Universidad de las Ciencias Informáticas
Facultad 6

Título: “Sistema para la Gestión de la Información de los Laboratorios de la Dirección de Calidad del Centro de Ingeniería Genética y Biotecnología: Diseño del Módulo de Microbiología. ”

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

Autora: Indira Darling O’Connor Núñez.

Tutora: Ing. Adisley Reyes Crespo.

Ciudad de la Habana, junio 2008

Año del 50 Aniversario de la Revolución

La sabiduría es un adorno en la prosperidad y un refugio en la adversidad.

Aristóteles

DECLARACIÓN DE AUTORÍA

Declaro ser autora de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año_____.

Indira Darling O'Connor Núñez

_____

Firma de la Autora

Ing. Adisley Reyes Crespo

_____

Firma de la Tutora

Dedicatoria

A mis padres Mariela y Orlando por su amor, apoyo, y comprensión.

A mis abuelos, por ser lo más grande que tengo en la vida Beba, Papi, Teresa, y Juan Bautista.

A mis hermanitos, que aunque nos peleemos siempre van a ser la alegría de mi corazón, un beso bien grande para Iramia, Marielita, Pipo y Orlandita.

A mis tíos, por su ternura y apoyo constante.

A mis primos que siempre estarán en mi corazón.

A mis amiguitas Rosa y Yusi, por ser más que unas hermanas para mi.

A mis amigos de la UCI: Ariel, Yurielkis Y Marisleydis, por estar siempre a mi lado y compartir conmigo lindos momentos en estos 5 años.

A mis eternas amigas Yaumara, Maylin y Yuleisis.

Y en especial a Toni.

Agradecimientos

Ha llegado el momento más importante de mi vida, hoy 30 de junio del año 2008 puedo decir que se ha hecho realidad el más grande de mis sueños: "Soy Ingeniera en Ciencias Informáticas", hoy puedo decir que no ha sido en vano todo el esfuerzo dedicado en estos 5 años de intensos estudios y trabajo. Es por ello que quiero agradecer aquellas lindas personas que de una forma u otra contribuyeron a que se hiciera realidad este sueño:

Primeramente a mis padres, que han sabido guiarme con su ejemplo, que aunque hoy no estén presenten quiero decirles que los amo.

A la Revolución Cubana y a nuestro Comandante en Jefe Fidel Castro Ruz por haber construido esta Universidad de excelencia y brindarnos la posibilidad de estudiar aquí y obtener los conocimientos necesarios para ser futuros profesionales dignos de este país.

A mi tutora Adisley, por su guía, apoyo constante y ayuda en todo momento.

A mis amigos Toni, Mari, Yuri, y Ariel, por su paciencia y apoyo cuando más los he necesitado.

A todos mis compañeros de la universidad por los momentos lindos que hemos compartidos.

A todos, muchas gracias.

Resumen

El presente trabajo forma parte del proyecto productivo LIMS de Calidad que se lleva a cabo en la Facultad de Bioinformática de la Universidad de las Ciencias Informáticas. Este proyecto consiste en la creación de un Sistema para la Gestión de la Información de los Laboratorios (LIMS) para contribuir con la gestión y control de la información que se genera en cada uno de los laboratorios de la Dirección de Calidad del Centro de Ingeniería Genética y Biotecnología (CIGB), información relacionada con el control de la calidad de los medicamentos que en el mismo se producen.

Dicho proyecto está compuesto por varios módulos, dentro de ellos se encuentra el Módulo de Microbiología, específicamente para este módulo un grupo de analistas de dicho de proyecto se encargaron de realizar el modelamiento de negocio e identificaron un grupo de requerimientos. Este trabajo está enfocado en la continuación de este proceso de informatización, diseñando el Módulo de Microbiología con vista a la implementación del mismo aplicando la metodología de desarrollo de software RUP.

PALABRAS CLAVES: LIMS, Microbiología, RUP

Índice

Índice de Figuras

Introducción

En la actualidad la gran cantidad de información que se almacena y se manipula en las organizaciones es la causa fundamental de que sean tan necesarios los sistemas de gestión de información. Se requiere que esta información se mantenga libre de errores, pues esto, es un elemento clave para la toma de decisiones.

Por ejemplo, la industria farmacéutica depende en gran medida de los laboratorios, los cuales se enfrentan a una gran carga de trabajo, y al mismo tiempo se les exige que la velocidad de generación de resultados elaborados y procesados sea cada vez mayor. Con el objetivo de incorporar al entorno de los laboratorios los beneficios y mejoras que aporta un sistema de información, surgen en la década del ´90 los Sistemas de Gestión de Información del Laboratorio (LIMS, del inglés Laboratory Information Management System). (1)

“Un LIMS proporciona un conjunto de herramientas basadas en Sistemas Informáticos que permiten la aplicación de técnicas de adquisición y gestión avanzada de la información producida en el laboratorio.”

En los últimos años, en Cuba se lleva a cabo un proyecto de informatización para la sociedad, obteniendo resultados satisfactorios en áreas importantes como la Salud, la Educación y la Investigación. El Centro de Ingeniería Genética y Biotecnología con 20 años de trayectoria de investigación científica y de obtención de productos reconocidos a nivel mundial, no se queda exento a estos avances tecnológicos, informatizándose exitosamente logrando grandes avances en el procesamiento de la información que este genera.

El trabajo realizado por el CIGB ha tenido gran impacto en la biomedicina, salud animal, mejoramiento vegetal y la bioindustria, ha desarrollado nuevas vacunas y fármacos para la salud humana que se encuentran actualmente en uso dentro del sistema de salud cubano, así como en diferentes países.

Todo lo referente a calidad del CIGB se gestiona en la Dirección de Calidad cuyas funciones se ponen de manifiesto a través de los **Departamentos de Control de la Calidad y Aseguramiento de la Calidad.** En ella se encuentran los grupos y laboratorios de análisis que son el origen de gran parte de los datos relacionados con los procedimientos de calidad. Por tanto, la incorporación a la misma de un sistema de información, adquiere una gran relevancia.

"El **Departamento de Aseguramiento de la Calidad** garantiza que se lleven a cabo las acciones planificadas y sistemáticas que son necesarias para proporcionar la confianza de que todos los productos y servicios satisfacen los requisitos de calidad establecidos. Vela por el cumplimiento de las Buenas Prácticas de Producción (BPP), Buenas Prácticas de Laboratorio (BPL) y Buenas Prácticas Clínicas (BPC). Este Departamento está compuesto por dos Secciones y dos grupos de trabajo:"

- ✓ **Sección de Mejoramiento de la Calidad (SMC)**
- ✓ **Sección de Inspección, Auditoría y Liberación de lotes**
  - Grupo de Inspección y Auditorias
  - Grupo de Liberación de Lotes
- ✓ **Grupo de Documentación**
- ✓ **Grupo de Metrología**

El **Departamento de Control de la Calidad** tiene entre sus funciones fundamentales las relacionadas con el muestreo, las especificaciones, los ensayos y la evaluación de la calidad de los productos que se generan en el Centro. Para el desempeño de las mismas, cuenta con la ayuda de dos grupos de trabajo y dos secciones:

- ✓ Grupo de Desarrollo
- ✓ Grupo de Recepción de Muestras y Manipulación de Expedientes
- ✓ Sección biológica compuesta por cinco laboratorios:
  - **Laboratorio de Microbiología** *(Ver Anexo 1)*
  - Laboratorio de Biología Molecular
  - Laboratorio de Ensayos Biológicos I
  - Laboratorio de Ensayos BiológicosII
  - Laboratorio de Inmunoquímica
- ✓ Sección físico-química compuesta por tres laboratorios:

- Laboratorio Análisis Químico
- Laboratorio de Cromatografía y Electroforesis
- Laboratorio de Sistemas Críticos

En el **Laboratorio de Microbiología (LM)** se realiza el control microbiológico de todos los procesos de producción, controlando la pureza y viabilidad de los bancos de células, controlando la posible contaminación en la etapa fermentativa, purificación y control de los Ingredientes Farmacéuticos Activos (IFA). Lleva a cabo el monitoreo microbiológico ambiental, control microbiológico de las muestras procedentes de los sistemas auxiliares. Se procesan además muestras de productos finales estériles y no estériles. Todos los medios de cultivo utilizados son controlados para promoción del crecimiento y esterilidad. El trabajo es realizado según las BPL.

El volumen de información que deben procesar los trabajadores del LM es considerable. Esta información plasmada en documentos debe ser revisada en la mayoría de los casos por los superiores, lo que genera una pérdida de tiempo significativo, cuando se requiere de un tiempo de respuesta mínimo. Y al mismo tiempo dificulta la elaboración de reportes rutinarios u ocasionales debido a la ineficiente forma de búsqueda existente.

Este problema no solo afecta al laboratorio de Microbiología, sino a cada uno de los laboratorios de la Dirección de Calidad del CIGB, por lo que se comienza a mediados del 2006 a desarrollar por parte del proyecto LIMS de Calidad un Sistema para la Gestión de la Información de los Laboratorios (LIMS), con el fin de agilizar el proceso de gestión y control de la información que se genera en cada uno de estos laboratorios. Dicho proyecto está constituido por diferentes módulos, entre los que se encuentra el Módulo de Microbiología. En este módulo un grupo de analistas realizaron los flujos de trabajo de Modelamiento del Negocio y Requerimientos, identificando 13 casos de uso del sistema con sus descripciones correspondientes y prototipos no funcionales. Con este trabajo se dará continuidad al proceso de desarrollo del Módulo de Microbiología.

De ahí que se identifique el **problema científico:** ¿Cómo transformar los requerimientos identificados para el Módulo de Microbiología del LIMS de Calidad del CIGB en elementos que puedan ser implementados?

El **objeto de estudio** correspondiente al problema científico es: El proceso de desarrollo de los sistemas de gestión de la información.

El **campo de acción** se enmarca en El proceso de diseño de los sistemas de gestión de la información.

El **objetivo general** del trabajo es: Diseñar el Módulo de Microbiología para el sistema de gestión de la información de los laboratorios de la Dirección de Calidad del CIGB.

Para el cumplimiento del objetivo propuesto se plantean las siguientes **tareas**:

- ✓ Revisión del Modelamiento del Negocio propuesto.
- ✓ Revisión del Modelo del Sistema propuesto.
- ✓ Realización de los CU-Diseño:
    - Diagramas de clases del Diseño
    - Diagramas de Secuencia del Diseño
- ✓ Elaboración de la Vista Lógica.
- ✓ Elaboración del diagrama de despliegue.
- ✓ Elaboración del mapa de navegación.
- ✓ Validación del Diseño.

La tesis está estructurada en Resumen, Introducción, Conclusiones, Recomendaciones, Bibliografía, Referencias Bibliográficas, Anexos y dos capítulos que abarcan los aspectos centrales distribuidos de la siguiente manera:

Capitulo 1: Fundamentación teórica: se realiza un estudio sobre los Sistemas de Gestión de la Información, abordando algunos aspectos de los Sistemas de Gestión de Información de los Laboratorios. Se realiza una breve descripción de las herramientas y metodología a utilizar, así como los artefactos a obtener definido por el Proceso Unificado de Desarrollo (RUP). Se abordan algunos aspectos significativos del framework seleccionado para el diseño y las principales características de los patrones de diseño a utilizar para el desarrollo del presente trabajo.

Capítulo 2. Diseño del Sistema: se describe brevemente la arquitectura del sistema, se presentan realizaciones de los casos de uso del diseño, con el objetivo de tener una mejor comprensión del

mismo. Y finalmente se realiza una validación del diseño, comparando el mismo con el código implementado por el equipo de programadores.

Capítulo 1. Fundamentación Teórica

En el presente capítulo se realiza un estudio de los Sistemas de Gestión de la Información, abordando algunos aspectos de los Sistemas para la Gestión de la Información de los Laboratorios. Se describe brevemente las herramientas y metodología a utilizar, así como los artefactos a obtener enmarcados en los roles a desempeñar definido por el Proceso Unificado de Desarrollo (RUP). Se abordan algunos aspectos significativos del framework seleccionado para el diseño y las principales características de los patrones de diseño a utilizar para el desarrollo del presente trabajo.

1.1 Sistema de Gestión de Información

La Gestión de Información es el proceso mediante el cual se obtienen, despliegan o utilizan recursos básicos (económicos, físicos, humanos, materiales) para manejar información dentro y para la sociedad a la que sirve. Tiene como elemento básico la gestión del ciclo de vida de este recurso y se desarrolla en cualquier organización. En particular, también se desarrolla en unidades especializadas que manejan este recurso en forma intensiva, llamadas unidades de información.

Funciones de la Gestión de Información: (2)

- ✓ Determinar necesidades internas de información, relativas a las funciones, actividades y procesos administrativos de la organización y a su satisfacción.
- ✓ Optimizar el flujo organizacional de la información y el nivel de la comunicación.
- ✓ Manejar eficientemente los recursos organizacionales de información, mejorar las inversiones sucesivas en los mismos y optimizar su aprovechamiento.
- ✓ Entrenar a los miembros de la organización en el manejo o la utilización de los recursos informacionales.
- ✓ Contribuir a modernizar u optimizar las actividades organizativas y los procesos administrativos relacionados con los mismos.
- ✓ Garantizar la calidad de los productos de la organización y asegurar su diseminación efectiva.
- ✓ Determinar las necesidades de información externa de la organización y satisfacerlas.

La información se ha convertido en el activo principal de las empresas, representando en la mayoría de los casos su principal ventaja estratégica. Es por ello que el desarrollo de sistemas de información se ve sometido actualmente a grandes exigencias en cuanto a productividad y calidad, y se hace

necesaria la aplicación de un nuevo enfoque en la producción del software, más cercano a una disciplina de ingeniería que a los hábitos y modos artesanales que, desafortunadamente, se han venido aplicando en más de una ocasión. El análisis y diseño de aplicaciones informáticas de gestión debe abordarse, por tanto, con técnicas y metodologías adecuadas, acompañadas por una precisa gestión de proyectos y una eficaz gestión de la calidad. Así mismo, es importante poder contar con el soporte de entornos y herramientas adecuadas, que faciliten la tarea del profesional informático y de los usuarios a la hora de desarrollar sistemas de información.

Los sistemas de gestión de Información para la gestión del conocimiento constituyen hoy una alternativa de imprescindible presencia en cada organización. Al permitir operar casi todos los activos tangibles e intangibles de la institución y llegar a convertirse en la herramienta integral de gerencia más cotizada y necesaria para alcanzar con éxito los resultados propuestos por la organización.

Los Sistemas de Gestión de la Información permiten: (2)

- ✓ Comprender la marcha de las organizaciones desde un enfoque analítico (donde queremos estar), evaluador (donde estamos) y creativo (donde podríamos estar).
- ✓ Develar oportunidades que merezcan ser explotadas y contrarrestar amenazas.
- ✓ Establecer los factores que resulten críticos y las necesidades asociadas al Sistema de Gestión de la Información.
- ✓ Estudiar el impacto de los Sistemas de Gestión de la Información en la posición del negocio y buscar nuevas oportunidades.

El Diseño de un Sistema de Gestión de Información precisa de: (2)

- ✓ Un análisis previo de las necesidades de Información de la organización,
- ✓ Un diagnóstico de la situación.
- ✓ Una auditoría de información que permita conocer los recursos de información disponibles y los que faltan, para qué y quienes lo utilizan, que valor se le añade en su uso, entre otros.

1.1.1 La web como sistema de información

La evolución de Internet como red de comunicación global y el surgimiento y desarrollo de la Web como servicio imprescindible para compartir información, creó un excelente espacio para la interacción

del hombre con la información hipertextual, a la vez que sentó las bases para el desarrollo de una herramienta integradora de los servicios existentes en Internet.

Los sitios Web, como expresión de sistemas de información, deben poseer los siguientes componentes: (3)

- ✓ Usuarios.
- ✓ Mecanismos de entrada y salida de la información.
- ✓ Almacenes de datos, información y conocimiento.
- ✓ Mecanismos de recuperación de información.

Actualmente, los sistemas de información se encuentran al alcance de las grandes masas de usuarios por medio de Internet; así se crean las bases de un nuevo modelo, en el que los usuarios interactúan directamente con los sistemas de información para satisfacer sus necesidades de información.

1.1.2 Sistemas de Gestión de Información de los Laboratorios (LIMS)

Un LIMS o "Laboratory Information Management System" es un programa de gestión de laboratorios que permite recoger, almacenar, calcular y gestionar datos en una amplia variedad de formas. Los LIMS representan una importante herramienta para la gestión global de un laboratorio en un entorno de calidad, agilizando temas de registro de datos primarios, archivo, trazabilidad, etc. y minimizando los errores debidos a la transferencia de información. (4)

En un entorno de calidad, el software utilizado por el laboratorio debe estar detalladamente documentado y debidamente validado como adecuado para su uso. El laboratorio también tiene que establecer y aplicar procedimientos para la protección de datos que garanticen su integridad, confidencialidad, almacenamiento, transmisión y proceso. (4)

Algunos beneficios específicos son: (4)

- ✓ Adquisición manual o automática de datos.
- ✓ Revisión y visualización de datos más completa, flexible y accesible.
- ✓ Generación más rápida y efectiva de informes.
- ✓ Información disponible cuando es requerida.

- ✓ Reduce o elimina los errores humanos.
- ✓ Mejora el tiempo de entrega de los resultados.
- ✓ Facilita la administración del laboratorio en operaciones como el estatus de lotes, progresos en el análisis, e identifica problemas.

Debido a estos beneficios, muchas empresas se han dedicado a la fabricación de estos sistemas para el manejo específico de sus propias necesidades. (4)

1.1.3 Ejemplos de proveedores de LIMS con más crecimiento en el mundo

STARLIMS

La Corporación STARLIMS es uno de los proveedores de LIMS con mayor crecimiento en el mundo, con más de 20 años de experiencia de LIMS, con un claro enfoque dirigido hacia la creación de fuertes relaciones con los clientes y el desarrollo de soluciones de calidad para la gestión de información de laboratorios. Establecidos en Hollywood, Florida, USA, la Corporación STARLIMS es considerada como la compañía de LIMS independiente mejor financiada.

Consolida procesos de laboratorio en una única y compatible plataforma con informes comprensibles, y con capacidades de supervisión sobre la red de trabajo. El resultado principal es resaltar el manejo de datos y la facilidad de compartirlos dentro del laboratorio y a través de la empresa. El software de STARLIMS es diseñado para una amplia variedad de laboratorios. La compañía sirve sobre 500 clientes en 40 países, incluyendo organizaciones gubernamentales, industriales y científicas.

MSC-LIMS

Es un sistema de gestión de información basado en Windows, desarrollado por Mountain States Consulting, LLC, una empresa productora de LIMS de los Estados Unidos.

MSC-LIMS es un producto flexible, seguro, diseñado para laboratorios de pequeño y mediano tamaño. Es compatible con versiones para uno o múltiples usuarios, para pequeños grupos de trabajo (de hasta veinte usuarios concurrentes). MSC-LIMS es conveniente para laboratorios que procesan hasta 75 000 y 300 000 muestras y análisis respectivamente en un año. La licencia de instalación y mantenimiento durante un año del LIMS en un puesto de trabajo cuesta 7000 dólares.

SQL LIMS

Con el sistema SQL LIMS se puede realizar el control de muestras, manejo de proceso, seguimiento del flujo de trabajo, análisis y almacenamiento de datos en el laboratorio. La tecnología LIMS de Applied Biosystems está construida en bases de datos Oracle y con una infraestructura de arquitectura abierta.

Esta tecnología es escalable, y está apoyada a través de la web por Applied Biosystems y mediante servicios encriptados. La última versión de SQL LIMS, la 4.1, es de amplia utilización en el campo de la genética.

LabSoft LIMS

Desde 1989 Computing Solutions, Inc. ha estado suministrando LabSoft LIMS exclusivamente a las industrias Químicas, Petroquímicas, de Bebidas y Alimenticias.

LabSoft LIMS es un sistema sumamente configurable. Puede construir una solución totalmente integrada de los datos del laboratorio logrando una calidad más alta y un coste reducido. Además ha demostrado ser eficiente, ya que desde 1989 que Computing Solutions, Inc. tiene clientes fijos, éste todavía opera en el 93 % de las empresas que disponen de él.

Labworks LIMS

EL sistema Labworks LIMS opera sobre diferentes plataformas según la cantidad de usuarios que lo utilicen y se adopta a todo tipo de empresas, independientemente de su envergadura. Su característica es modular, por lo que el sistema puede expandirse de acuerdo a las necesidades de trabajo.

Especificaciones:

- ✓ Ingreso de la muestra en forma manual, por códigos de barras o hand-held.
- ✓ Seguimiento.
- ✓ Comunicación directa con los instrumentos.
- ✓ Cálculo, tendencia, e historia.
- ✓ Informes automáticos (impresos, por mail).
- ✓ Programación de mantenimiento y calibración del instrumento.

1.1.4 ¿Por qué desarrollar un nuevo LIMS?

Luego de haber realizado una investigación sobre los Sistemas de Gestión de la Información de los Laboratorios (LIMS), y a pesar de las innumerables ventajas que presentan los mismos, se concluye que es más factible desarrollarlo que comprarlo, pues:

- ✓ Un gran por ciento de las empresas que desarrollan LIMS en el mundo son estadounidenses, por lo que debido al bloqueo económico existente en nuestro país, seria imposible adquirir uno de estos productos.
- ✓ Resulta menos costoso desarrollarlo, pues el sistema que necesita la Dirección de Calidad del CIGB siempre va a requerir de mantenimiento y actualización por ser tan extenso y variable.
- ✓ Las características del CIGB impiden poder comprar un LIMS para su uso particular, pues es variable el flujo de información existente y esto provoca que se necesite un sistema adaptable a las necesidades especificas del CIGB en función de lograr un Sistema de Calidad en el Centro.
- ✓ Al ser desarrollado en nuestro país se puede difundir a otros centros nacionales con características similares a las del CIGB para mejorar y agilizar los procesos relacionados con el control de la calidad de sus producciones.

1.2 Herramientas y metodologías utilizadas

Como se ha dicho anteriormente este trabajo forma parte del proyecto LIMS de Calidad de la Facultad de Bioinformática de la Universidad de las Ciencias Informáticas, este proyecto lleva 2 años desarrollando un LIMS para mejorar la gestión de la información de la Dirección de Calidad del CIGB, por lo que las herramientas y metodologías a utilizar ya fueron previamente definidas por este proyecto. A continuación se hace una breve descripción de estas herramientas y metodologías a utilizar.

