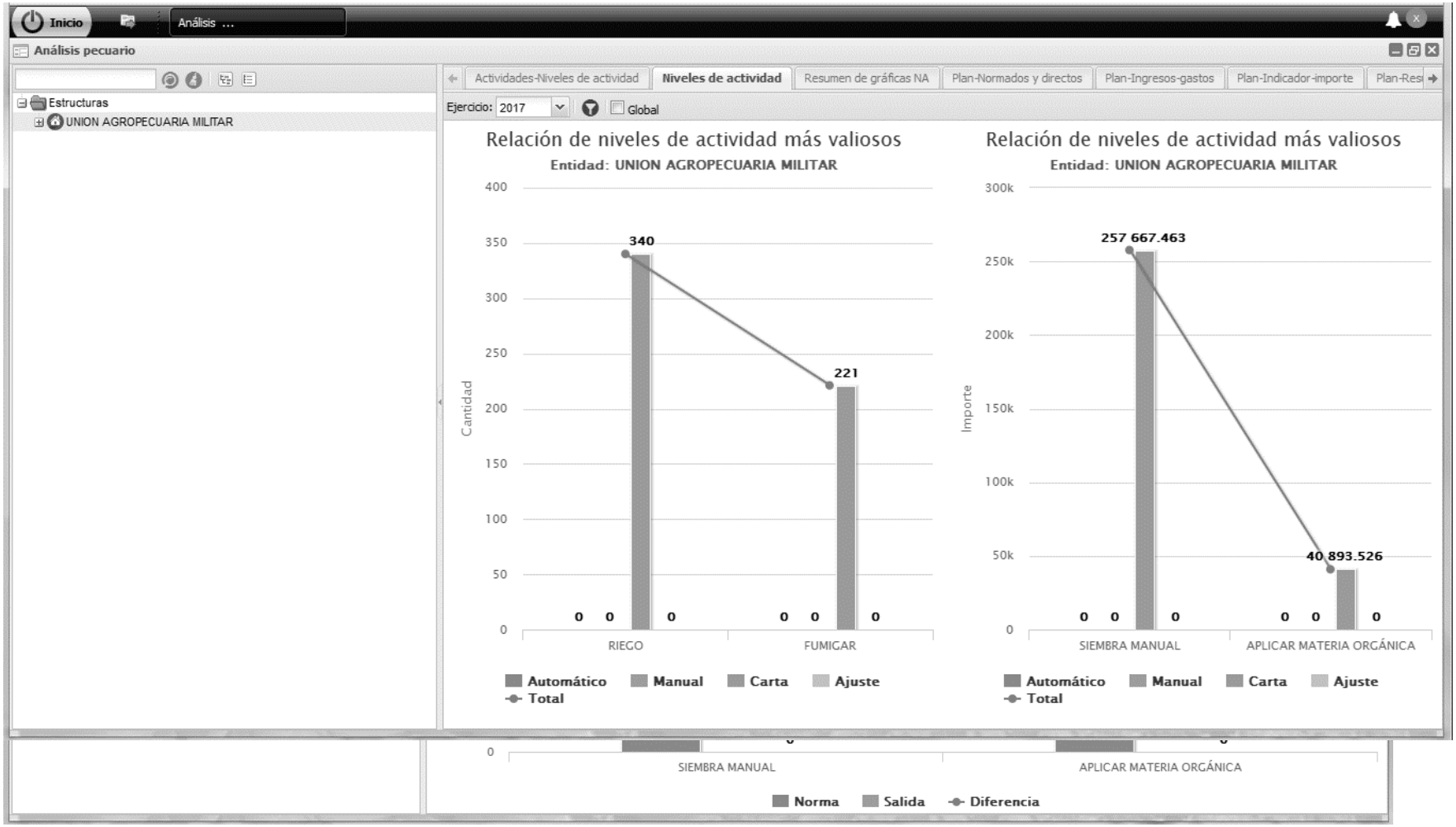


RF_23: Visualizar plan-indicador-importe por ejercicio.



RF_24: Visualizar plan-resumen por centro de balance por ejercicio.