1.2.1 Metodología de desarrollo de software

Proceso Unificado de Desarrollo (RUP)

La dificultad que presentan hoy en día los desarrolladores a la hora de realizar una aplicación de software es la necesidad de saber cómo organizar las actividades para cada desarrollador por separado y para el equipo, definir qué artefactos deben ser creados y contar con una serie de criterios

que permitan controlar y medir los productos que se obtienen, por lo tanto se necesita de una metodología capaz de dirigir estas actividades y así convertir los requisitos de los usuarios en un producto software.

La metodología de desarrollo que se emplea en el presente trabajo es el Proceso Unificado de Desarrollo. Este proceso está definido por tres aspectos fundamentales: dirigido por casos de uso, centrado en la arquitectura, e iterativo e incremental. Estas son las verdaderas características que definen a RUP, y cada una de ellas tiene igual importancia, pues la arquitectura proporciona la estructura sobre la cual guiar las iteraciones, mientras que los casos de uso definen los objetivos y dirigen el trabajo de cada iteración.

Las fases del ciclo de vida del software en el Proceso Unificado son:

1. Inicio (puesta en marcha): definir el alcance del proyecto y definir los casos de uso.
2. Elaboración (definición, análisis, diseño): definir las características y cimentar la arquitectura.
3. Construcción (implementación): crear el producto.
4. Transición (fin del proyecto y puesta en producción) transferir el producto a sus usuarios.

Dentro de estas fases se realizan varias actividades agrupadas en 9 flujos de trabajo principales, como se muestra a continuación:

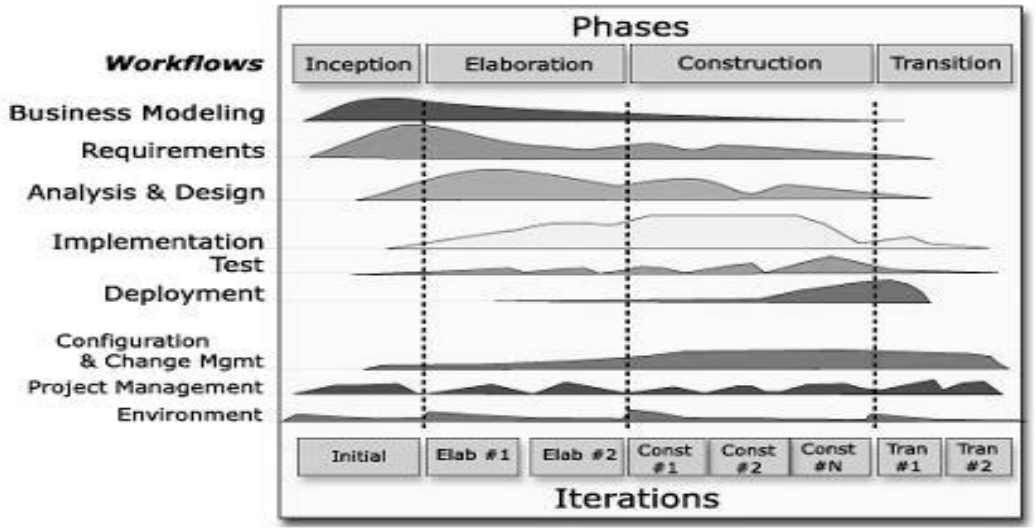

Fig 1: Fases y flujos de trabajo de la metodología RUP

RUP define los roles a jugar por cada miembro del equipo de desarrollo en cada una de las etapas por las que transcurre el sistema y facilita la comunicación entre los diferentes miembros del equipo de desarrollo. Emplea como lenguaje de modelado UML para la realización de los artefactos resultantes de cada flujo de trabajo.

En el desarrollo del presente trabajo de diploma se desempeñaron los roles de diseñador, diseñador de interfaz de usuario y arquitecto; los artefactos que se obtienen en el diseño propuesto por RUP para estos roles son: clases del diseño, realización de los caso de uso-diseño, paquetes de diseño, diagrama de despliegue, mapa de navegación, y la vista lógica. En el *Epígrafe 1.6* se hablará más claramente de estos artefactos a obtener a partir de los roles definidos por RUP:

1.2.2 Lenguaje de Modelado

UML

UML son las siglas de **Unified Modeling Languaje** (Lenguaje Unificado de Modelado), es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. (5)

Con UML se construyen sistemas por medio de conceptos orientados a objetos. Está compuesto por tres partes: elementos de construcción (tales como clases, objetos, mensajes), relaciones entre los

elementos (tales como asociación, generalización) y diagramas (por ejemplo, diagrama de actividad). *(Ver anexo 3)*

El UML es un lenguaje para construir modelos; no guía al desarrollador en la forma de realizar el análisis y diseño orientados a objetos ni le indica cual proceso de desarrollo adoptar. (6)

1.2.3 Herramienta CASE

Existen varias herramientas creadas para el desarrollo de la ingeniería de software, que tienen el fin de desarrollar programas utilizando técnicas de diseño y metodologías bien definidas, soportadas por herramientas automatizadas. Las herramientas CASE siglas en inglés de Computer Aided Software Engineering (Ingeniería de Software Asistida por Ordenador), son diversas aplicaciones informáticas que automatizan una parte del ciclo de desarrollo de software. Ejemplos de herramientas CASE, que se utilizan para el modelado de artefactos son: Umbrello, MagicDraw, Visual Paradigm y Rational Rose y Rational RequisitePro.

Visual Paradigm 6.1

Visual Paradigm es una potente herramienta CASE para visualizar y diseñar elementos de software. Utiliza UML como lenguaje de modelado, y soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, implementación, pruebas y despliegue. Tiene entre sus características el soporte de aplicaciones Web, y la integración con las siguientes herramientas Java: Eclipse/IBM WebSphere, JBuilder, NetBeans IDE, Oracle JDeveloper, BEA Weblogic. Con Visual Paradigm se puede generar código para diferentes lenguajes de programación, entre los que se encuentra PHP5, lenguaje con el que se programará la aplicación web.

Visual Paradigm ofrece: (7)

- ✓ Entorno de creación de diagramas para UML 2.0
- ✓ Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad
- ✓ Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- ✓ Capacidades de ingeniería directa (versión profesional) e inversa.
- ✓ Modelo y código que permanece sincronizado en todo el ciclo de desarrollo

- ✓ Disponibilidad de múltiples versiones, para cada necesidad.
- ✓ Disponibilidad de integrarse en los principales IDEs.
- ✓ Disponibilidad en múltiples plataformas

Después de culminado este epígrafe podemos concluir que para la modelación de la aplicación Web se utilizará RUP como metodología de desarrollo, ya que es un proceso bien definido, estructurado y adaptable a las características de cada proyecto en específico, UML como lenguaje de modelado para la realización de los artefactos resultantes en el diseño, y Visual Paradigm como herramienta CASE por ser multiplataforma y presentar una interfaz agradable al usuario.

1.3 Roles y Artefactos

Un rol define el comportamiento y responsabilidades de un individuo, o de un grupo de individuos trabajando juntos como un equipo. Para el desarrollo de este trabajo la autora va a desempeñar tres roles fundamentales: diseñador, diseñador de interfaz de usuario y arquitecto. A continuación se describen brevemente cada uno de estos roles y los artefactos a obtener por cada uno de ellos según la metodología RUP.

1.3.1 Artefactos a obtener por el Diseñador

El diseñador es el que va a estar encargado de los modelos que se construyen en el flujo de trabajo de análisis y diseño. Se encarga de diseñar una parte del sistema, dentro de las limitaciones de los requisitos, la arquitectura, y el proceso de desarrollo para el proyecto. Identifica y define las responsabilidades, operaciones, atributos, y las relaciones de elementos de diseño. El diseñador asegura que el diseño es consistente con la arquitectura de software, y se detalla a un punto que se pueda proceder con la implementación. Es el encargado de diseñar las partes del sistema, teniendo en cuenta los requerimientos, tanto funcionales como no funcionales y la arquitectura definida.

Artefactos a obtener por el Diseñador

Clases del Diseño:

Representa la descripción de un conjunto de objetos que comparten las mismas responsabilidades, comportamiento, atributos y relaciones.

Realización CU-Diseño:

- Diagramas de Clases del Diseño (extensión de UML para aplicaciones Web):

  Los diagramas de clases son los más utilizados en el modelado de sistemas orientados a objetos. Un diagrama de clases muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Los diagramas de clases se utilizan para modelar la vista de diseño estática de un sistema. Principalmente, esto incluye modelar el vocabulario del sistema, modelar las colaboraciones o modelar esquemas. Los diagramas de clases también son la base para un par de diagramas relacionados: los diagramas de componentes y los diagramas de despliegue. Los diagramas de clases son importantes no sólo para visualizar, especificar y documentar modelos estructurales, sino también para construir sistemas ejecutables, aplicando ingeniería directa e inversa.

- Diagramas de Interacción: Diagramas de Secuencia del Diseño.

  Los diagramas de interacción se utilizan para modelar los aspectos dinámicos de un sistema. Consta de un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos.

  Cuando ya tenemos un esquema de las clases necesarias para realizar un caso de uso, debemos describir como interactúan sus correspondientes objetos del diseño, para ello se realiza un diagrama de colaboración o uno de secuencia. Siendo este último el mas recomendado para la actividad del diseño que representa el orden e interacción entre los objetos. Un diagrama de secuencia debe trabajar con las instancias de los actores, instancias de las clases, o sea los objetos y las transmisiones de mensajes entre ellos.

Paquetes del diseño:

Un paquete de diseño es un conjunto de clases, las relaciones, las realizaciones de casos de uso, diagramas y otros paquetes .Se utiliza para estructurar el modelo de diseño dividiéndolo en partes más pequeñas.

1.3.2 Artefactos a obtener por el Arquitecto

El arquitecto es el responsable de la arquitectura del Software, que incluye las decisiones técnicas claves que limitan el diseño global y la aplicación del proyecto. (8)

Artefactos a obtener por el Arquitecto

Diagrama de Despliegue

El modelo de despliegue muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los links de comunicación entre ellos, y las instancias de los componentes y objetos que residen en ellos. Su propósito es capturar la configuración de los elementos de procesamiento, y las conexiones entre estos elementos en el sistema. El modelo consiste en uno o mas nodos (elementos de procesamiento con al menos un procesador, memoria, y posiblemente otros dispositivos), dispositivos (nodos estereotipados con una capacidad de procesamiento en el nivel modelado de abstracción), y conectores, entre nodos, y entre nodos y dispositivos. El modelo de despliegue también mapea procesos dentro de estos elementos de procesamiento, permitiendo la distribución del comportamiento a través de los nodos que son representados.

1.3.3 Artefactos a obtener por el Diseñador de Interfaz de Usuario

El diseñador de Interfaz de Usuario Coordina el diseño de la interfaz de usuario. Está también involucrado en la recopilación de los requisitos y el diseño de los prototipos candidatos de la interfaz de usuario para satisfacer esas necesidades.

Artefactos a obtener por el Diseñador de Interfaz de Usuario

Mapa de Navegación

El mapa de navegación expresa la estructura de los elementos de la interfaz de usuario en el sistema, junto con sus caminos de navegación potenciales.

1.4 Tecnologías para el Desarrollo

Con el objetivo de optimizar el desarrollo de la aplicación Web, se desarrollará la misma utilizando el framewor Symfony 1.0.11 y PHP 5 como lenguaje de programación.

1.4.1 Framework Symfony

Los *frameworks* están diseñados con el propósito de facilitar el desarrollo de software. Es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado.

Symfony es un framework desarrollado por Fabien Potencier, programado en PHP 5 y enfocado a optimizar el desarrollo de aplicaciones web en el mismo lenguaje de programación.

Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

Proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas.

Características de Symfony (9)

- ✓ Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares).
- ✓ Independiente del sistema gestor de bases de dato).
- ✓ Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- ✓ Basado en la premisa de “convenir en vez de configurar”, en la que el desarrollador solo debe configurar aquello que no es convencional.
- ✓ Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- ✓ Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- ✓ Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.
- ✓ Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros

1.4.2 Lenguaje de programación: Personal Home Page (PHP)

PHP es un lenguaje de programación usado generalmente para la creación de contenido para sitios Web. PHP es un acrónimo recurrente que significa "PHP Hypertext Pre-processor" (inicialmente PHP Tools, o Personal Home Page Tools), y se trata de un lenguaje interpretado usado para la creación de

aplicaciones para servidores, o creación de contenido dinámico para sitios Web. PHP5 es la última versión de dicho lenguaje; esta incorpora el paradigma de la Programación Orientada a Objetos (POO) e incluye todas las ventajas que provee el nuevo Zend Engine 2 como:

- ✓ Mejor soporte para la Programación Orientada a Objetos (POO), que en versiones anteriores era extremadamente rudimentario, con PHP Data Objects.
- ✓ Mejoras de rendimiento.
- ✓ Mejor soporte para MySQL con extensión completamente rescrita.
- ✓ Mejor soporte a XML (XPath, DOM, etc).
- ✓ Soporte nativo para SQLite.
- ✓ Soporte integrado para SOAP.
- ✓ Iteradores de datos.
- ✓ Excepciones de errores.

Los principales usos del PHP son los siguientes:

- ✓ Programación de páginas web dinámicas, habitualmente en combinación con el motor de base datos MySQL, aunque cuenta con soporte nativo para otros motores, incluyendo el estándar ODBC, lo que amplía en gran medida sus posibilidades de conexión.
- ✓ Programación en consola, al estilo de Perl o Shell scripting.
- ✓ Creación de aplicaciones gráficas independientes del navegador, por medio de la combinación de PHP y Qt/GTK+, lo que permite desarrollar aplicaciones de escritorio en los sistemas operativos en los que está soportado.

Ventajas

- ✓ Es un lenguaje multiplataforma.
- ✓ Alto rendimiento. PHP es muy eficiente. Mediante el uso de un único servidor, puede servir millones de accesos al día. Los indicadores comparativos de rendimiento publicados por Zend

Technologies (http://www.zend.com) muestran que PHP supera ampliamente a sus competidores en esta faceta.

- ✓ Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad como MySQL, PostgreSQL, mSQL, Oracle, dbm, filepro, Hyperwave, Informix, InterBase y Sybase, entre otras. El uso de ODBC (del inglés Open Database Connectivity Standard, Estándar de conectividad abierta de base de datos) permite establecer una conexión a cualquier base de datos que suministre un controlador ODBC.
- ✓ Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados extensiones).
- ✓ Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- ✓ Es libre y Open Source, por lo que se presenta como una alternativa de fácil acceso para todos además no se está forzado a pagar actualizaciones anuales para tener una versión que funcione. Se puede modificar el código y copiar.
- ✓ Facilidad de aprendizaje y uso. La sintaxis de PHP se basa en otros lenguajes de programación, principalmente en C y Perl. Si ya conoce un lenguaje de tipo C como C++ o Java, no tardará nada en utilizar PHP de manera productiva.
- ✓ Permite las técnicas de Programación Orientada a Objetos.
- ✓ Biblioteca nativa de funciones sumamente amplia e incluida.
- ✓ No requiere definición de tipos de variables.
- ✓ Tiene manejo de excepciones.

El framework Symfony y el lenguaje de programación PHP 5, son tecnologías seleccionadas por el proyecto para el desarrollo de la aplicación Web. Ya que Symfony es un framework enfocado a optimizar el desarrollo de aplicaciones Web utilizando PHP 5 como lenguaje de programación.

1.5 Patrones

El objetivo de los patrones es crear un lenguaje común a una comunidad de desarrolladores para comunicar experiencia sobre los problemas y sus soluciones. Un patrón describe un problema que ocurre una y otra vez en nuestro entorno y describe también el núcleo de la solución al problema, de forma que pueda utilizarse esta solución un millón de veces, sin tener que hacer dos veces lo mismo.

1.5.1 Patrones de Arquitectura

Un patrón de arquitectura de software describe un problema particular y recurrente del diseño, que surge en un contexto específico, y presenta un esquema genérico y probado de su solución. (10)

Para el desarrollo de la aplicación se utilizará el *framework Symfony* el cual está basado en el patrón arquitectónico Modelo-Vista-Controlador (MVC). Este patrón está formado por tres niveles: (9)

- ✓ El modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- ✓ La vista transforma el modelo en una página web que permite al usuario interactuar con ella.
- ✓ El controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

La arquitectura MVC *(Ver Anexo 4)* separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. Si por ejemplo una misma aplicación debe ejecutarse tanto en un navegador estándar como en un navegador de un dispositivo móvil, solamente es necesario crear una vista nueva para cada dispositivo; manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, etc.). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación.

1.5.2 Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular, identifica: Clases, Instancias, Roles, Colaboraciones y la distribución de responsabilidades**.**

La arquitectura MVC implementa internamente los patrones de asignación de responsabilidades GRASP, como son por ejemplo:

Alta Cohesión:

Surge ante la problemática de cómo mantener la complejidad dentro de límites manejables y su solución es asignar una responsabilidad de modo que la cohesión siga siendo alta.

En la perspectiva del diseño orientado a objetos, la cohesión (o, más exactamente, la cohesión funcional) es una medida de cuan relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme.

Una clase con baja cohesión hace muchas cosas no afines o un trabajo excesivo. No conviene este tipo de clases pues presentan los siguientes problemas:

- ✓ Son difíciles de comprender
- ✓ Son difíciles de reutilizar
- ✓ Son difíciles de conservar
- ✓ Son delicadas: las afectan constantemente los cambios

Las clases con baja cohesión a menudo representan un alto grado de abstracción o han asumido responsabilidades que deberían haber delegado a otros objetos.

Ventajas del patrón Alta Cohesión:

- ✓ Mejoran la claridad y la facilidad con que se entiende el diseño.
- ✓ Se simplifican el mantenimiento y las mejoras en funcionalidad.
- ✓ A menudo se genera un bajo acoplamiento.
- ✓ La ventaja de una gran funcionalidad soporta una mayor capacidad de reutilización, porque una clase muy cohesiva puede destinarse a un propósito muy específico.

Bajo Acoplamiento:

Este patrón surge ante la problemática de cómo dar soporte a una dependencia escasa y a un aumento de la reutilización y su solución es asignar una responsabilidad con vista a mantener el bajo acoplamiento.

El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras. Una clase con alto (o fuerte) acoplamiento recurre a muchas otras. Este tipo de clases no es conveniente, pues presentan los siguientes problemas:

- ✓ Los cambios de las clases afines ocasionan cambios locales.
- ✓ Son más difíciles de entender cuando están aisladas.
- ✓ Son más difíciles de reutilizar porque se requiere la presencia de otras clases de las que dependen.

Ventajas del patrón Bajo Acoplamiento

- ✓ No se afectan por cambios de otros componentes
- ✓ Fáciles de entender por separado
- ✓ Fáciles de reutilizar

Experto:

Este patrón surge ante la problemática: ¿Cuál es el principio fundamental en virtud del cual se asignan las responsabilidades en el diseño orientado a objetos? Y su solución es asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.

Experto es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos. Con él no se pretende designar una idea oscura ni extraña; expresa simplemente la "intuición" de que los objetos hacen cosas relacionadas con la información que poseen.

Ventajas del patrón Experto

- ✓ Se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento.
- ✓ El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clases sencillas y más cohesivas que son más fáciles de comprender y de mantener. Así se brinda soporte a una alta cohesión.

Creador:

Surge ante problemática: ¿Quién debería ser responsable de crear una nueva instancia de alguna clase?, y su solución es asignarle a la clase B la responsabilidad de crear una instancia de clase A en uno de los siguientes casos:

- ✓ B agrega los objetos A.
- ✓ B contiene los objetos A.
- ✓ B registra las instancias de los objetos A.
- ✓ B utiliza específicamente los objetos A.
- ✓ B tiene los datos de inicialización que serán transmitidos a A cuando este objeto sea creado (así que B es un Experto respecto a la creación de A).

B es un creador de los objetos A.

Si existe más de una opción, prefiera la clase B que agregue o contenga la clase A.

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento.

Ventajas del patrón Creador

- ✓ Se brinda soporte a un bajo acoplamiento, lo cual supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización. Es probable que el acoplamiento no aumente, pues la clase creada tiende a ser visible a la clase creador, debido a las asociaciones actuales que nos llevaron a elegirla como el parámetro adecuado.

Controlador:

El patrón Controlador surge ante la problemática: ¿Quién debería encargarse de atender un evento del sistema?, y su solución es asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente una de las siguientes opciones:

- ✓ El "sistema" global (controlador de fachada).
- ✓ La empresa u organización global (controlador de fachada).

- ✓ Algo en el mundo real que es activo (por ejemplo, el papel de una persona) y que pueda participar en la tarea (controlador de tareas).
- ✓ Un manejador artificial de todos los eventos del sistema de un caso de uso, generalmente denominados "Manejador<NombreCasodeUso>" (controlador de casos de uso).

El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado.

MVC engloba otro patrón de diseño (GOF): (11)

Patron Decorator

A veces se desea adicionar responsabilidades a un objeto pero no a toda la clase. Las responsabilidades se pueden adicionar por medio de los mecanismos de Herencia, pero este mecanismo no es flexible porque la responsabilidad es adicionada estáticamente. La solución flexible es la de rodear el objeto con otro objeto que es el que adición la nueva responsabilidad. Este nuevo objeto es el Decorator. *(Ver Anexo 4)*

El Decorator se debe usar para: (12)

- ✓ Adicionar responsabilidades a objetos individuales dinámicamente sin afectar otros objetos.
- ✓ Para agregar responsabilidades que pueden ser retiradas
- ✓ Cuando no es práctico adicionar responsabilidades por medio de la herencia.

Consecuencias: (12)

- ✓ *Ventajas*
  - Es más flexible que la Herencia estática.
  - Permite que la adición de nuevas responsabilidades (nuevas clases de Decorators) independiente de las clases los Objetos que ellas extienden.
- ✓ *Desventajas*
  - Un Decorator y su Component no son idénticos. Desde el punto de vista de la identidad de los objetos, un DecoratorComponent no es idéntico al Component. Por esto no se puede confiar en la identidad de los objetos cuando se usan Decorators
  - El patrón Decorator hace que hayan muchos objetos pequeños que son muy parecidos.

1.5.3 Aplicación de los patrones de diseño

Los patrones de diseño mencionados en el epígrafe anterior son utilizados en la modelación de la aplicación Web. A continuación se explica detalladamente cual es el uso de cada uno de ellos dentro del desarrollo de la aplicación Web.

Bajo Acoplamiento:

Se utiliza para desacoplar las vistas de los modelos y desacoplar los modelos de la forma en que se muestran e ingresan los datos, se pone en evidencia el uso de este patrón por ejemplo, en la clase Actions quien hereda solamente de sfActions lográndo un bajo acoplamiento de clases.

Alta Cohesión:

Se evidencia en casa elemento diseñado ya que cada cual está altamente especializado en su tarea por ejemplo la vista en mostrar los datos al usuario, el controlador en las entradas y el modelo en su objetivo del negocio. Symfony admite la asignación de responsabilidades con una alta cohesión, como ejemplo de ellos se tiene la clase Actions la cual es responsable de definir las acciones para las plantillas y colaborar con otras para realizar diferentes operaciones (instanciar objetos y acceder a las properties), es decir, que está formada por diferentes funcionalidades las que se encuentran estrechamente relacionadas lo que hace posible que el software sea flexible frente a grandes cambios.

Experto:

Es uno de los más utilizados, por ejemplo: en Propel, el cual es la librería externa que utiliza Symfony para realizar su capa de abstracción en el modelo, se encapsula toda la lógica de los datos y son generadas las clases con todas las funcionalidades comunes de las entidades.

Controlador:

Se pone de manifiesto en el controlador frontal pues este maneja todas las peticiones Web que se realizan al sistema, ya que es el punto de entrada único de toda la aplicación en un entorno determinado. Cuando este recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL entrada por el usuario.

Creador:

Se evidencia en todas las clases Peer contenidas en el modelo, en ellas se encuentran las acciones definidas para dar respuesta a cada funcionalidad del sistema. Cada clase Peer es la encargada de

crear el objeto de su respectiva clase entidad, evidenciando de este modo que cada clase Peer es "creador" de su clase entidad correspondiente.

Patron Decorator

Debido a la arquitectura basada en el patrón MVC y a la composición que tienen cada una de estas partes en Symfony se hará uso implícitamente del patrón *Decorator*, específicamente en la Vista ya que esta está compuesta por diversas partes dentro de las que está la plantilla y el *layout*. Este último es el que almacena el código común para todas las páginas de la aplicación, por lo que el contenido de la plantilla se integra a él, es decir, es el que decora la plantilla. *(Ver Anexo 5)*

1.6 Conclusiones

Con el desarrollo del presente capitulo se realizó un estudio sobre el proceso de desarrollo de los sistemas de gestión de información. Se abarcó además las características fundamentales del framework Symfony así como el patrón de arquitectura MVC que sustenta a este framework. Se describieron brevemente las herramientas y metodologías a utilizar para el desarrollo de este trabajo. Se caracterizaron los roles a desempeñar, así como sus respectivos artefactos a obtener. Y además se mencionan y se describen los patrones fundamentales (GOF y GRASP) a utilizarse en el desarrollo de este trabajo.

Capítulo 2: Diseño

En este capítulo se describe brevemente la arquitectura del sistema, se presentan además las realizaciones de los casos de uso del diseño, con el objetivo de tener una mejor comprensión del mismo. Y finalmente se realiza una validación del diseño, comparando el mismo con el código implementado por el equipo de programadores.

2.1 Levantamiento de requisitos propuesto

El equipo de analistas del proyecto realizó el levantamiento de requisitos del Módulo de Microbiología, el cual presenta como características 91 requisitos funcionales agrupados en 13 casos de uso arquitectónicamente significativos, a los cuales se les realizaron las descripciones textuales y prototipos de interfaz de usuario. A continuación se muestran los diagramas de casos de usos del sistema elaborado en esta etapa.

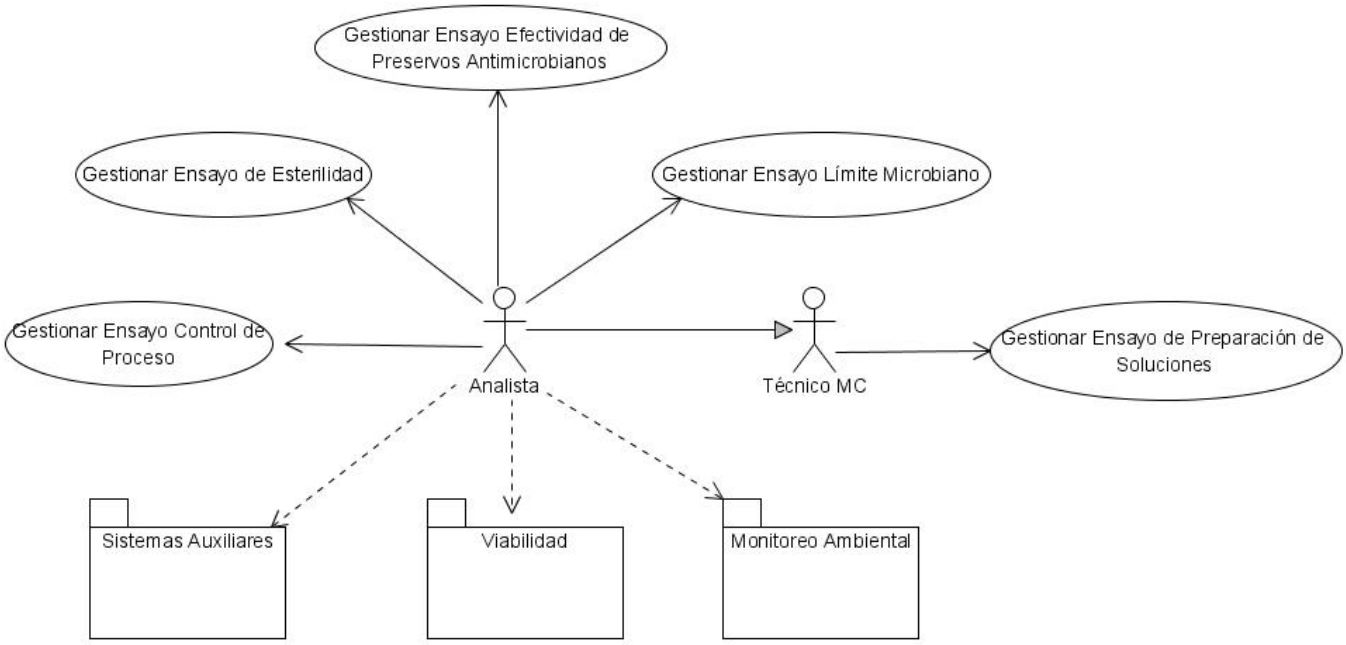

Fig 2: Diagrama de caso de uso del Sistema del Módulo de Microbiología.

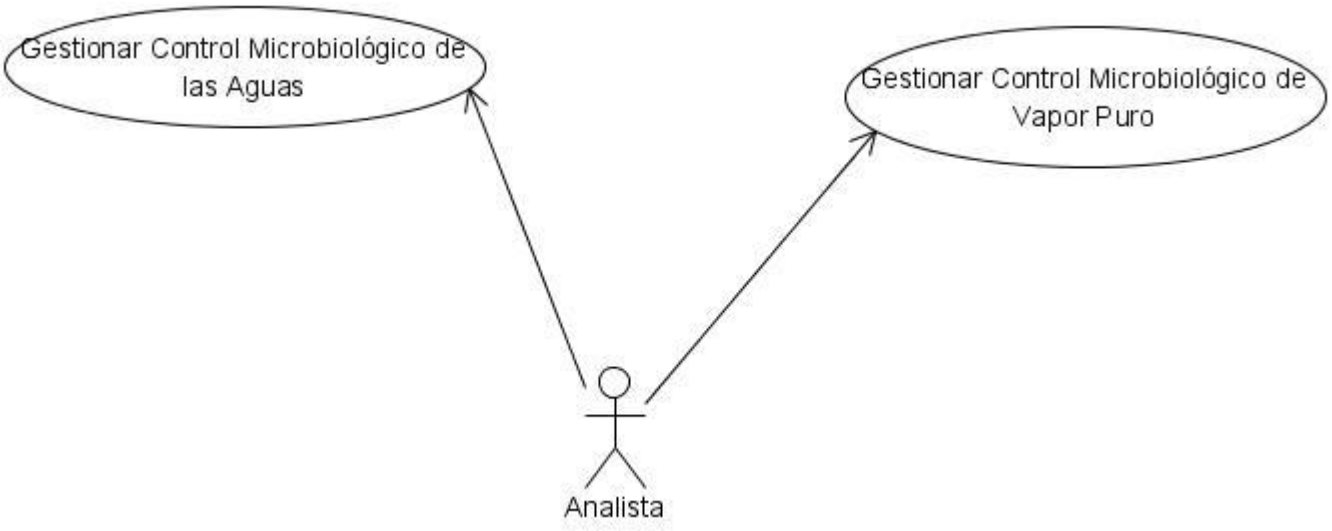

Fig 3: Paquete Sistemas Auxiliares

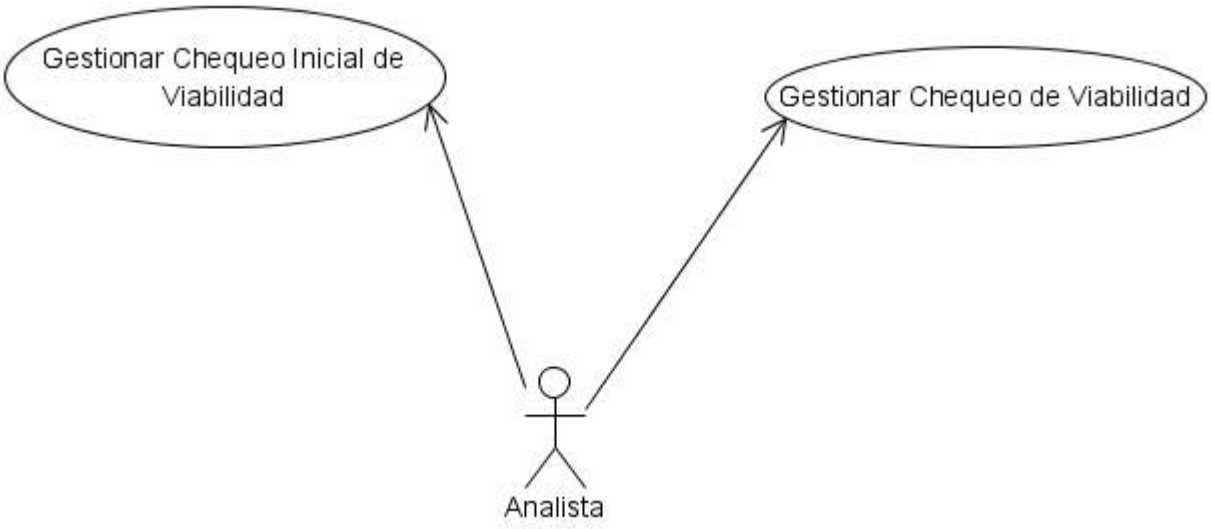

Fig 4: Paquete Viabilidad

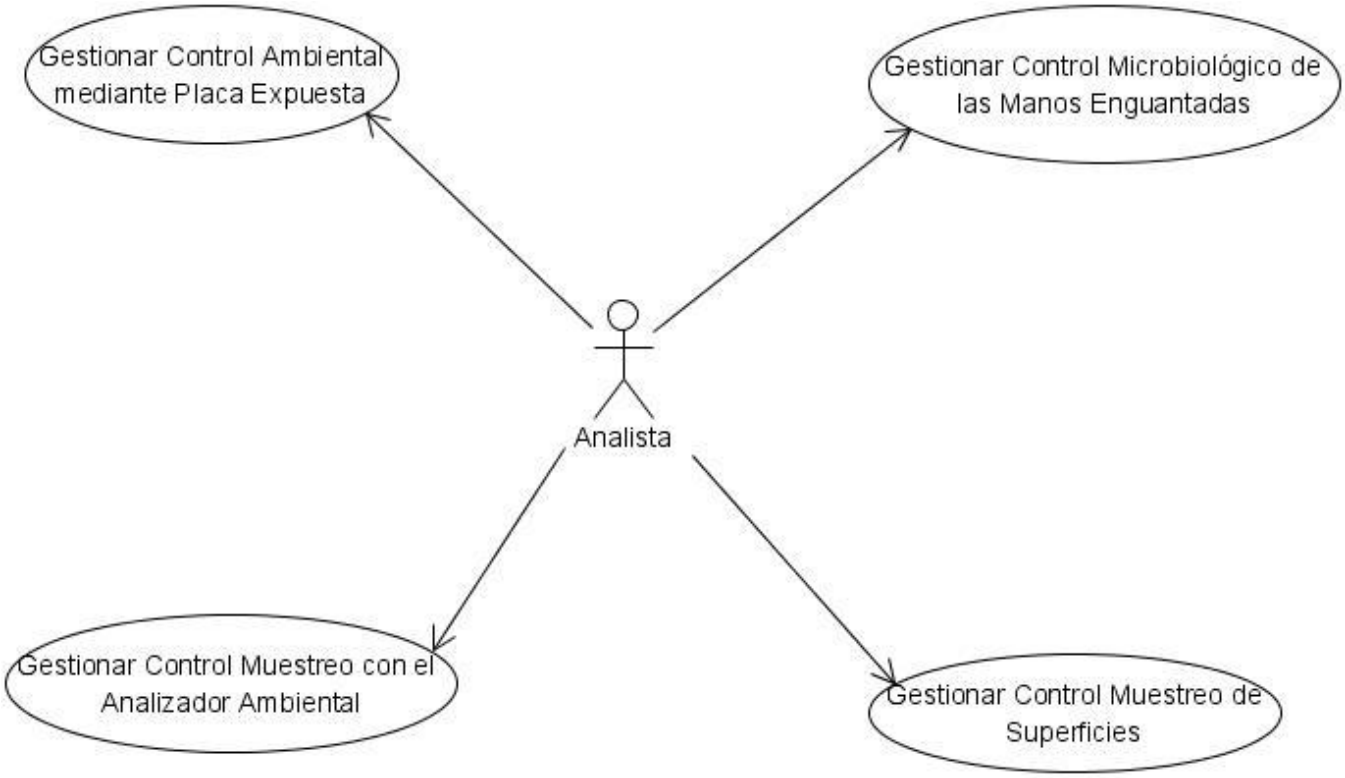

Fig 5: Paquete Monitoreo Ambiental

Además se identificaron 26 requisitos no funcionales que expresan las cualidades o características que debe tener el sistema. Todos los artefactos obtenidos en esta etapa sirven de entrada para la realización del diseño del Módulo de Microbiología.

2.2 Referencia a la arquitectura del sistema

A continuación se realizará una breve referencia a la arquitectura a utilizar para el desarrollo del Sistema para la Gestión de la Información de los Laboratorios a través de la Vista Lógica y la Vista de Despliegue.

2.2.1 Vista Lógica

Esta sección tratará la vista lógica del proceso de desarrollo. Esta vista está representada por un subconjunto del artefacto Modelo de diseño, la cual representa los elementos de diseño más

importantes para la arquitectura del sistema. Para estructurar el sistema, se identificaron en este módulo 16 paquetes a partir de los casos de uso arquitectónicamente significativos, los cuales constituyen la totalidad de los casos de uso existentes. Estos paquetes a su vez pueden contener un conjunto de paquetes, como por ejemplo tenemos a los paquetes: Viabilidad, Sistemas Auxiliares y Monitoreo Ambiental. Cada paquete está compuesto por un caso de uso y la realización del mismo (diagrama de clases del diseño, y diagramas de secuencia del diseño).

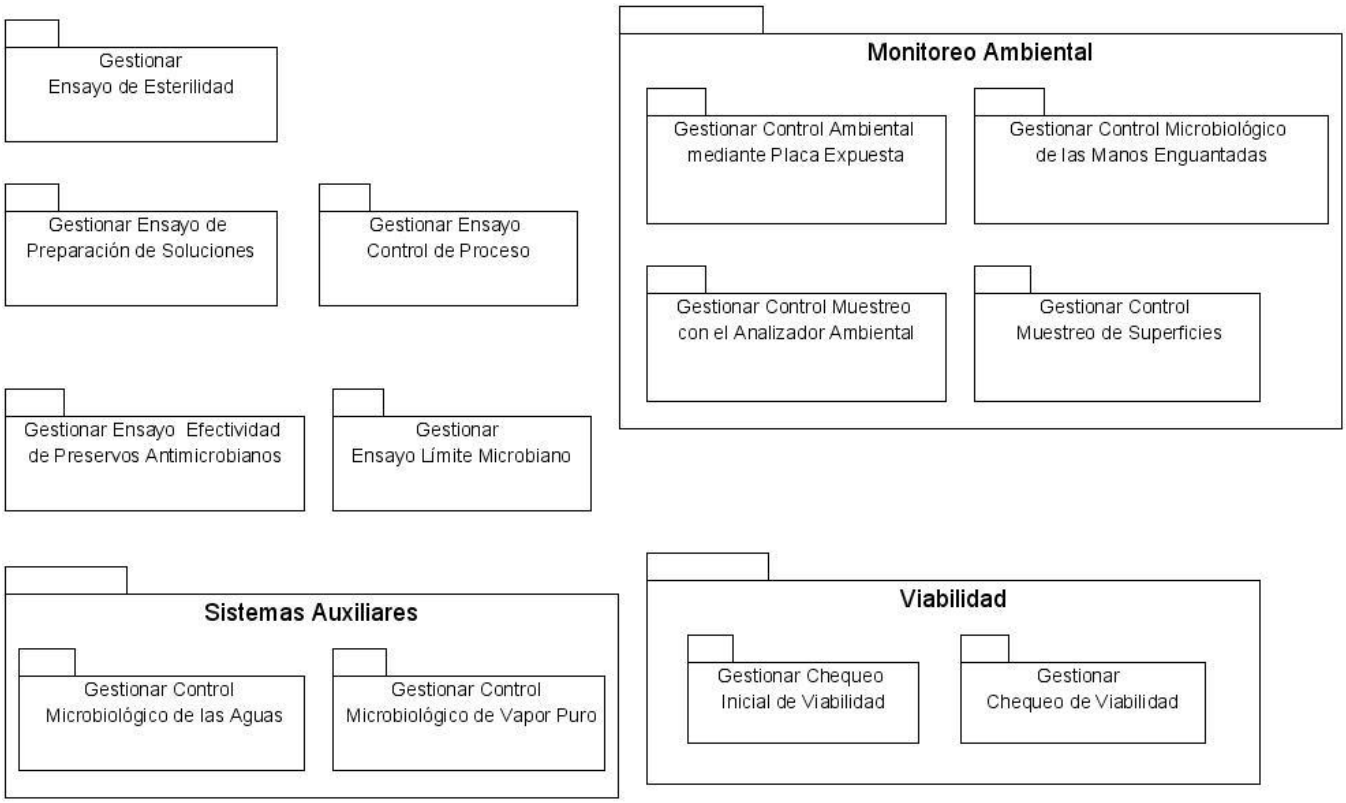

Fig 6: Vista Lógica

2.2.2 Vista Despliegue

La Vista de Despliegue suministra una base para la comprensión de la distribución física de un sistema a través de nodos, los cuales pueden ser elementos de procesamientos o dispositivos que son importantes para la arquitectura del sistema.

La vista de despliegue va estar representada mediante un diagrama de despliegue. La aplicación va a quedar conformada de la siguiente manera:

Un servidor de aplicaciones al cual se le van a conectar las máquinas clientes, estas tendrán incorporadas una impresora para obtener en copia dura las planillas y documentos importantes, además de un escáner para capturar información no digital. Contendrá además un servidor de base de datos para almacenar toda la información del sistema.

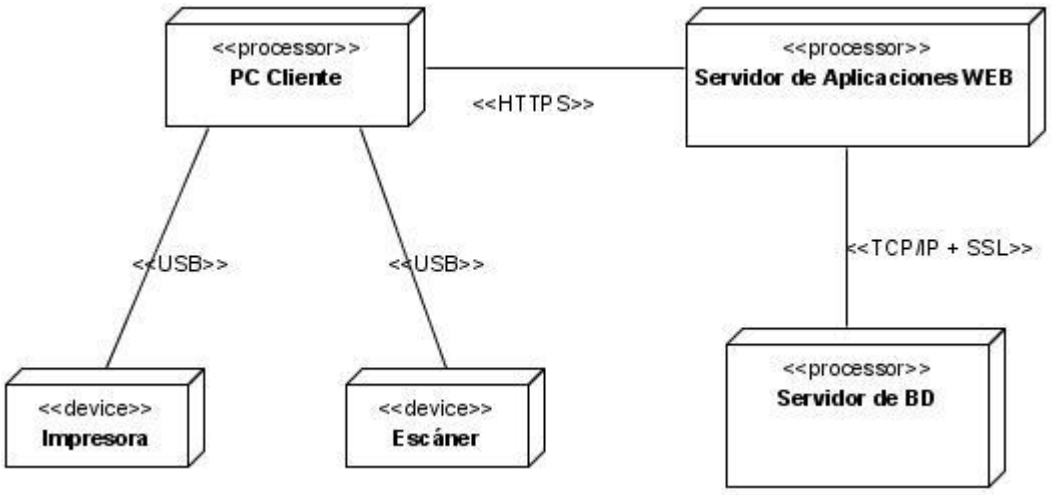

Fig 7: Diagrama de Despliegue

2.3 Mapa de Navegación

Un mapa de navegación es la representación gráfica de la organización de la información de una estructura web. Permite elaborar escenarios de comportamiento de los usuarios y expresa todas las relaciones de jerarquía y secuencia. (13)

La importancia de elaborar un mapa de navegación del sitio web radica en la comprensión del orden de presentación de las páginas web y la flexibilidad de moverse entre ellas (hipervínculos).

En el diseño del mapa de navegación se debe seleccionar la página de entrada al sitio web, ordenar de manera jerárquica las páginas con los contenidos (por niveles o categorías) y establecer los vínculos entre ellas permitiendo una navegación hipertextual.

A continuación se muestra el mapa de navegación correspondiente al módulo.