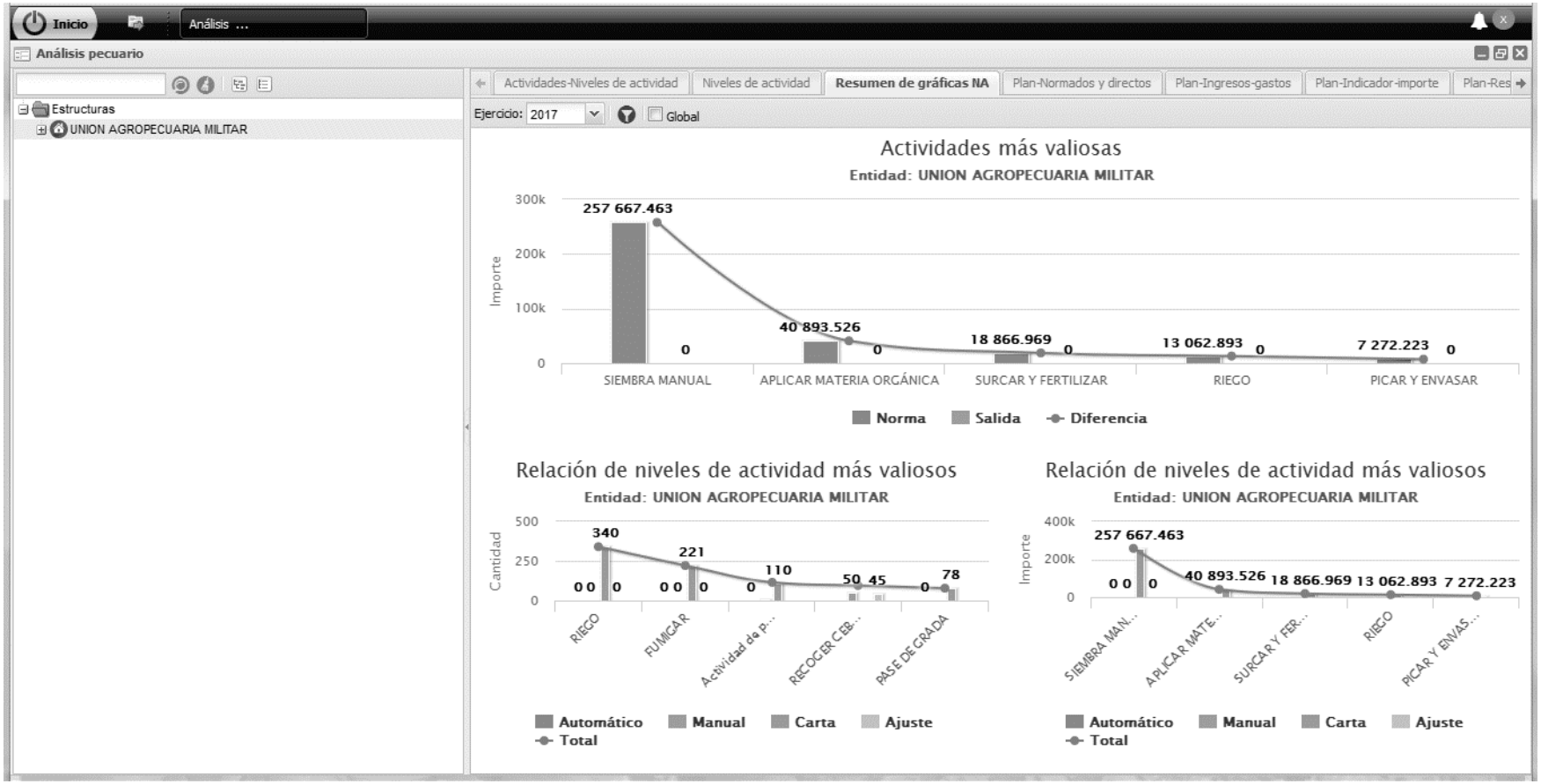


Anexo 2

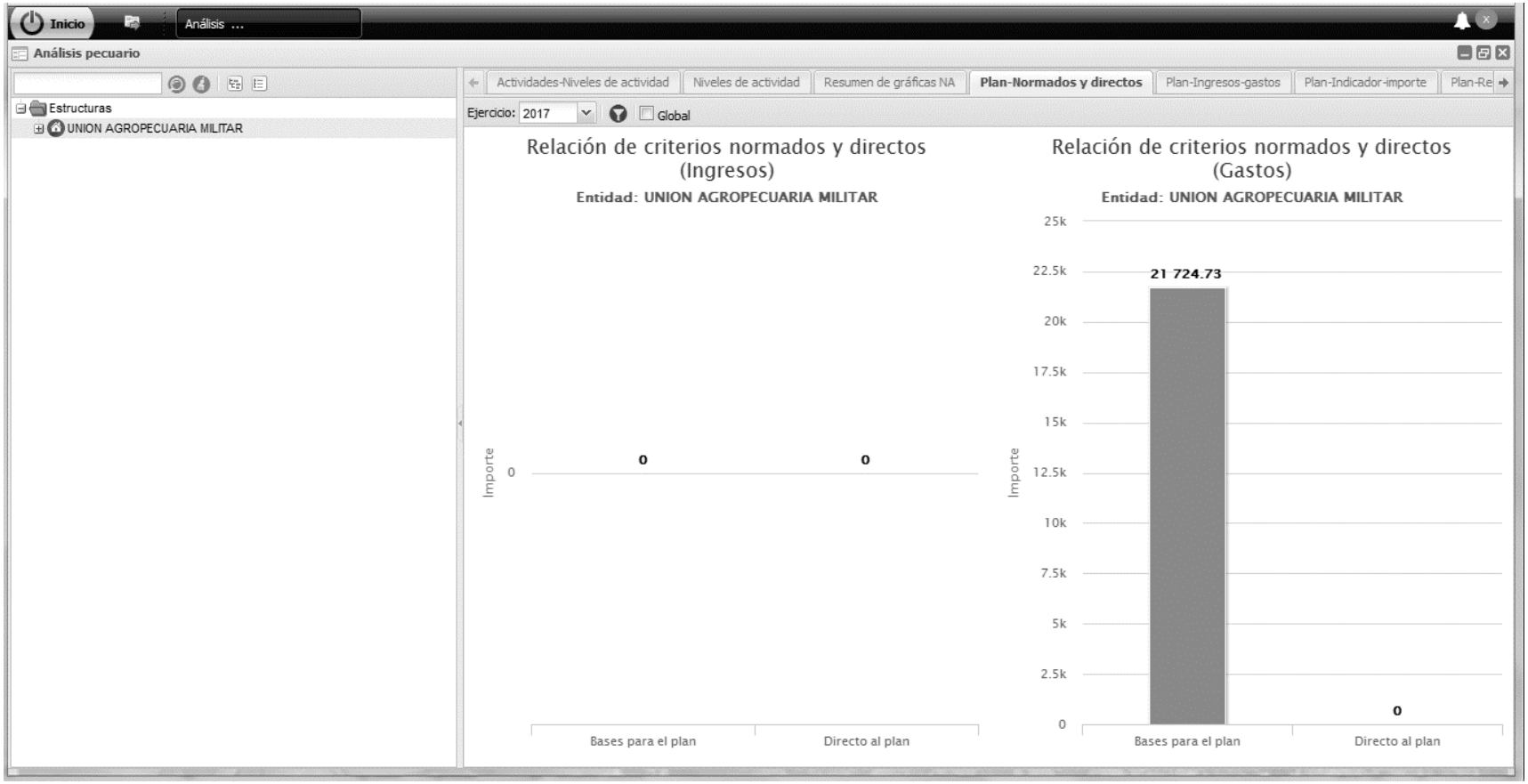
Prototipo de interfaz de usuario

RF_19: Visualizar niveles de actividad por ejercicio.