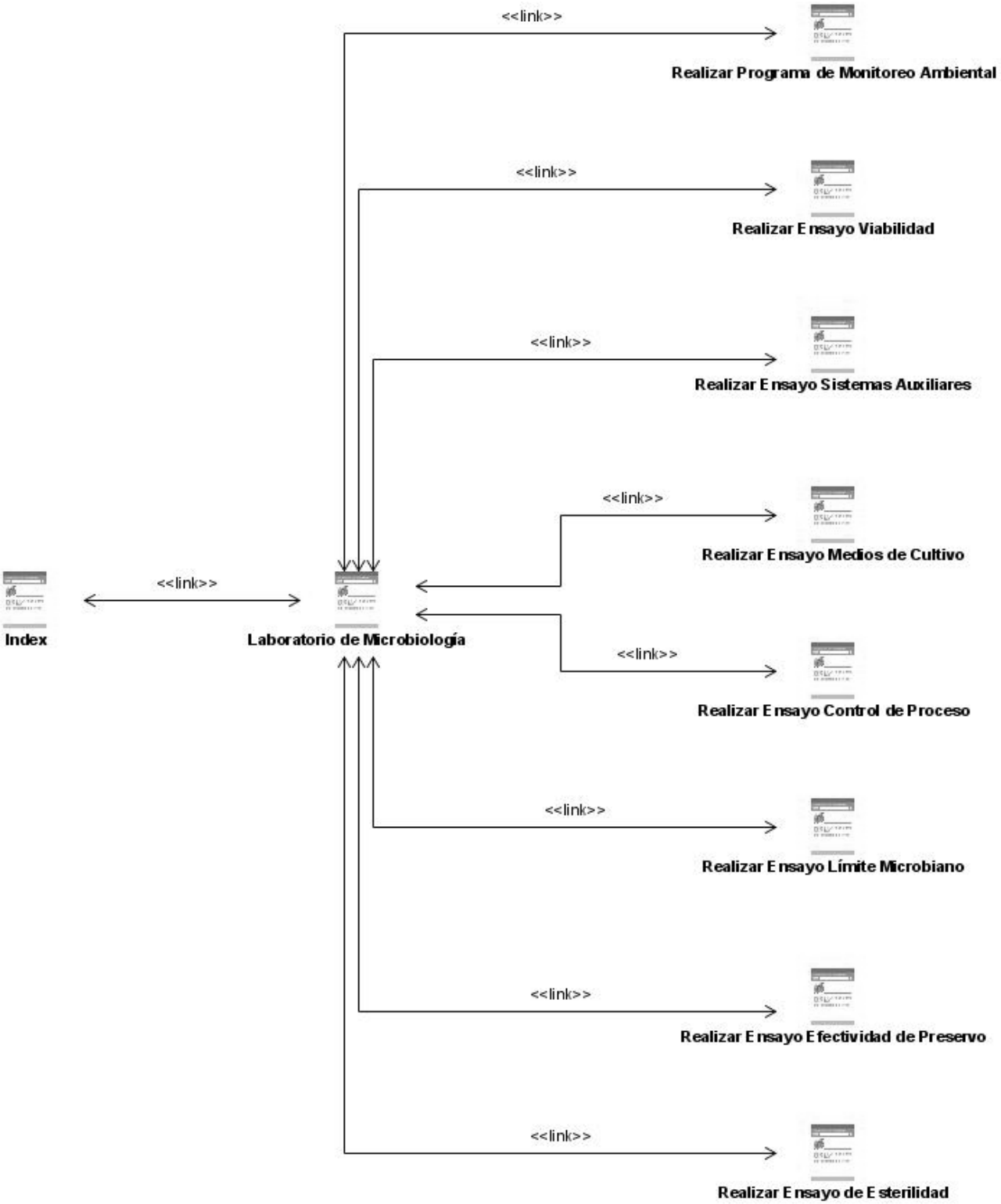

Fig 8: Mapa de Navegación (Parte 1).

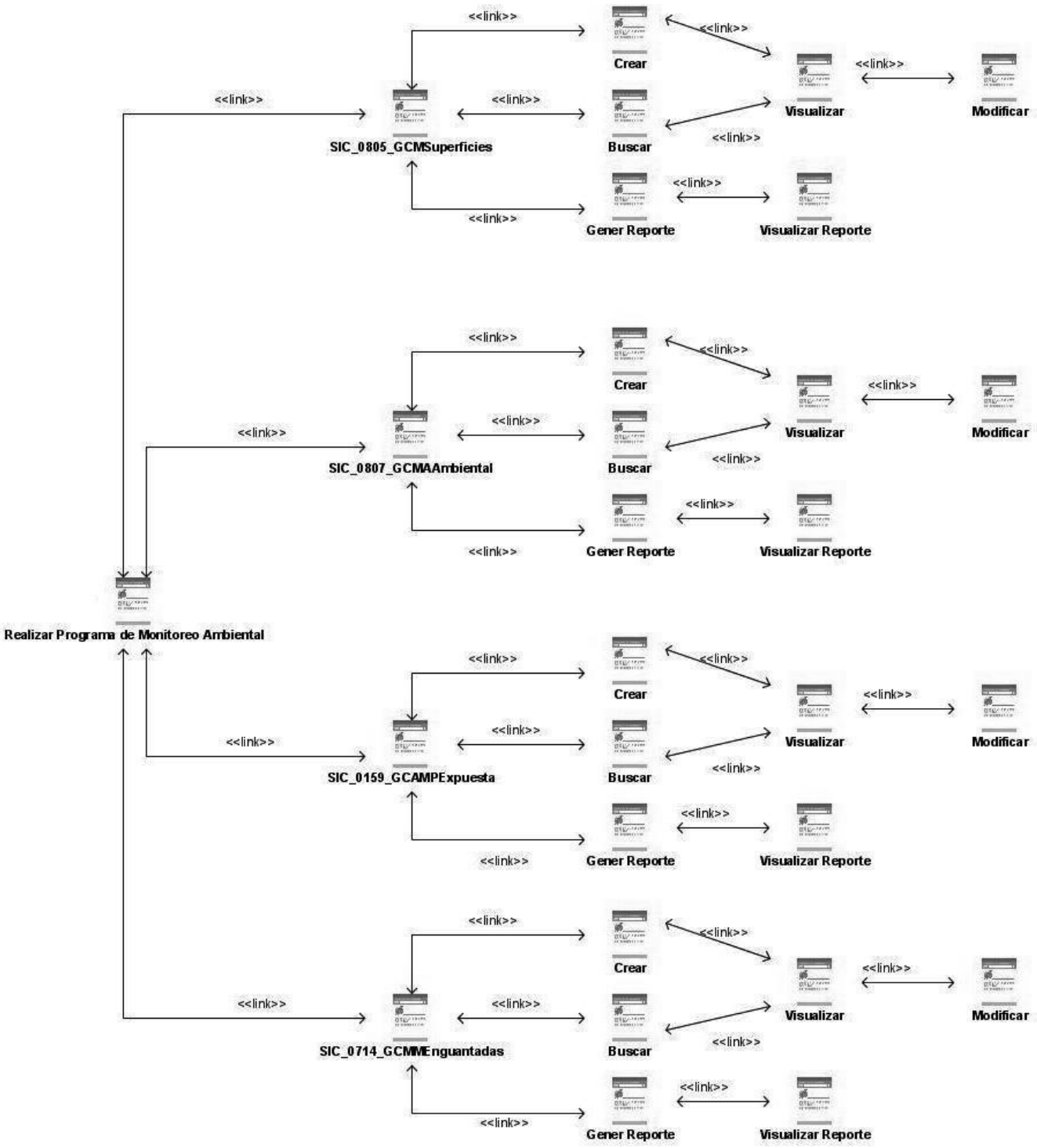

Fig 9: Mapa de Navegación (Parte 2).

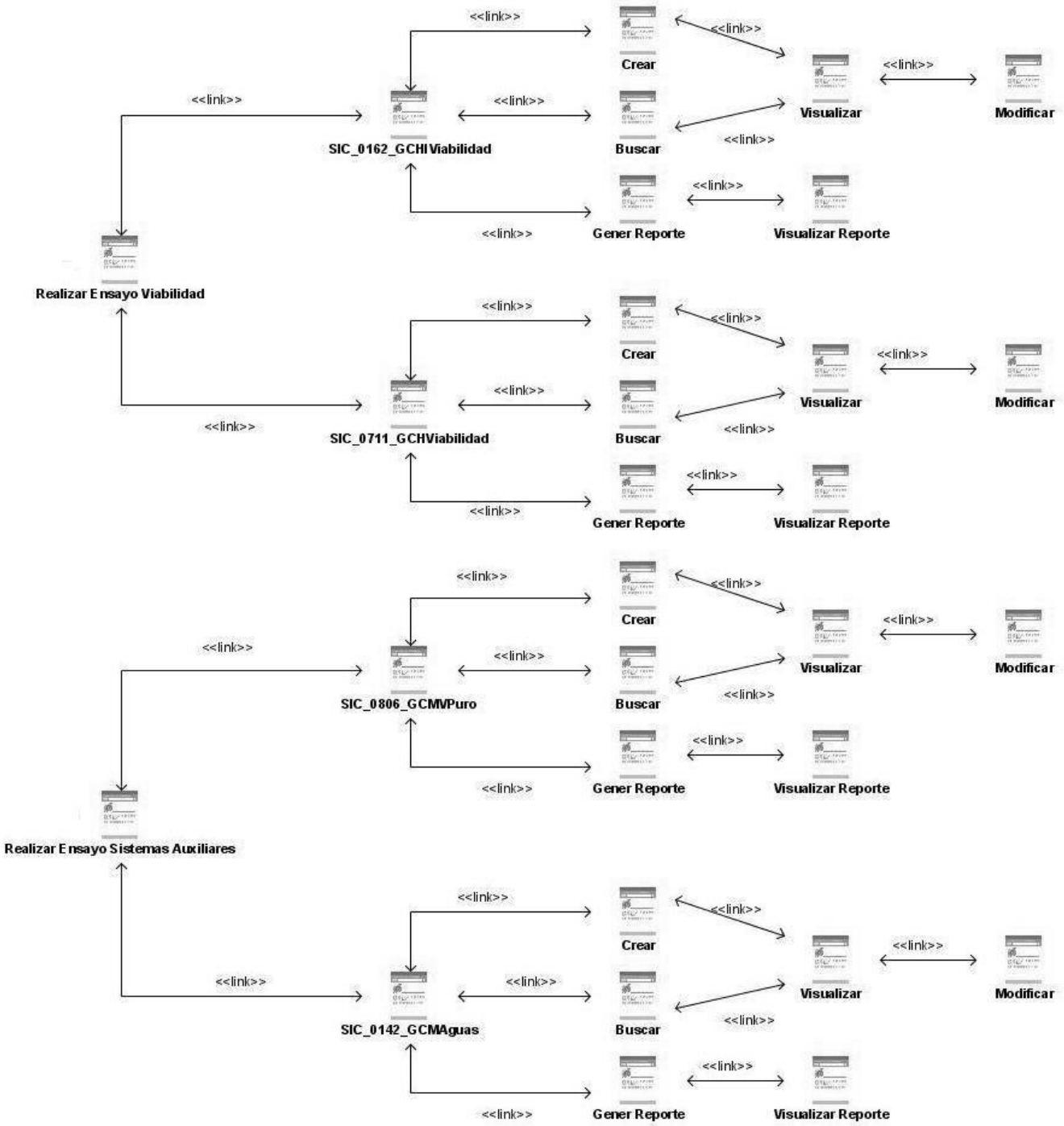

Fig 10: Mapa de Navegación (Parte 3).

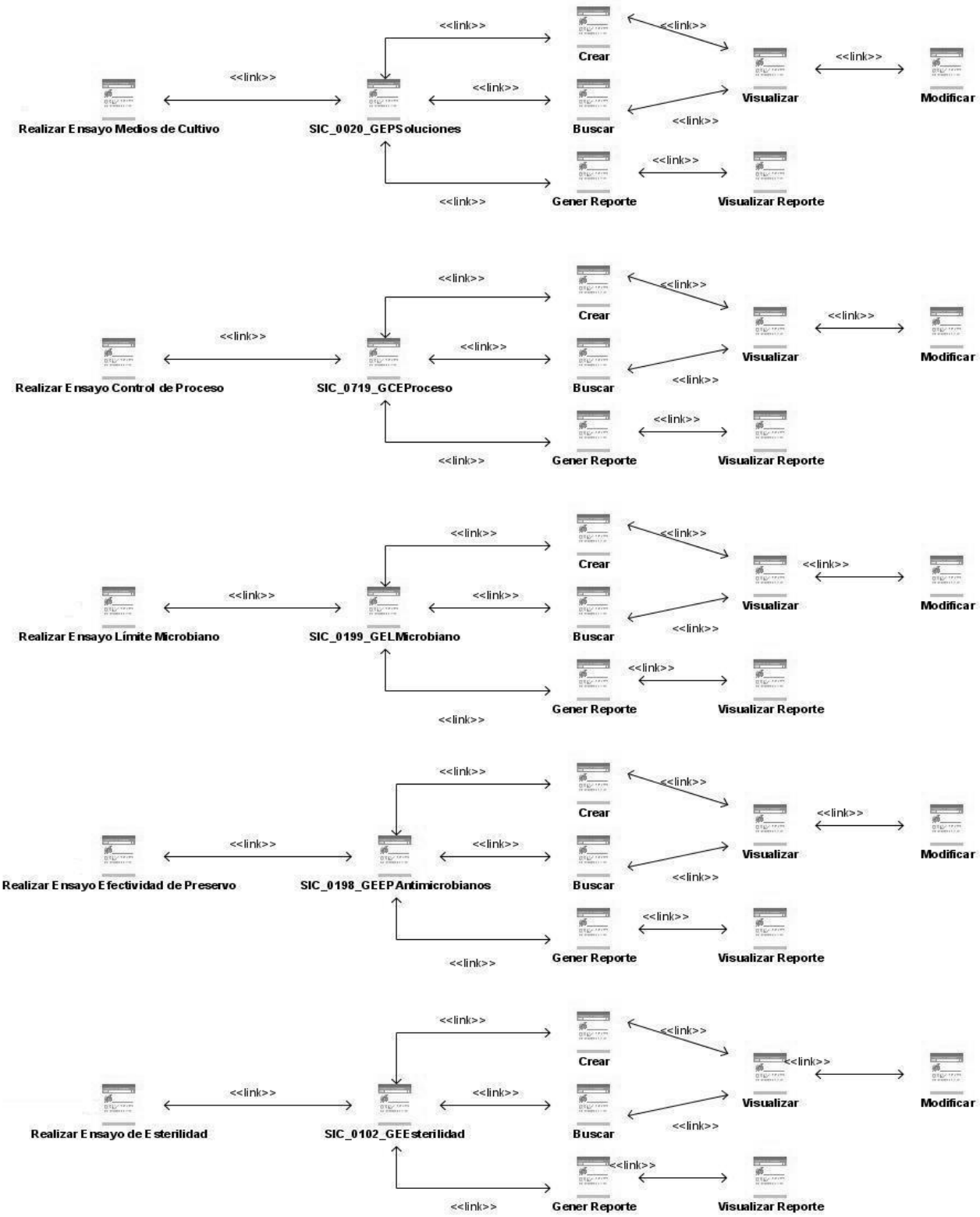

Fig 11: Mapa de Navegación (Parte 4).

2.4 Modelo de Diseño

El Modelo de Diseño es uno de los artefactos fundamentales en este flujo de trabajo ya que es una abstracción de la implementación del sistema, como resultado de un análisis realizado a los requerimientos del mismo. El diseño no debe ser ambiguo para obtener un modelo final suficiente para la implementación. Este modelo consiste en colaboraciones de clases que pueden ser agrupadas en subsistemas y paquetes además de describir la realización de los casos de uso. El objetivo principal de este trabajo es realizar el diseño a los 13 casos de uso propuesto en el levantamiento de requisitos propuesto por el equipo de analista. A continuación se muestra el resultado del mismo.

2.4.1 Diagramas de Clases del Diseño

Los diagramas de clases del diseño deben expresar exactamente cómo debe ser construido el sistema, pues se emplea una nomenclatura de los atributos y los métodos basada en el lenguaje que se utilizará en la implementación, además se muestran las clases y elementos necesarios así como las relaciones entre estas. Se utilizan para modelar la vista de diseño estática de un sistema.

Con el uso del framework Symfony se introduce el patrón arquitectónico Modelo-Vista-Controlador (MVC), por lo que los diagramas de clases del diseño correspondiente reflejan cómo se realizan las interacciones entre los principales elementos y clases del sistema, cumpliendo con el patrón arquitectónico definido. Para ello se agruparon las clases en paquetes, a continuación se muestran los mismos y su composición en cada uno de los diagramas de clases:

Vista:

- ✓ Páginas clientes
- ✓ Formularios
- ✓ Páginas servidoras
- ✓ Layout

Controlador:

- ✓ Controlador frontal: es el único punto de entrada a la aplicación: Carga la configuración y determina la acción a ejecutarse. Todas las peticiones web son manejadas por este controlador frontal.

- ✓ Acciones: Verifican la integridad de las peticiones y preparan los datos requeridos por la capa de presentación. Estas son el corazón de la aplicación, puesto que contienen toda la lógica de la misma. Además utilizan el modelo y definen variables para la vista. Como por ejemplo cuando se realiza una petición web en una aplicación Symfony, la URL define una acción y los parámetros de la petición.

Modelo: Contiene 4 clases representativas: Objeto, ObjetoPeer, BaseObjeto y BaseObjetoPeer. Las clases entidades y las generadas por Symfony (Base, Peer y BasePeer) que están implicadas en el funcionamiento de cada diagrama de diseño se encuentran en este paquete que es general para todos los diagramas, en el que van a estar todas las clases persistentes, con sus respectivas relaciones y las bases, las peer y las bases peer correspondientes a cada una de ellas. En cada diagrama de diseño una nota especifica cuáles clases del modelo se van a usar.

- ✓ Clases Base: Son las que se generan directamente a partir del esquema, no se deben modificar, porque cada vez que se genera el modelo, estas se borran.
- ✓ Clases Objeto: Heredan de las clases con nombre Base, estas no se modifican cuando se genera el modelo, por lo que en las mismas se añaden los métodos propios.
- ✓ Clases Peer: tienen métodos estáticos para trabajar con las tablas de la base de datos, proporcionan los medios necesarios para obtener los registros de las tablas y sus métodos devuelven normalmente uno o varios objetos de la clase relacionada.

A continuación se muestra un fragmento del Paquete Modelo, mostrando las relaciones entre las clases correspondientes al caso uso Gestionar Control Microbiológico de Vapor Puro (SIC-0806).

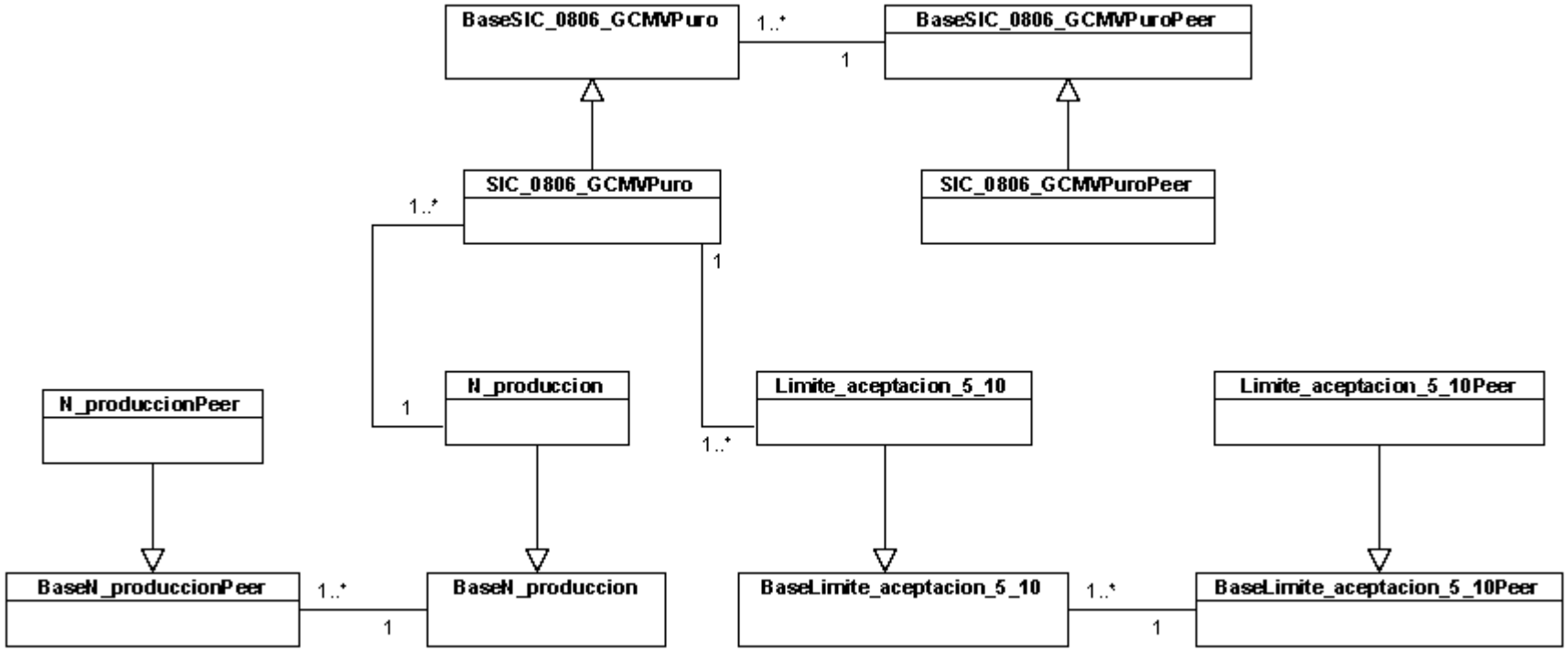

Fig 12: Fragmento del Paquete Modelo

Caso de Uso: Gestionar Ensayo Control de Proceso.

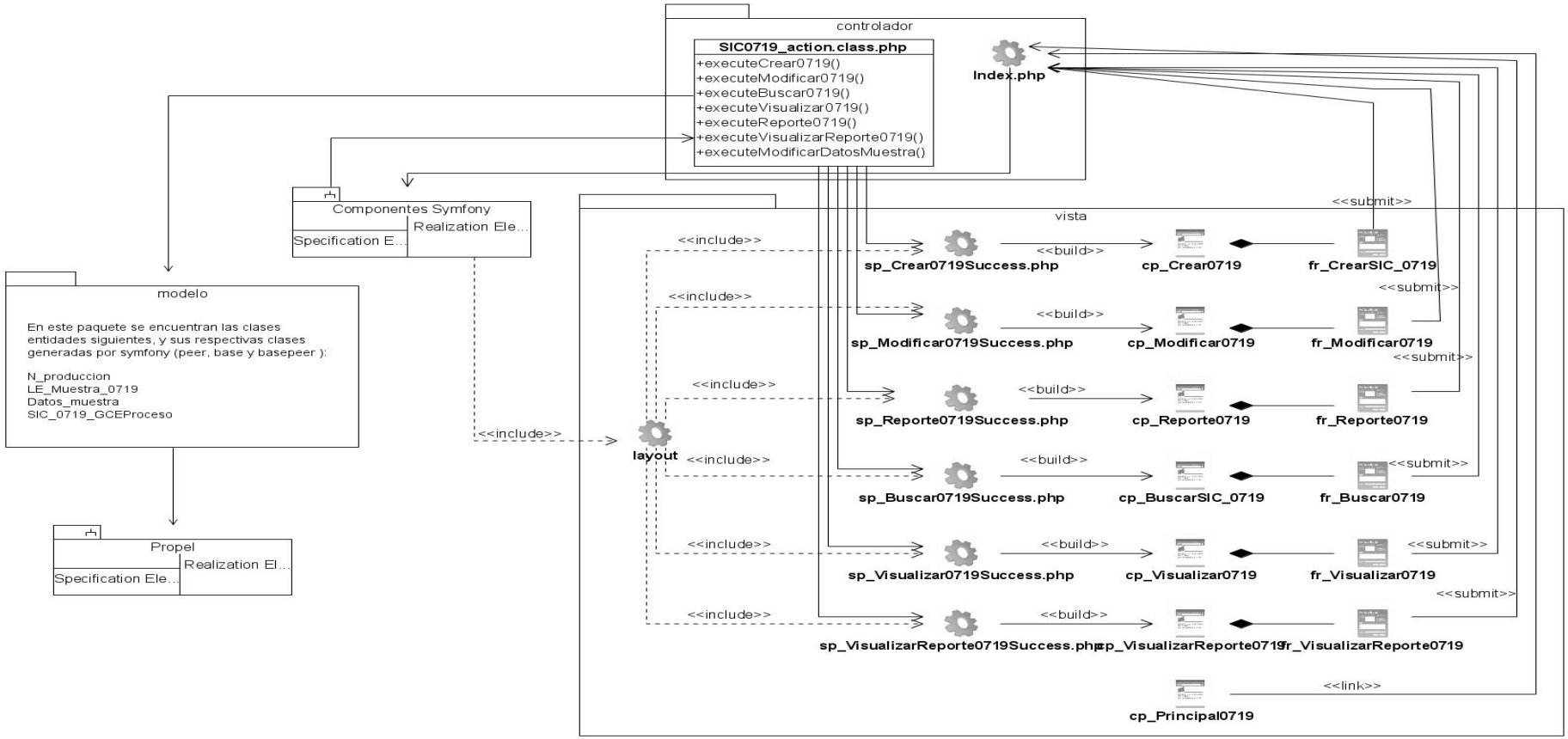

Fig 13: Diagrama de clases del diseño del caso de uso Gestionar Ensayo Control de Proceso.

Caso de Uso: Gestionar Chequeo de Viabilidad.

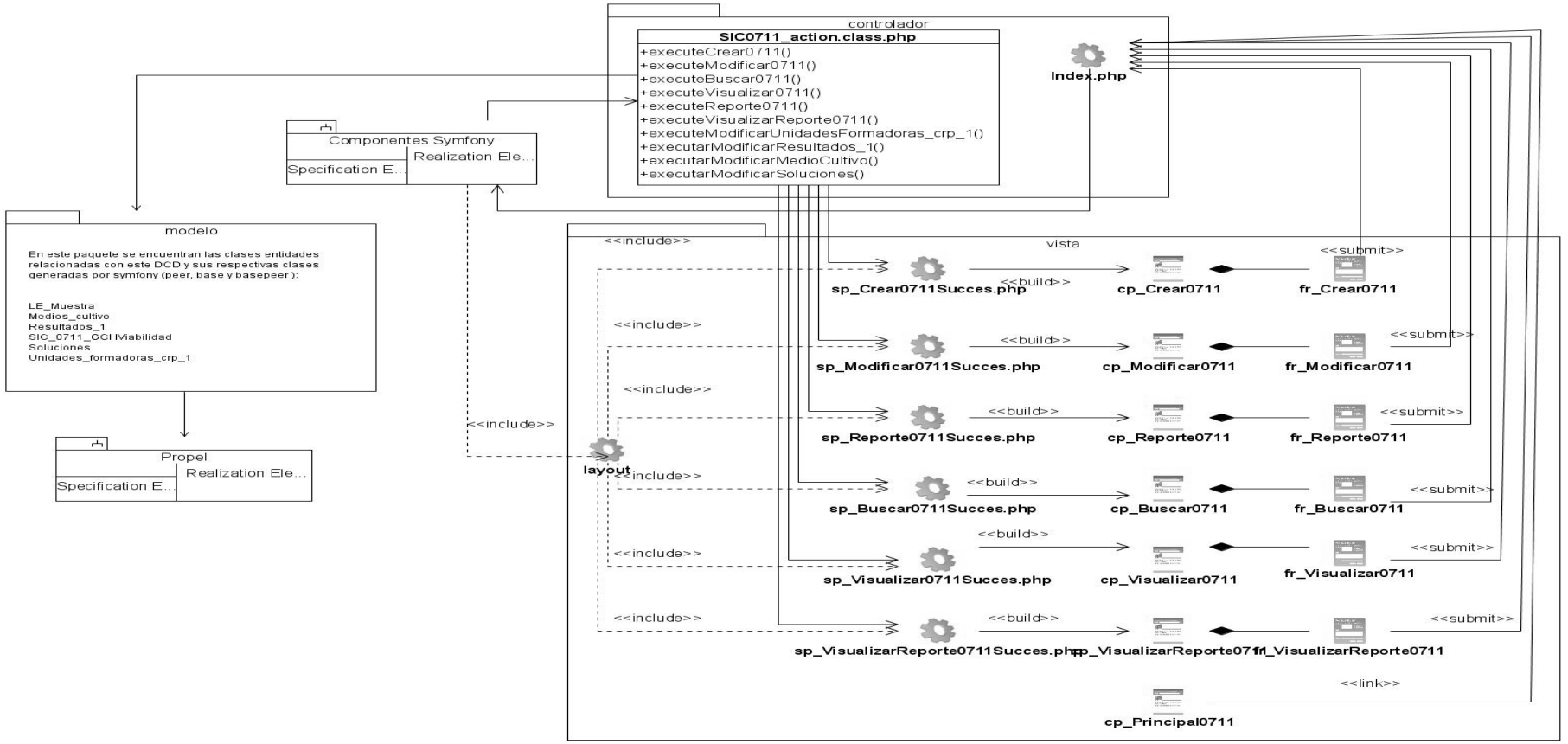

Fig 14: Diagrama de clases del diseño del caso de uso Gestionar Chequeo de Viabilidad.

Caso de Uso: Gestionar Control Microbiológico de las Aguas.

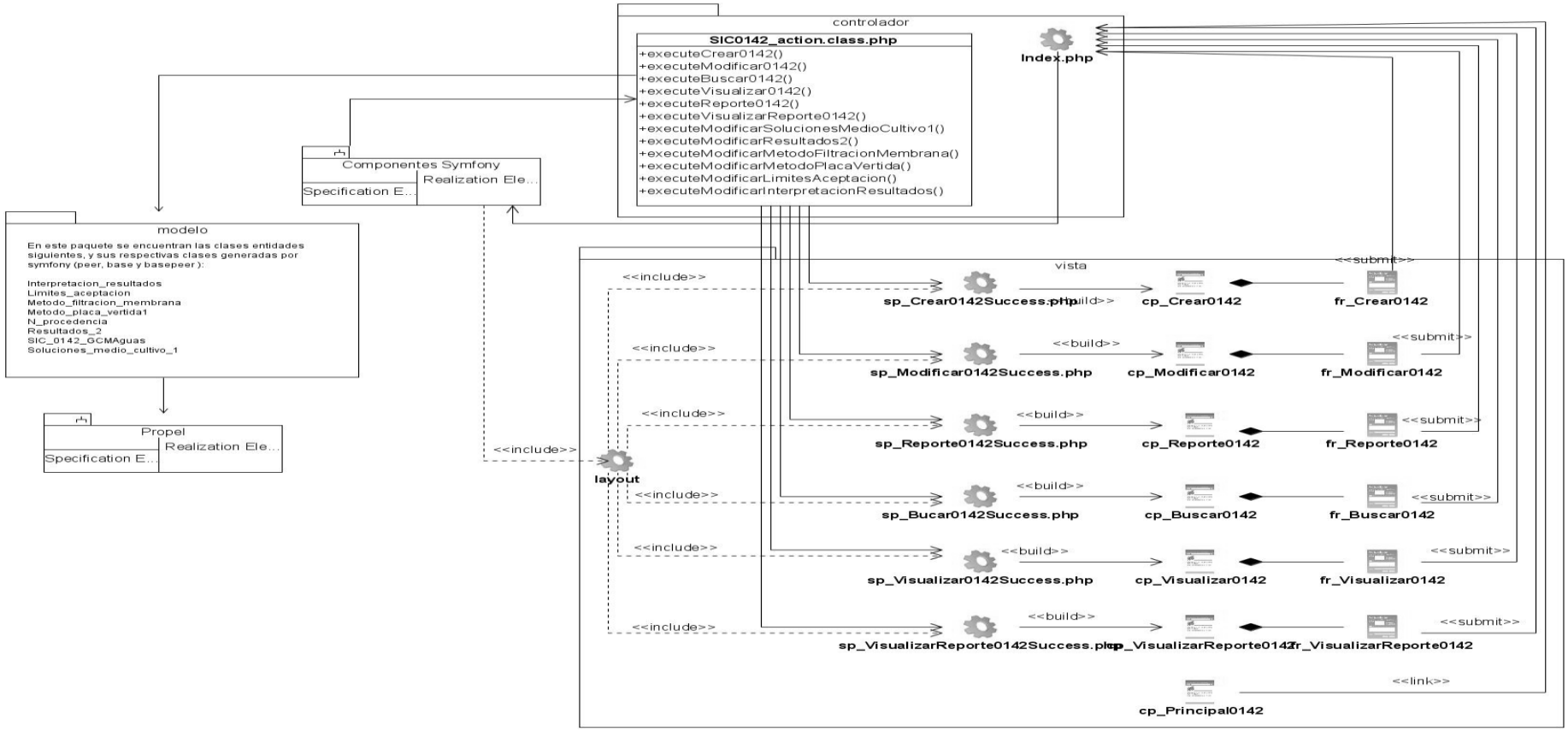

Fig 15: Diagrama de clases del diseño del caso de uso Gestionar Control Microbiológico de las Aguas.

Caso de Uso: Gestionar Control Ambiental mediante Placa Expuesta.

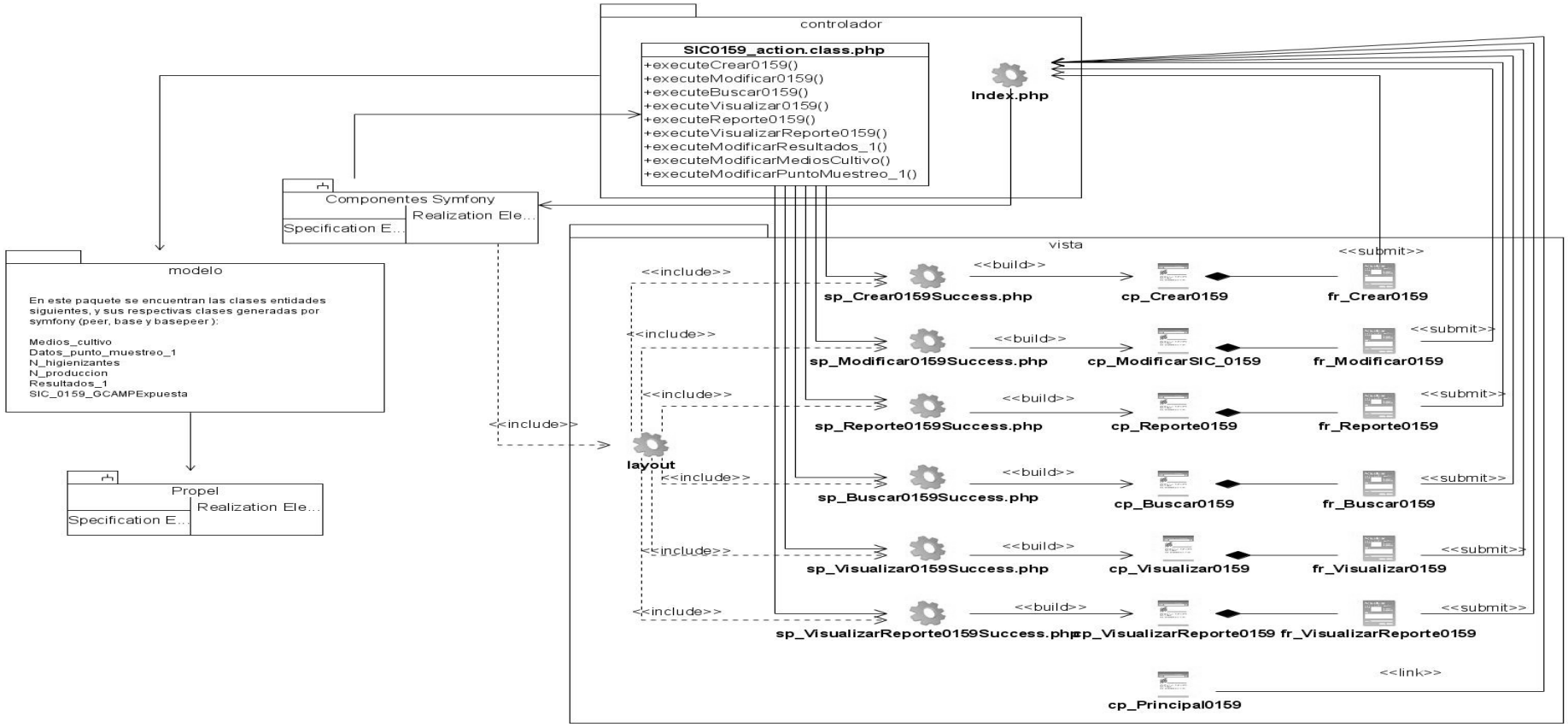

Fig 16: Diagrama de clases del diseño del caso de uso Gestionar Control Ambiental mediante Placa Expuesta.

Caso de Uso: Gestionar Chequeo Inicial de Viabilidad.

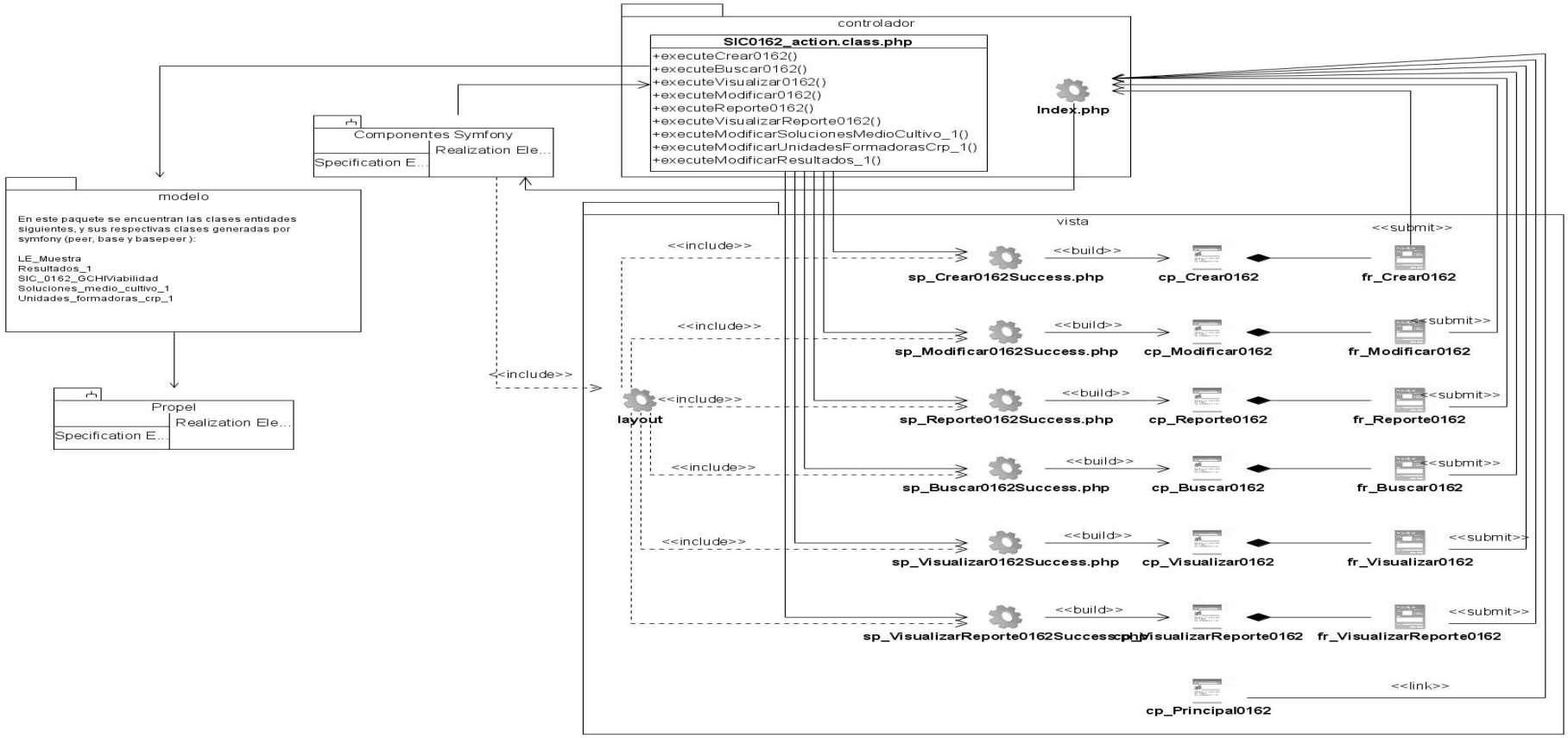

Fig 17: Diagrama de clases del diseño del caso de uso Gestionar Chequeo Inicial de Viabilidad.

Caso de Uso: Gestionar Ensayo Efectividad de Preservos Antimicrobianos.

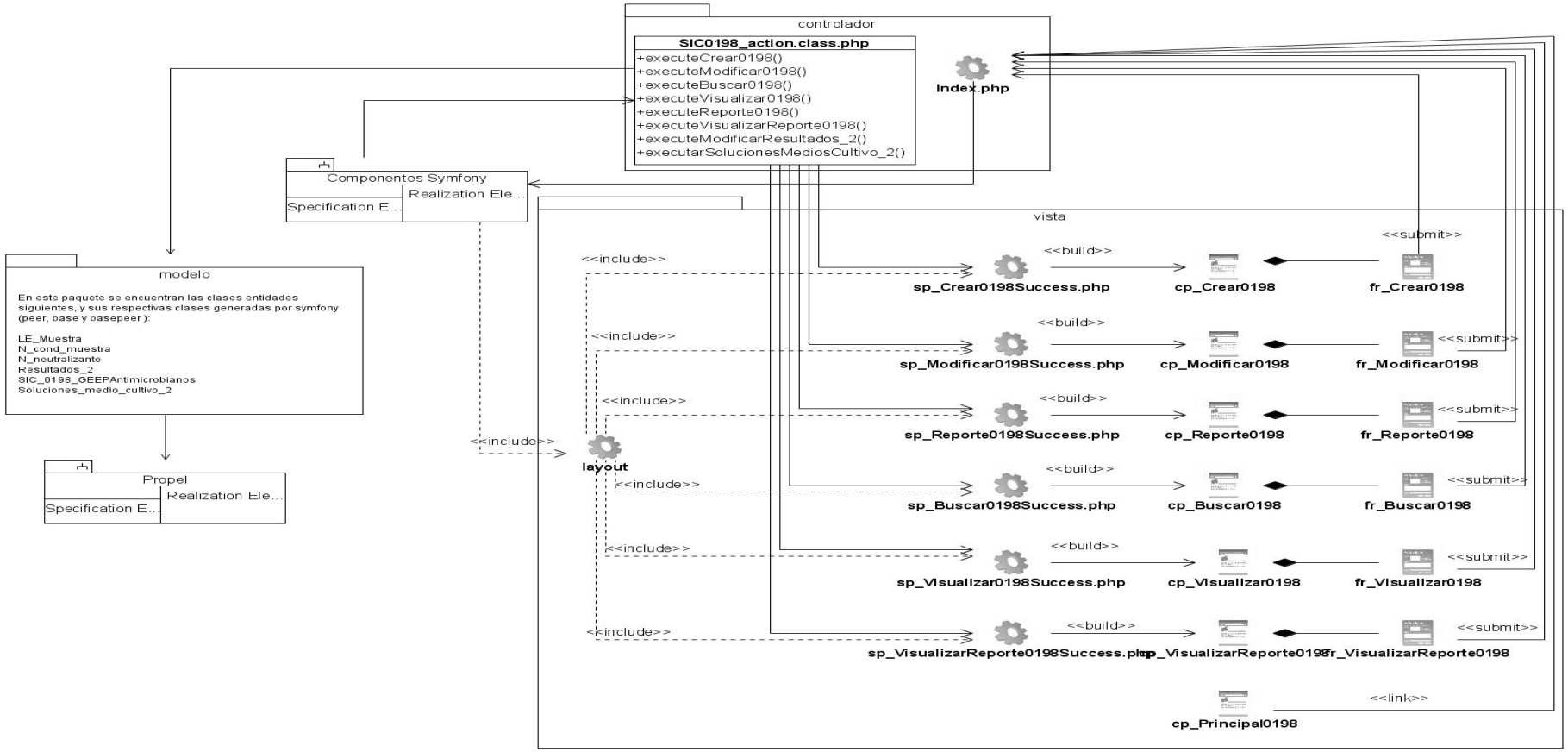

Fig 18: Diagrama de clases del diseño del caso de uso Gestionar Ensayo Efectividad de Preservos Antimicrobianos.

Caso de Uso: Gestionar Ensayo Límite Microbiano.