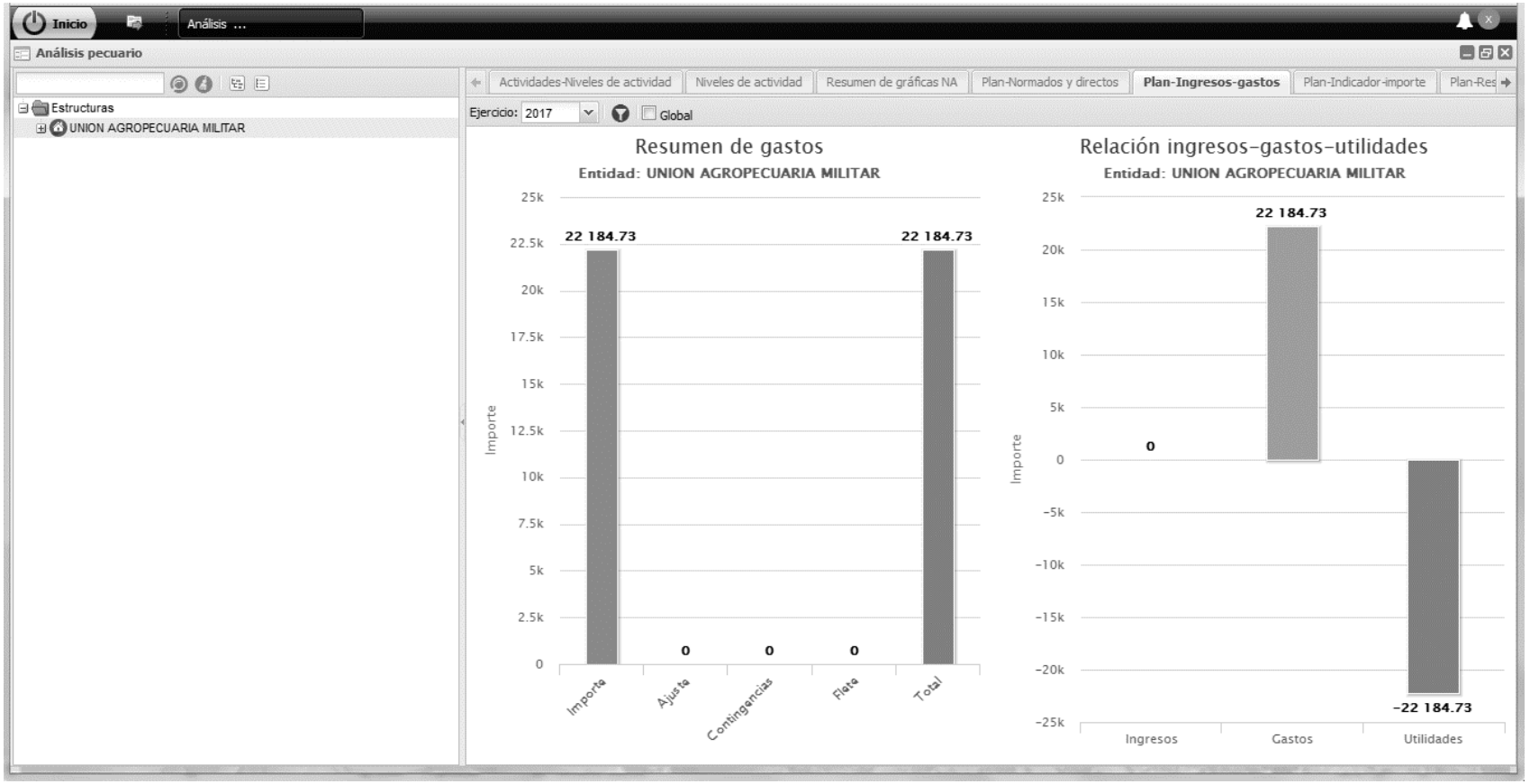
RF_20: Visualizar resumen de gráficas de niveles de actividad por ejercicio.