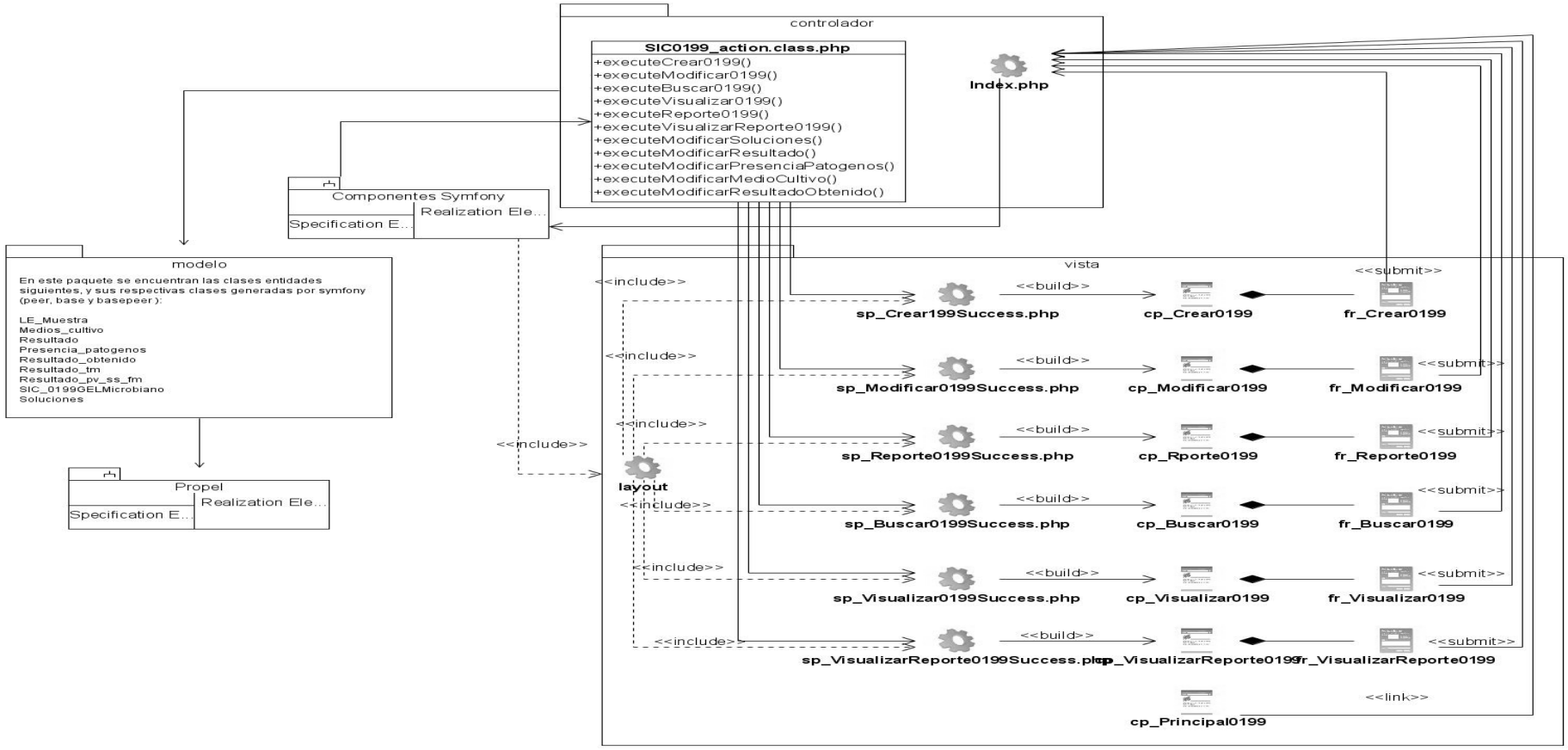

Fig 19: Diagrama de clases del diseño del caso de uso Gestionar Ensayo Límite Microbiano.

Caso de Uso: Gestionar Ensayo de Esterilidad.

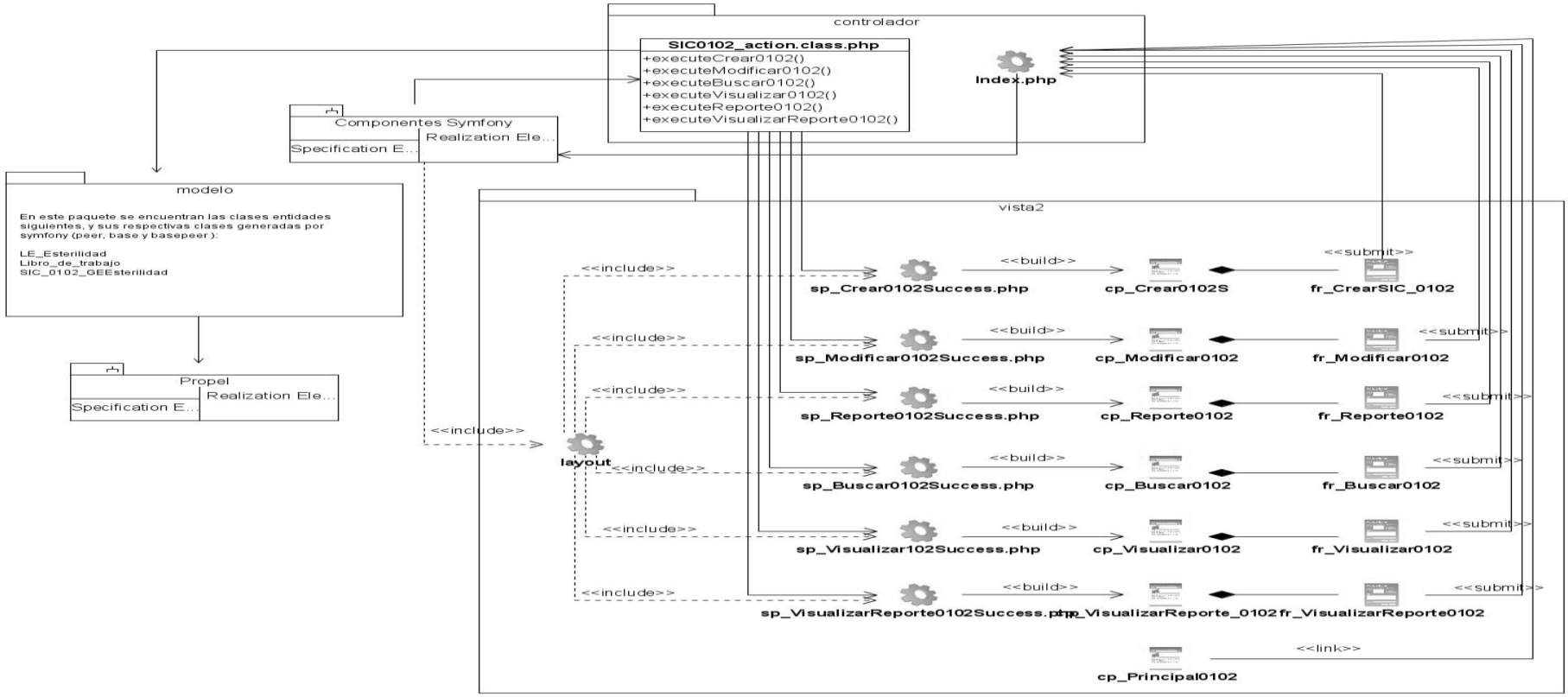

Fig 20: Diagrama de clases del diseño del caso de uso Gestionar Ensayo de Esterilidad.

Caso de Uso: Gestionar Control Microbiológico de las Manos Enguantadas.

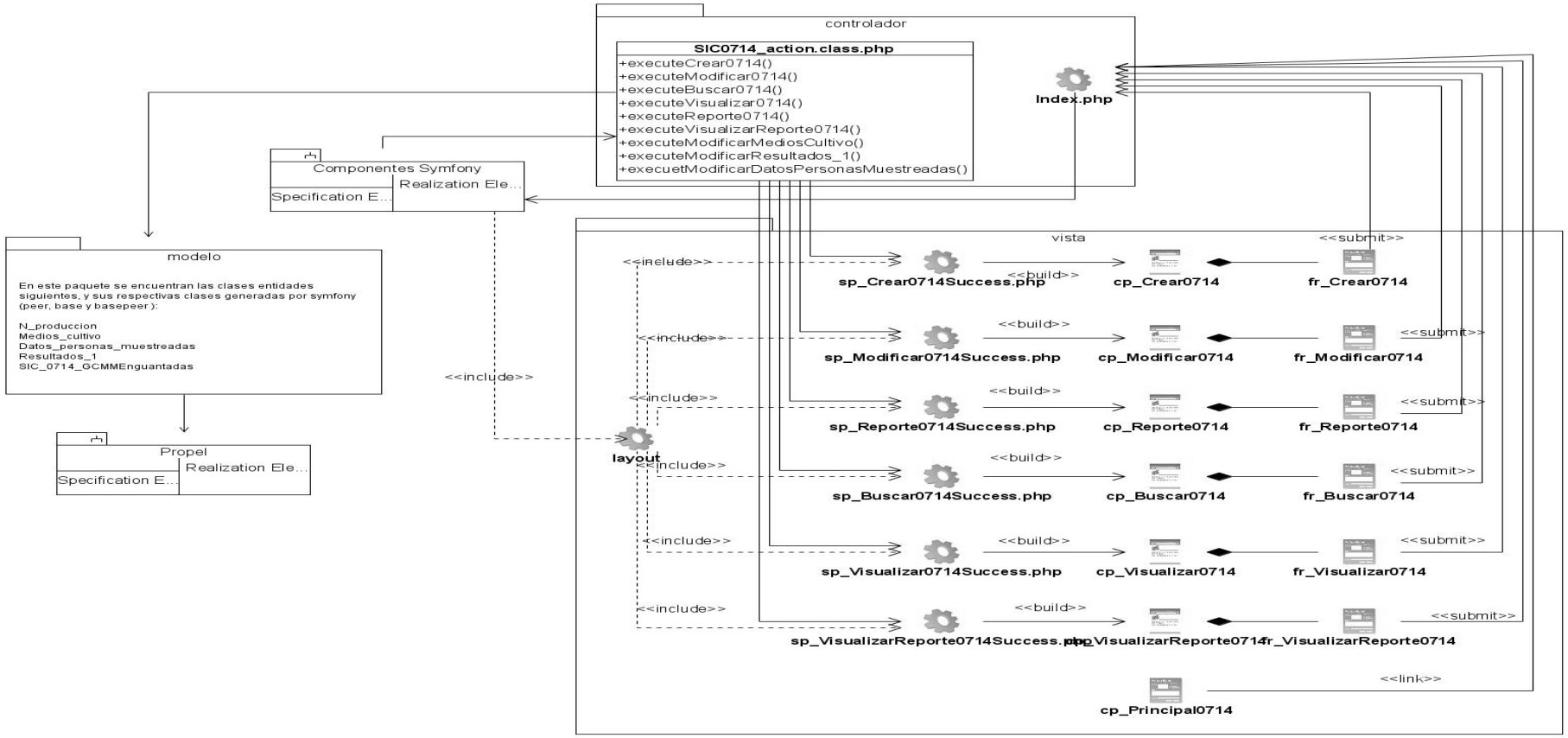

Fig 21: Diagrama de clases del diseño del caso de uso Gestionar Control Microbiológico de las Manos Enguantadas.

Caso de Uso: Gestionar Ensayo de Preparación de Soluciones.

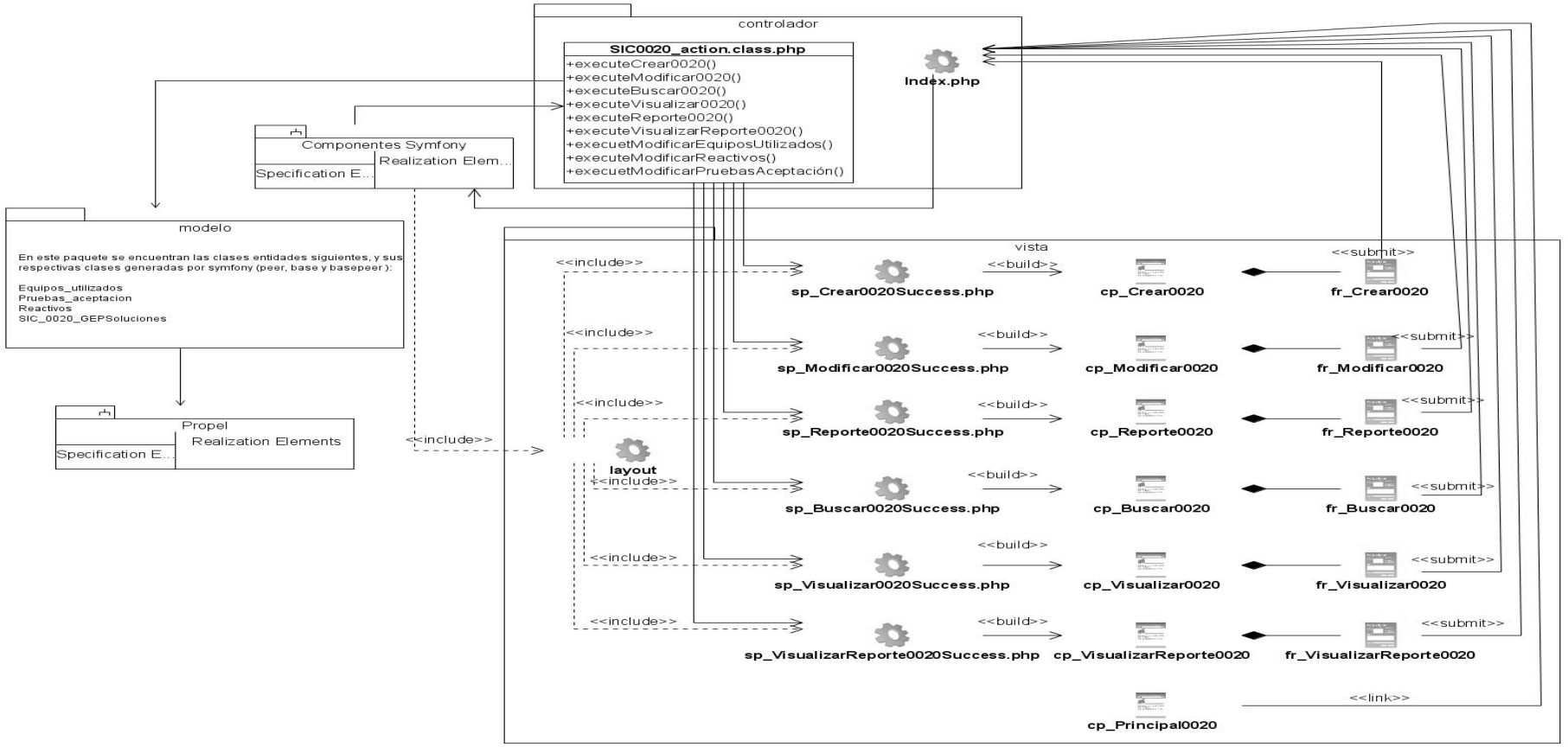

Fig 22: Diagrama de clases del diseño del caso de uso Gestionar Ensayo de Preparación de Soluciones.

Caso de Uso: Gestionar Control Muestreo de Superficies.

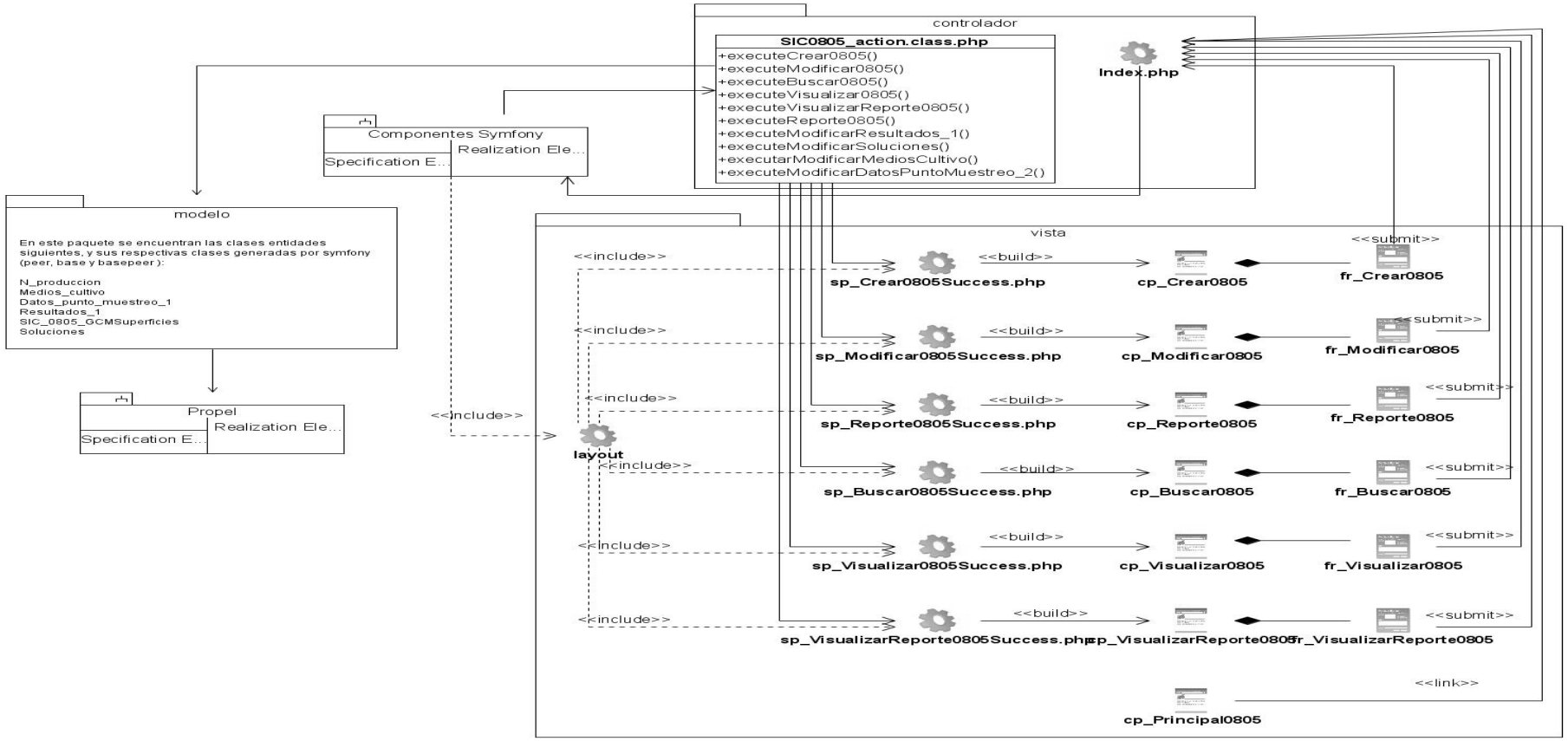

Fig 23: Diagrama de clases del diseño del caso de uso Gestionar Control Muestreo de Superficies.

Caso de Uso: Gestionar Control Microbiológico de Vapor Puro.

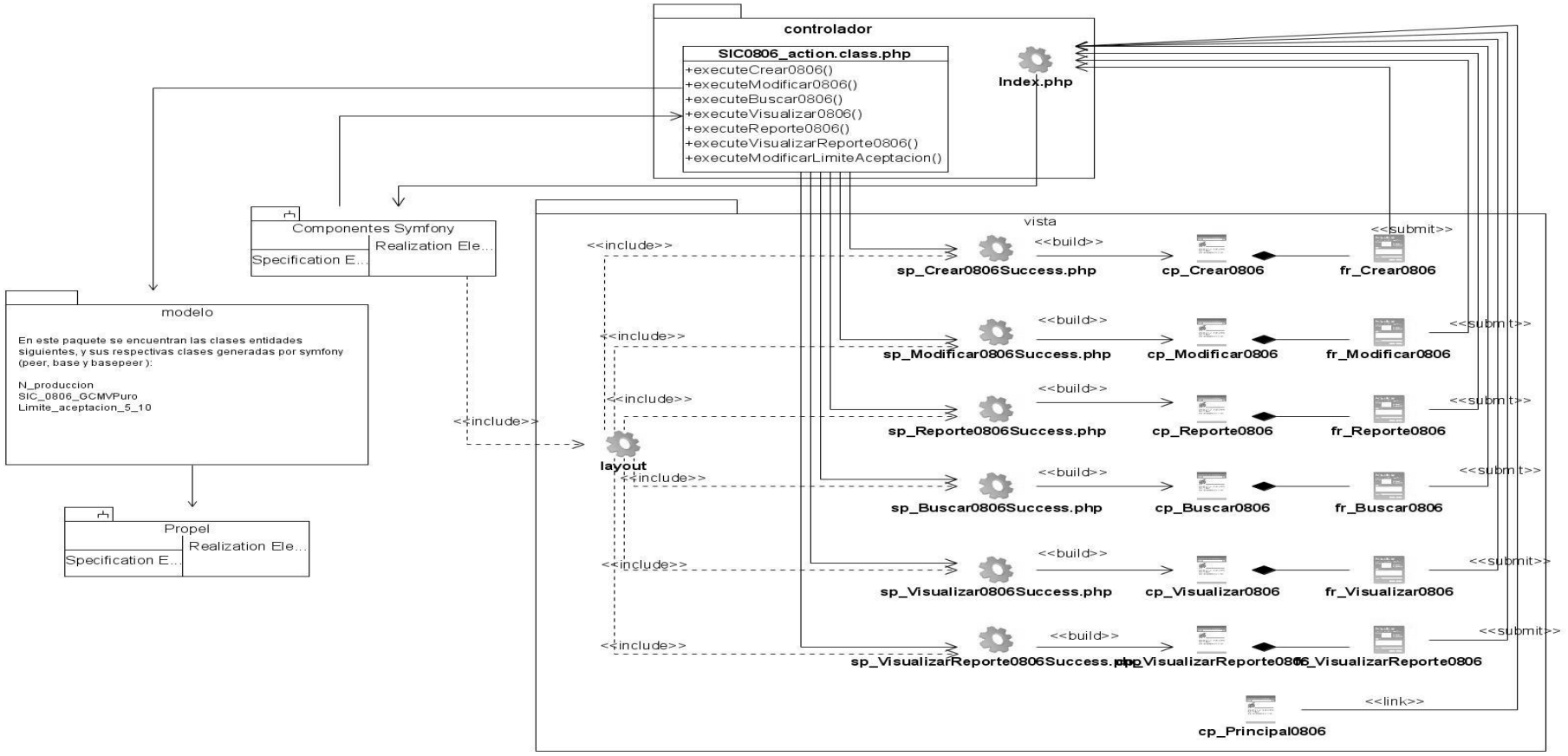

Fig 24: Diagrama de clases del diseño del caso de uso Gestionar Control Microbiológico de Vapor Puro.

Caso de Uso: Gestionar Control Muestreo con el Analizador Ambiental.