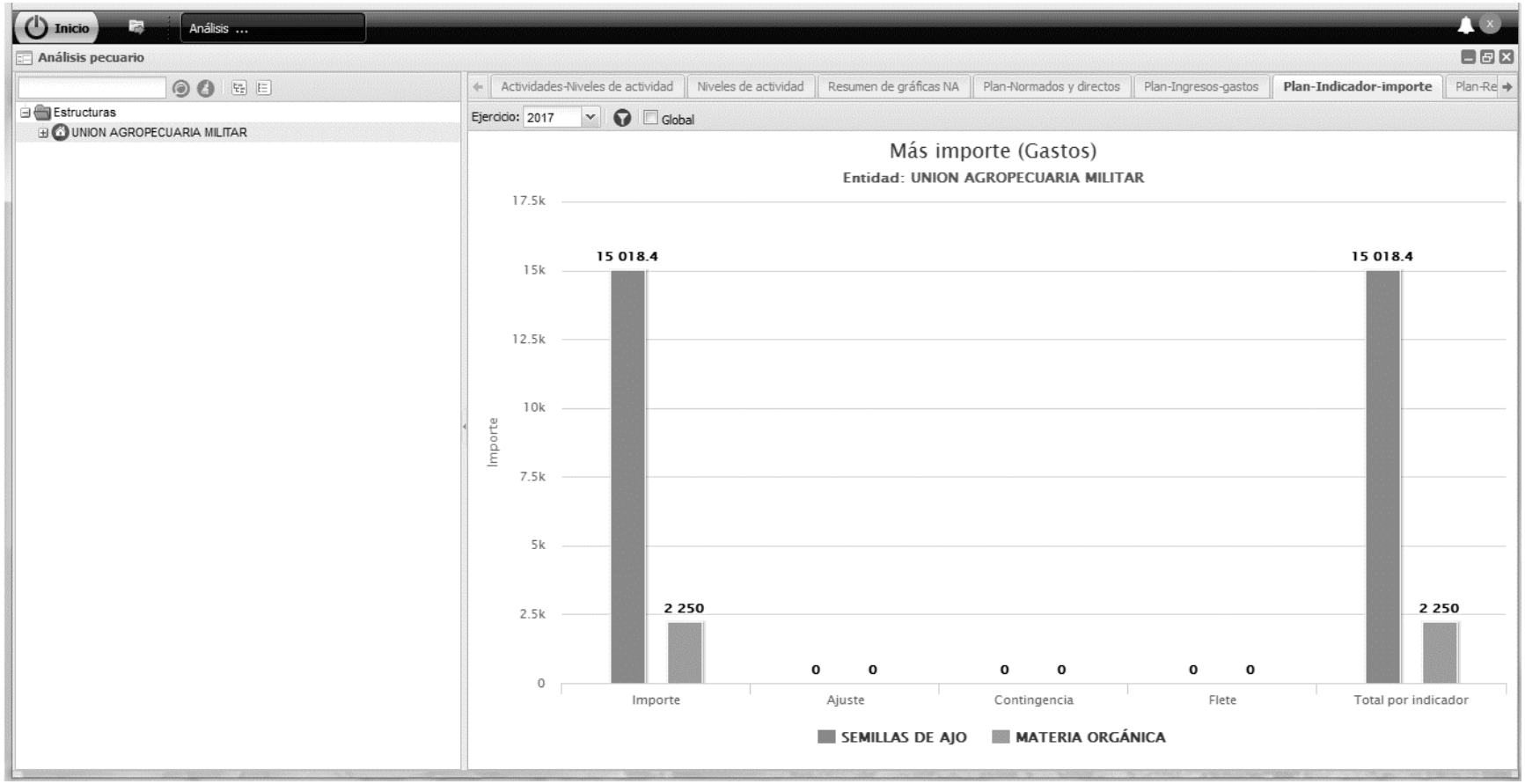
RF_21: Visualizar plan-normados y directos por ejercicio.



RF_22: Visualizar plan-ingresos-gastos por ejercicio.



RF_23: Visualizar plan-indicador-importe por ejercicio.



RF_24: Visualizar plan-resumen por centro de balance por ejercicio.


Inicio
Análisis ...
Análisis pecuario
Estructuras
UNION AGROPECUARIA MILITAR
Niveles de actividad
Niveles de actividad
Resumen de gráficas NA
Plan-Normados y directos
Plan-Ingresos-gastos
Plan-Indicador-importe
Plan-Resumen por CB
Ejercicio: 2017
Global
Relación ingresos-gastos por centro de balance
Entidad: UNION AGROPECUARIA MILITAR
25k
20k
15k
10k
5k
0
Importe
22 184.73
0
UNION AGROPECUARIA MILITAR
Ingresos
Gastos

Anexo 4

Descripción del caso de prueba Exportar importes de producciones.

Condiciones de ejecución:

- ✓ El usuario debe estar autenticado.
- ✓ Debe estar seleccionado una sola producción.
- ✓ Se debe seleccionar la opción **Inicio/ Planificación/ Planificación agropecuaria/ Producción/ Producciones/ Listar Producciones/ Exportar producciones.**

Tabla 4.4-1Exportar importes de producciones.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Exportar importes de producciones.	Con la realización de este requisito se exporta un Excel con el registro de los importes asociados a las producciones	EP 1.1: Exportar importes de producciones satisfactoriamente.	1. Se presiona clic izquierdo sobre una producción. 2. Se presiona clic izquierdo sobre el botón desplegable de exportar que se encuentra en el menú de herramientas del panel de producciones. 3. Se exporta un documento excel con los siguientes campos. 4. Entidad: entidad a la que pertenece el documento. 5. Denominación de la producción. 6. Producto. 7. NA: nivel de actividad.

			8. VPH. 9. VPE. 10. Fecha de inicio. 11. Fecha. 12. Fecha de fin. 13. Aprobado: si está aprobado o no. 14. Asegurado: si está asegurado o no. 15. Importe según la norma: importe de la producción.