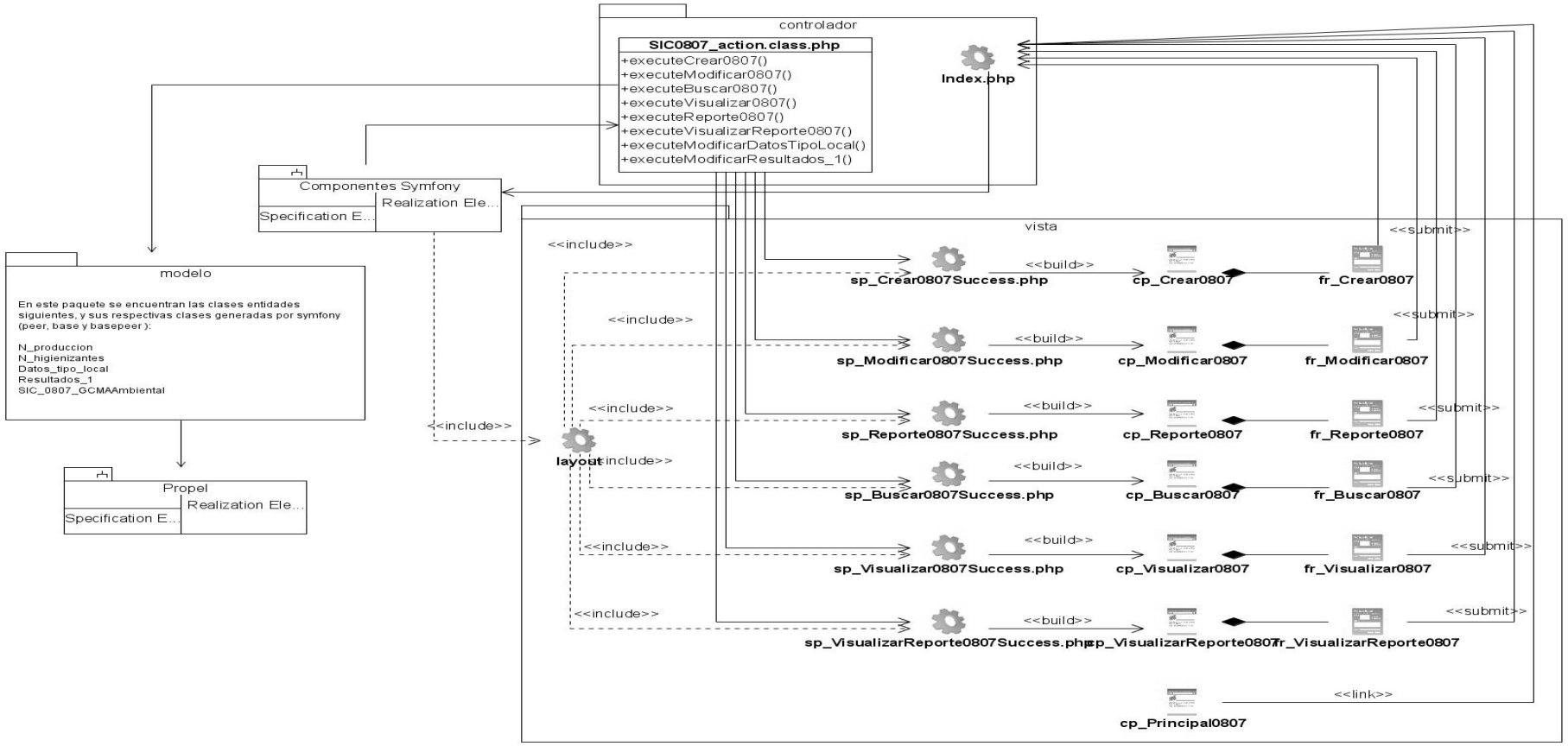

Fig 25: Diagrama de clases del diseño del caso de uso Gestionar Control Muestreo con el Analizador Ambiental.

2.4.2 Diagramas de Secuencia del Diseño

En los diagramas de secuencia para aplicaciones Web es necesario considerar que hay mensajes que son usados simbólicamente para reflejar interacciones propias de la tecnología, tal es el caso de los vínculos, envíos de formulario, construcción de páginas, navegaciones y redireccionamientos (link, submit, build navigate, redirect). Como consecuencia de la interacción entre el actor y la Aplicación Web, muchas de esas interacciones se ejecutan entre los distintos elementos que la componen, su representación en un diagrama de secuencia contribuye a entender mejor su funcionamiento, colaborando con el objetivo de documentar la solución.

En este epígrafe se pondrán los diagramas de secuencia de los casos de uso del sistema, los cuales ilustran el funcionamiento que tendrá el sistema ante alguna petición del actor.

Para lograr hacer una modelación de forma efectiva y que el resultado de la misma sea comprensible, son muy útiles los Mecanismos de Diseño, artefacto propuesto y descrito en RUP 2003, entre sus beneficios están:

- ✓ Mantener la homogeneidad en el diseño.
- ✓ Reutilizar soluciones anteriormente probadas.
- ✓ Reutilizar documentación.

De estos tres aspectos, la reutilización de la documentación es la que más se vincula con este trabajo. A continuación se explica brevemente el por qué de esta afirmación:

Luego de cada petición del usuario ésta es decodificada por sfController la cual es la clase del controlador y la transfiere a la acción correspondiente. Pero primeramente antes de ejecutar cualquier acción la clase sfController verifica la seguridad y valida los formularios, este paso siempre se usa una y otra vez en cualquier petición que hace el usuario. A continuación en la *(Figura 26)* mostramos un pequeño fragmento donde queda evidenciada el uso de la clase sfController.

En cada uno de los diagramas de secuencia vamos omitir los pasos donde se usa la clase sfController. Iremos directamente desde la petición del usuario hasta la acción correspondiente. Se dejará solo el escenario Generar Reporte SIC-0806 del caso de uso Gestionar Control Microbiológico de Vapor Puro

(SIC-0806) como ejemplo donde quede representado el uso de la clase sfController, y así de esta forma se evidencia el uso de la reutilización de la documentación en este trabajo.

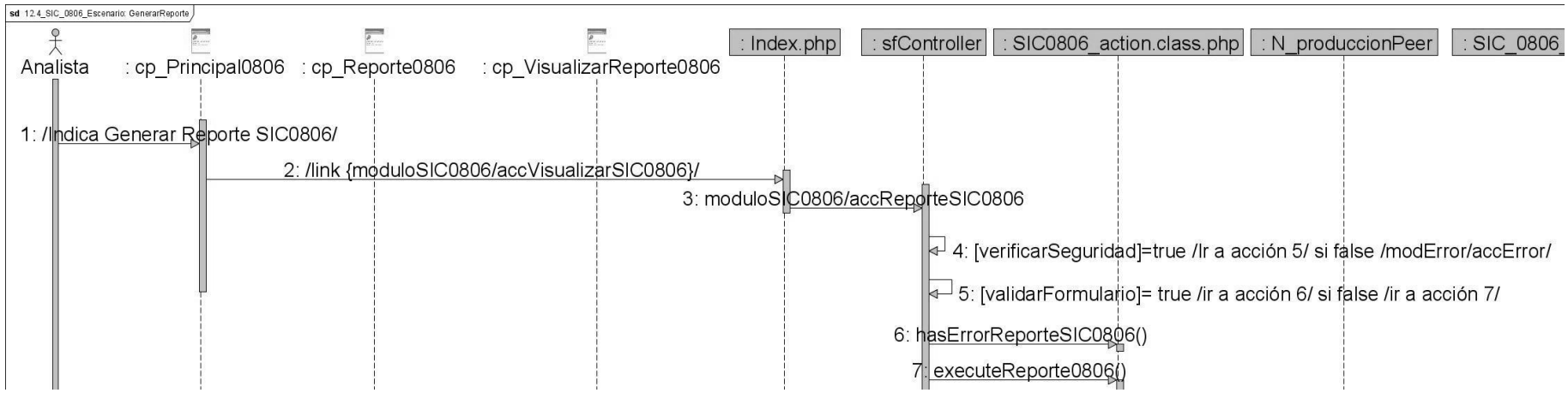

Fig 26: Fragmento del Dsq escenario Generar Reporte SIC-0806: Ejemplo del uso de la clase sfController.

Para una mejor interpretación de los diagramas de secuencia se explica a continuación pasos a realizar en los mismos. Se toma como ejemplo escenarios del caso de uso Gestionar Control Microbiológico de Vapor Puro (SIC-0806), ya que los otros diagramas de los restantes casos de uso son muy similares a estos.

El método Actualiza_Campos_de_Limites_de_Aceptacion() se encuentra representado en los escenarios Crear y Modificar, es utilizado como un método representativo para indicar al programador lo que debe hacer. Es utilizado cuando se indica agregar datos en los campos de Límites de Aceptación. Una vez que el usuario indique agregar datos de Límites de Aceptación debe de validarse cada campo, seguidamente cada dato entrado debe ir mostrándose uno debajo del otro y permitiendo que el analista pueda introducir nuevamente datos en esos campos. Estos pasos se ve reflejado en los escenarios Crear y Modificar de cada SIC. A continuación se muestra un pequeño fragmento del escenario Crear SIC-0806 donde podemos encontrar este método.

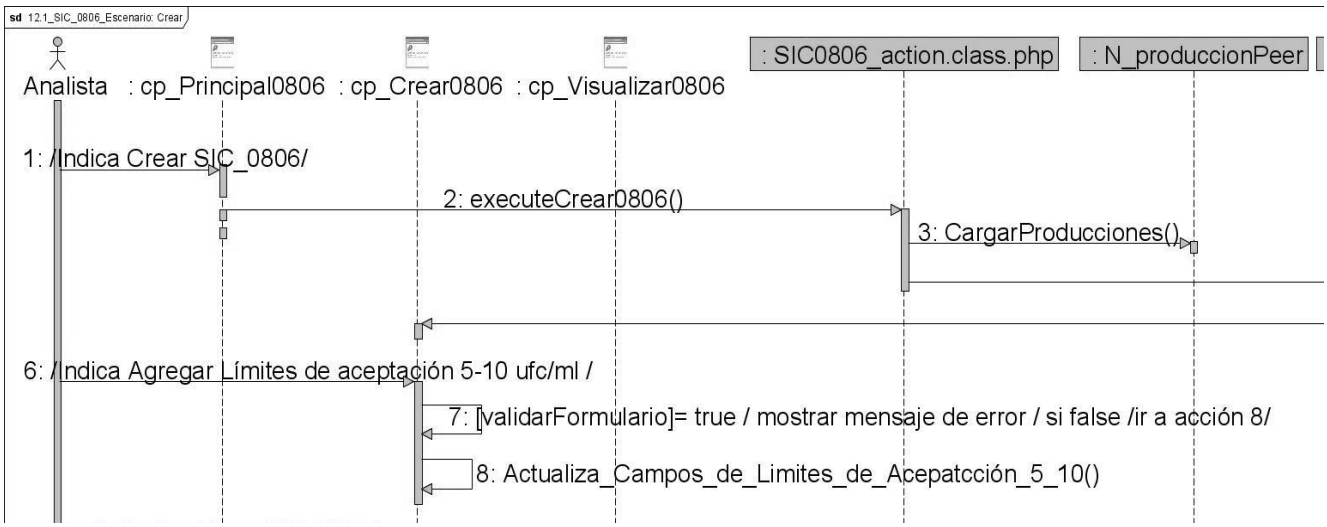

Fig 27: Fragmento del Dsq escenario Crear SIC-0806: Ejemplo de los agregar datos en los campos Límites de aceptación.

Cuando se indica crear el SIC-0806 se llama al método Crear_SIC_0806_GCMVPuro() de la clase SIC_0806_GCMVPuroPeer, y se le pasan todos sus atributos como parámetros, este método devuelve un

entero que es el que identifica a este SIC, un poco más adelante explicaremos que es lo que se hace con este identificador. En este método Crear_SIC_0806_GCMVPuro() se construye el objeto SIC_0806_GCMVPuro, y se le hace un *set* a cada uno de sus atributos pasándole como parámetros los nuevos datos introducidos. Luego de creado el objeto SIC_0806_GCMVPuro y devuelto su identificador, cada dato introducido en los campos de Límites de aceptación son guardados en su tabla correspondiente de la base de datos, a través del método Crear_Soluciones_medio_cultivo_1() que se encuentra en la clase Limites_aceptacionPeer.

Cada método Crear_Objeto(), es el encargado de construir al objeto de la peer donde se encuentra y recibe como parámetro cada atributo del objeto a construir, también recibe como parámetro el identificador del SIC creado, así de esta forma poder identificar a que SIC corresponde. Aquí me refiero solamente a los objetos que pertenezcan a un SIC.

Los pasos que se utilizan para representar como es que se crean los objetos a través del método Crear_Objeto() de la clase ObjetoPeer es el que se muestra en la siguiente figura:

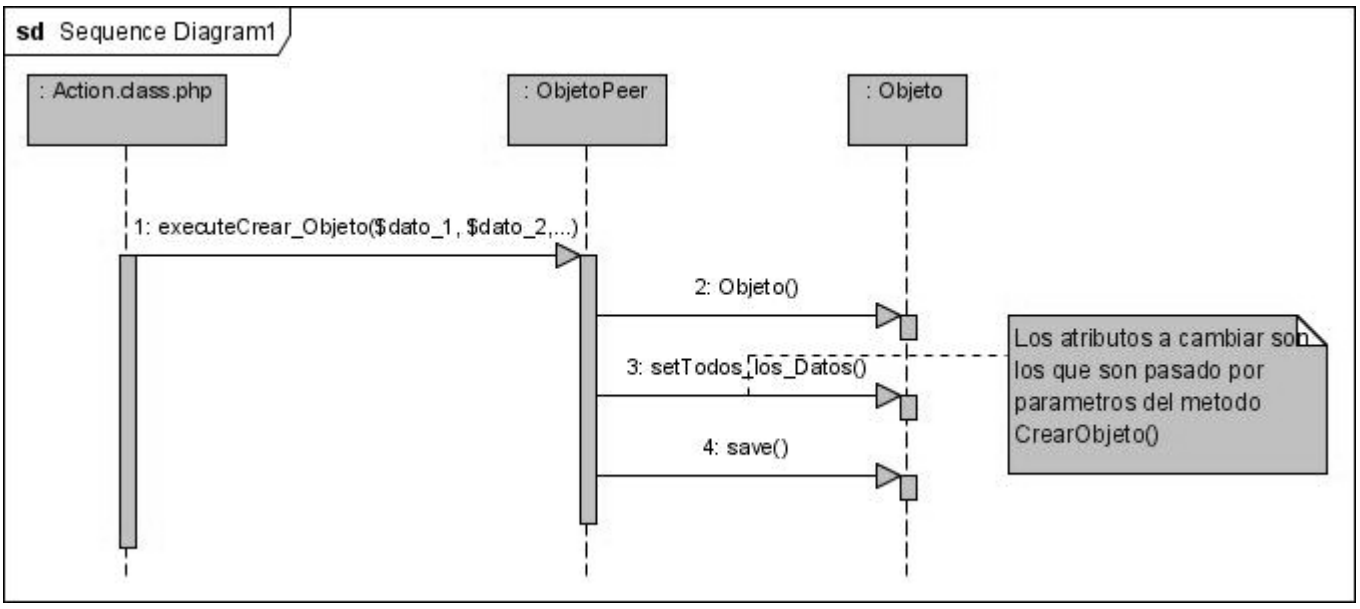

Fig 28: Pasos representado el Método Crear_Objeto()

Si se dan cuenta en el mensaje número tres se pone el mensaje *setTodos_los_Datos(),* este mensaje no es un método, solo se trata de representar que se le hace un *set* a cada atributo del objeto, ya que cada objeto que existen en este módulo tienen muchos atributos y si pusiera todos los set de cada uno de ellos los diagramas de secuencia se harían mucho más extensos, cuando pudieran representarse con un único mensaje: *setTodos_los_Dato().*

Los escenarios Buscar y Visualizar y Generar Reporte del caso de uso Gestionar Control Microbiológico de Vapor Puro (SIC-0806) se pueden entender perfectamente por parte de los programadores.

Caso de Uso: Gestionar Control Microbiológico de Vapor Puro

1. Escenario: Generar Reporte SIC-0806

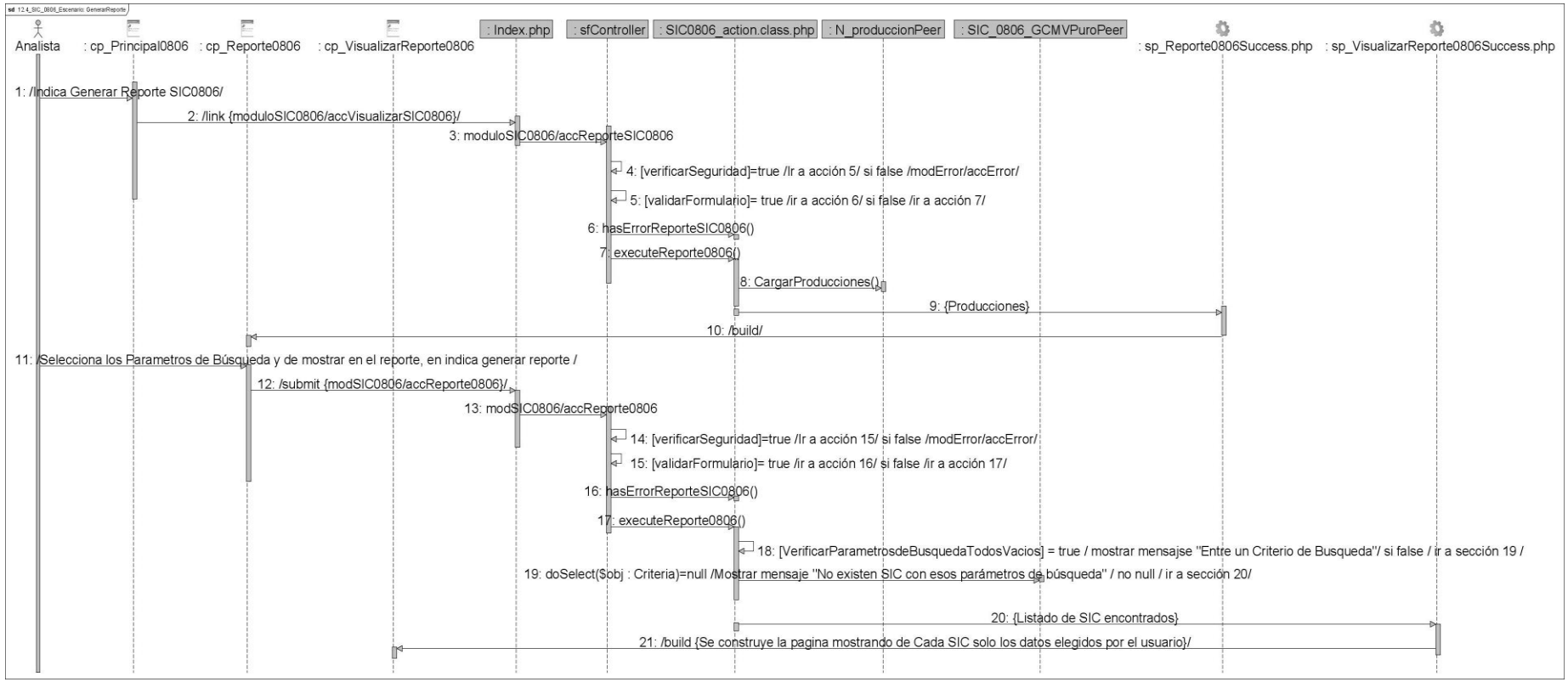

Fig 29: Dsq caso de uso Gestionar Control Microbiológico de Vapor Puro: Escenario Generar Reporte SIC-0806

2. Escenario: Crear SIC-0806

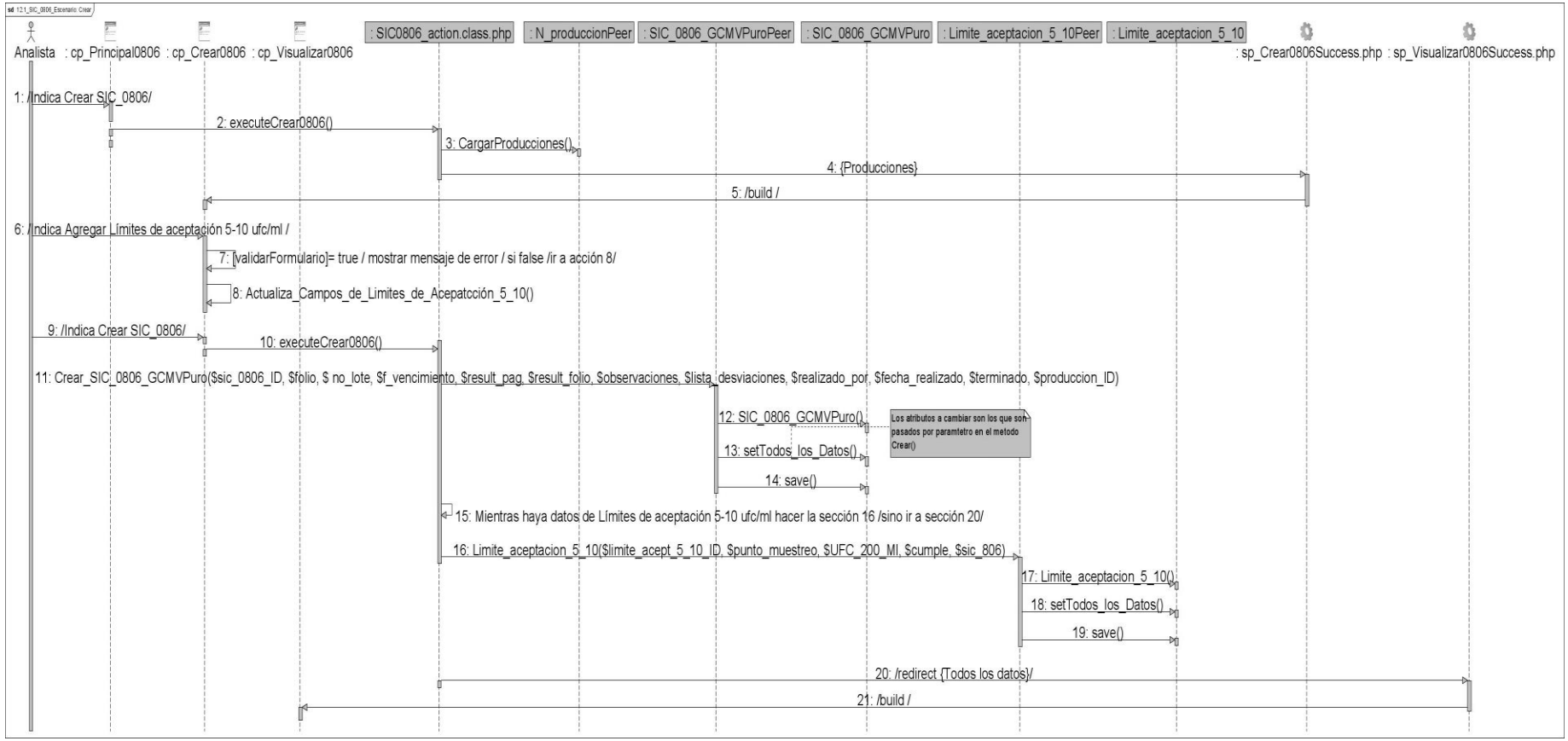

Fig 30: Dsq caso de uso Gestionar Control Microbiológico de Vapor Puro: Escenario Crear: SIC-0806

3. Escenario: Buscar y Visualizar SIC-0806

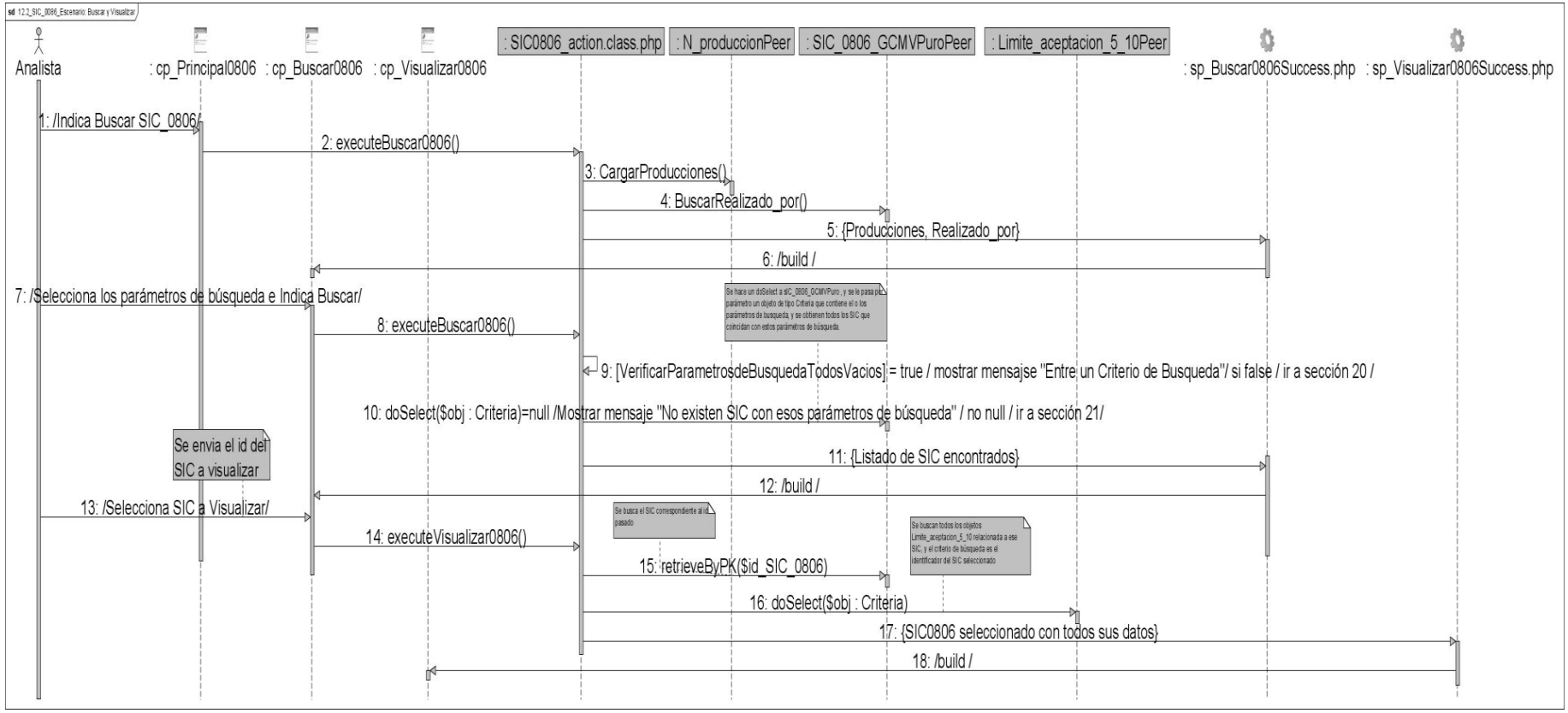

Fig 31: Dsq caso de uso Gestionar Control Microbiológico de Vapor Puro: Escenario Buscar y Visualizar SIC-0806

4. Escenario: Modificar SIC-0806

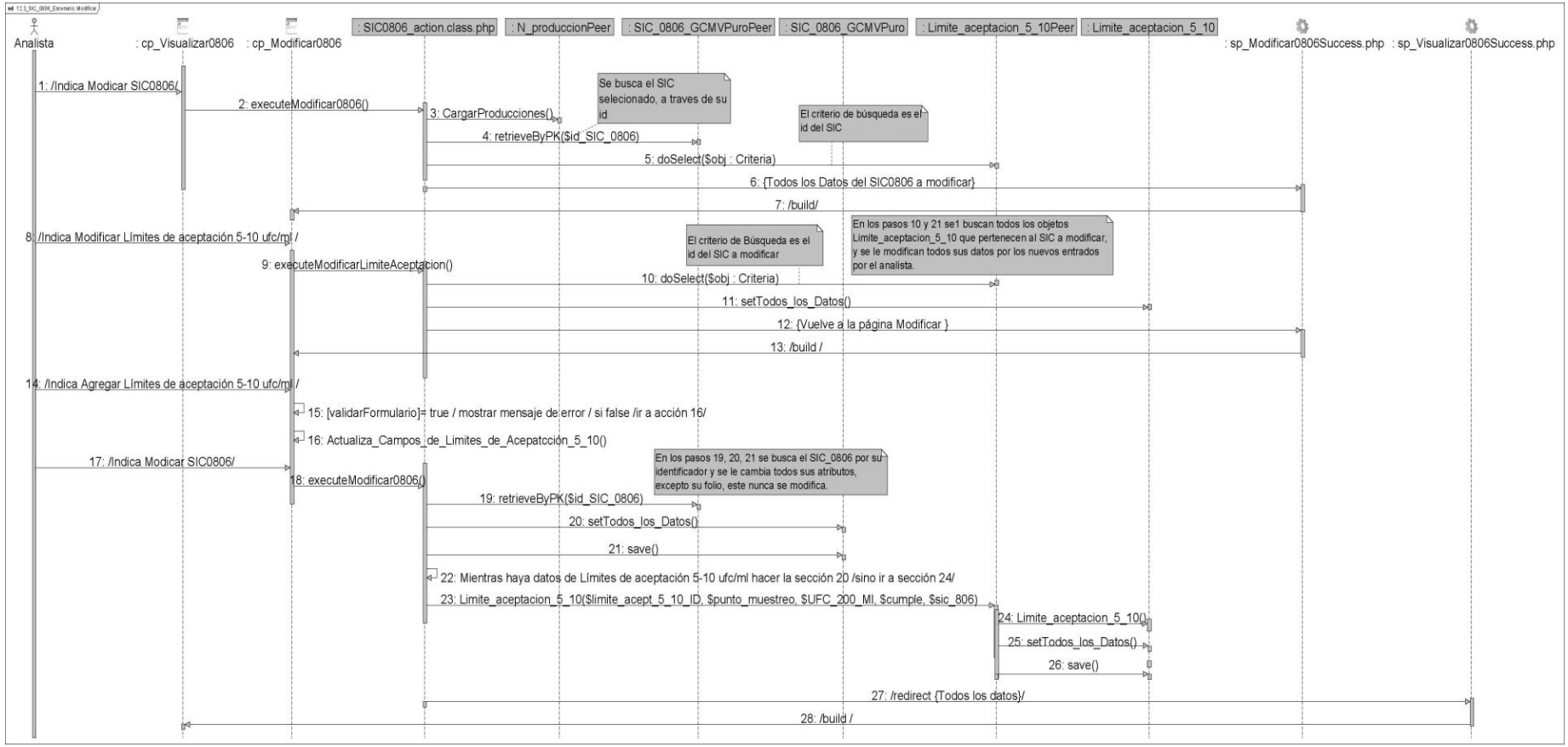

Fig 32: Dsq caso de uso Gestionar Control Microbiológico de Vapor Puro: Escenario Modificar SIC-0806

Caso de Uso: Gestionar Control Muestreo con el Analizador Ambienta.

1- Escenario: Crear SIC-0807

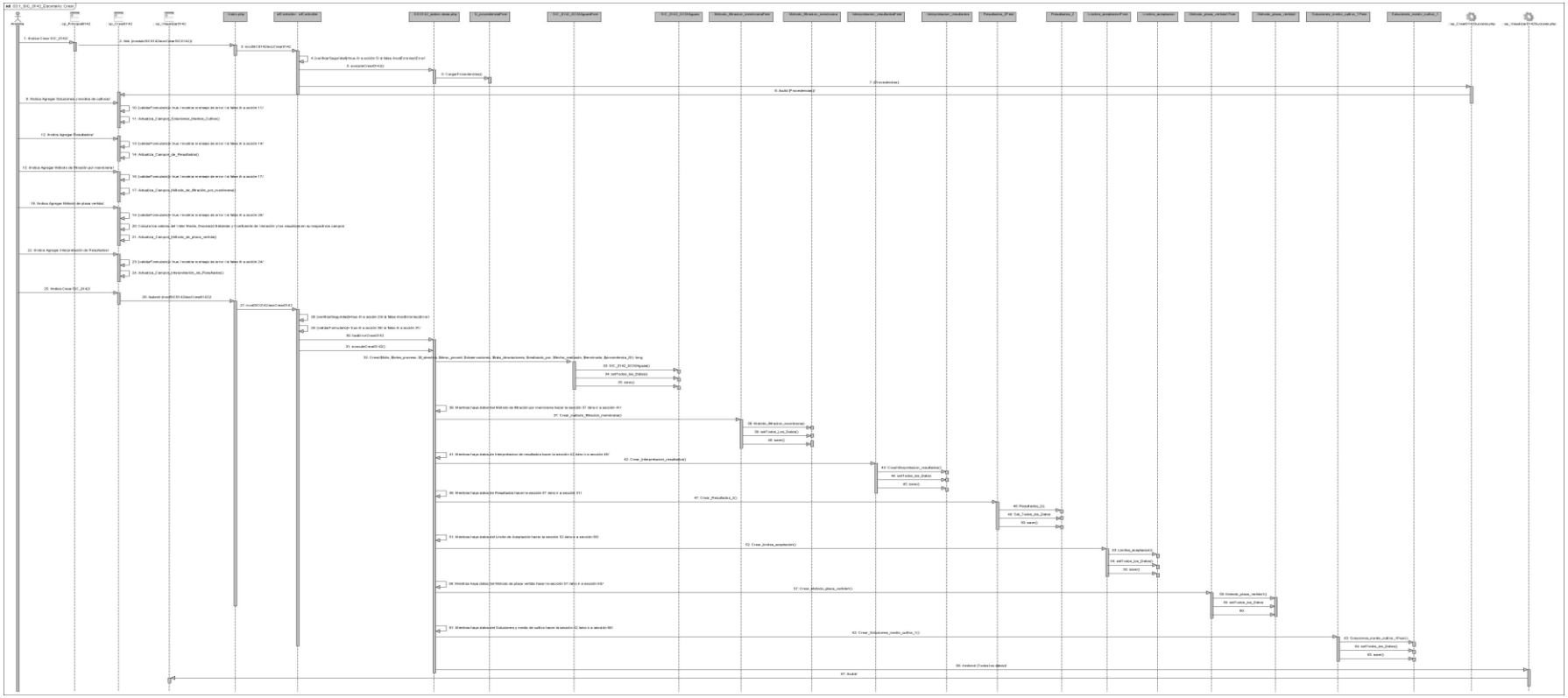

Fig 33: Dsq caso de uso Gestionar Control Muestreo con el Analizador Ambienta: Escenario Crear SIC-0807

2- Escenario: Buscar y Visualizar SIC-0807

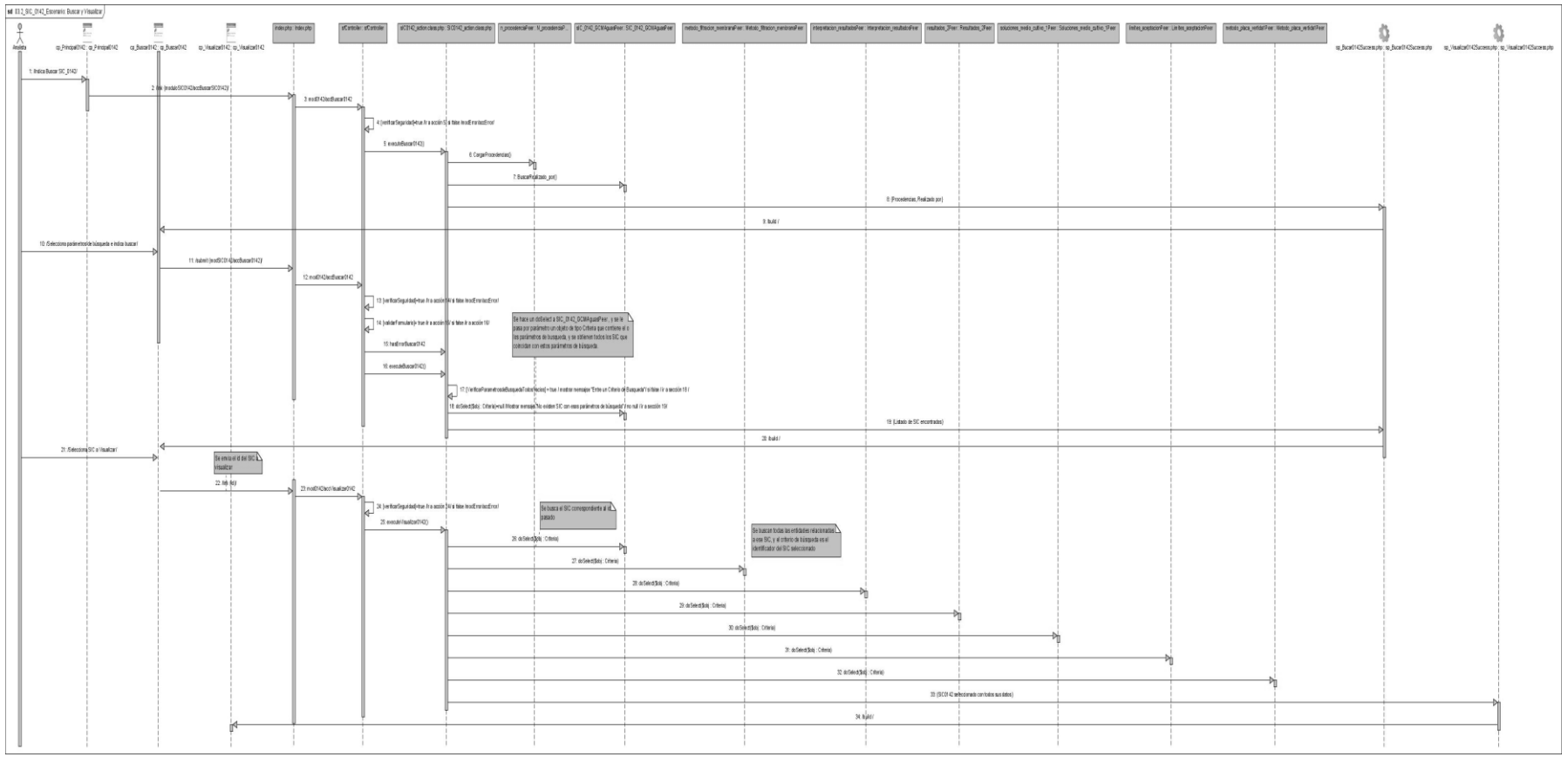

Fig 34: Dsq caso de uso Gestionar Control Muestreo con el Analizador Ambienta: Escenario Buscar y Visualizar SIC-0807

3- Escenario: Modificar SIC-0807

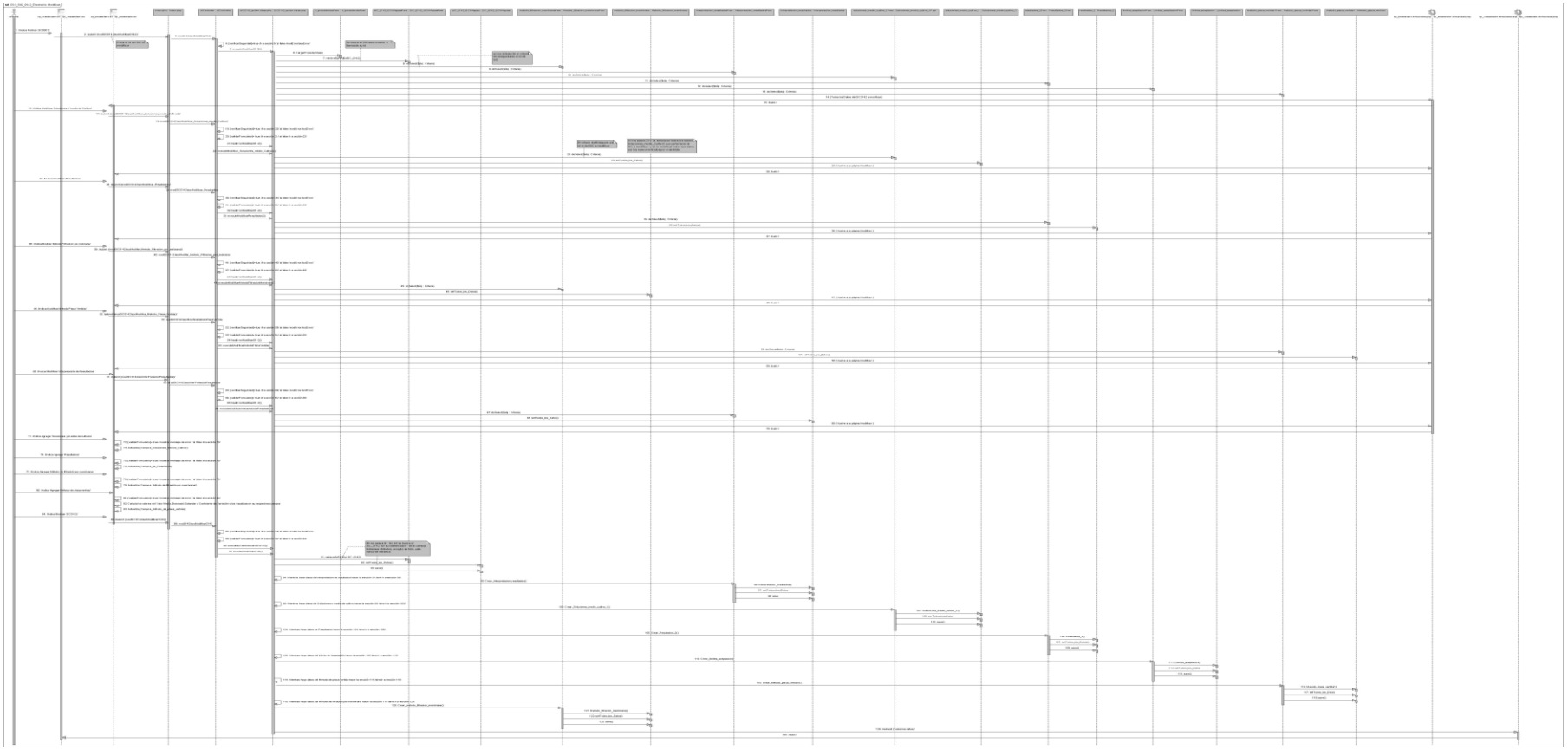

Fig 35: Dsq caso de uso Gestionar Control Muestreo con el Analizador Ambienta: Escenario Modificar SIC-0807

4- Escenario: Generar Reporte SIC-0807

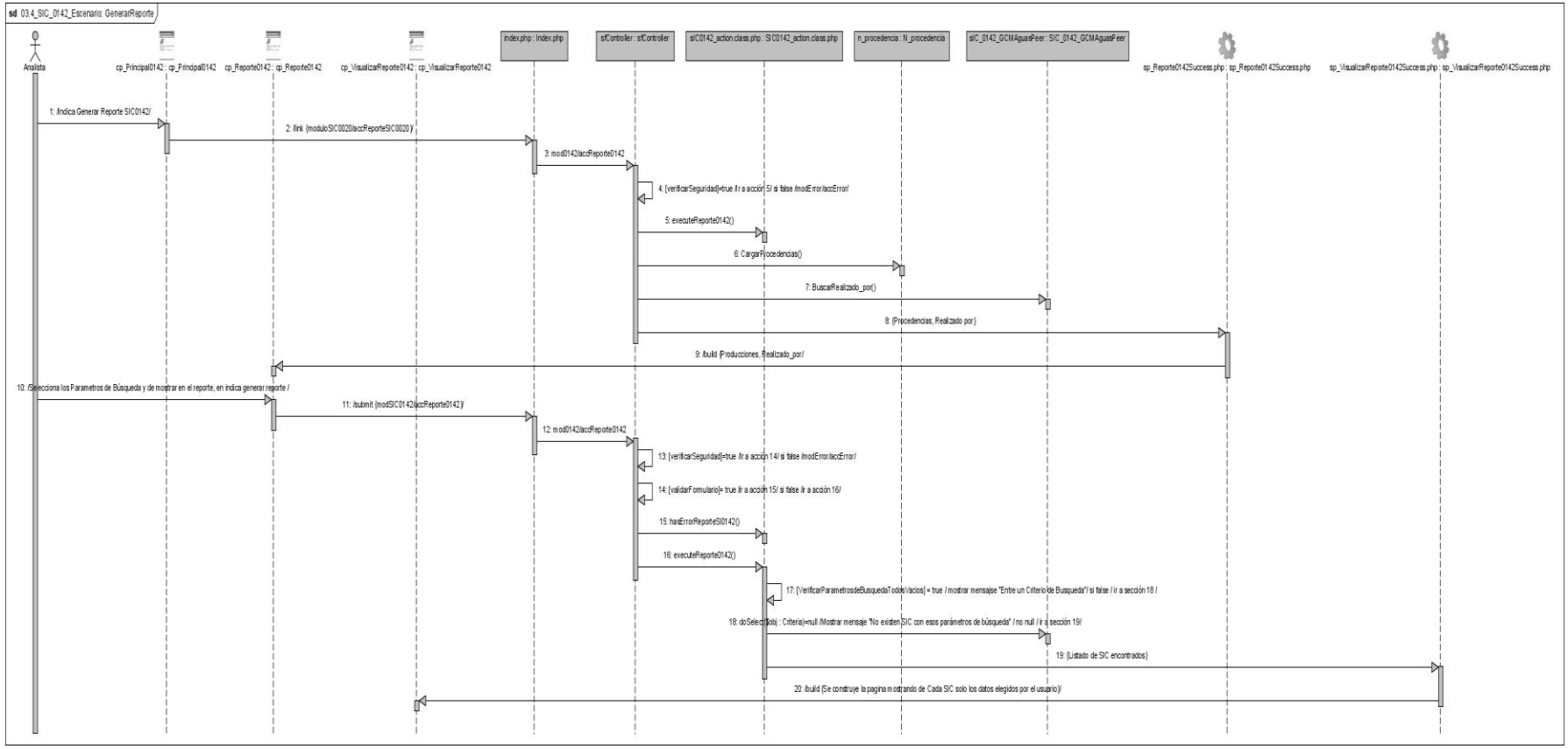

Fig 36: Dsq caso de uso Gestionar Control Muestreo con el Analizador Ambienta: Escenario Generar Reporte SIC-0807

Caso de Uso: Gestionar Chequeo de Viabilidad.

1- Escenario: Crear SIC-0711

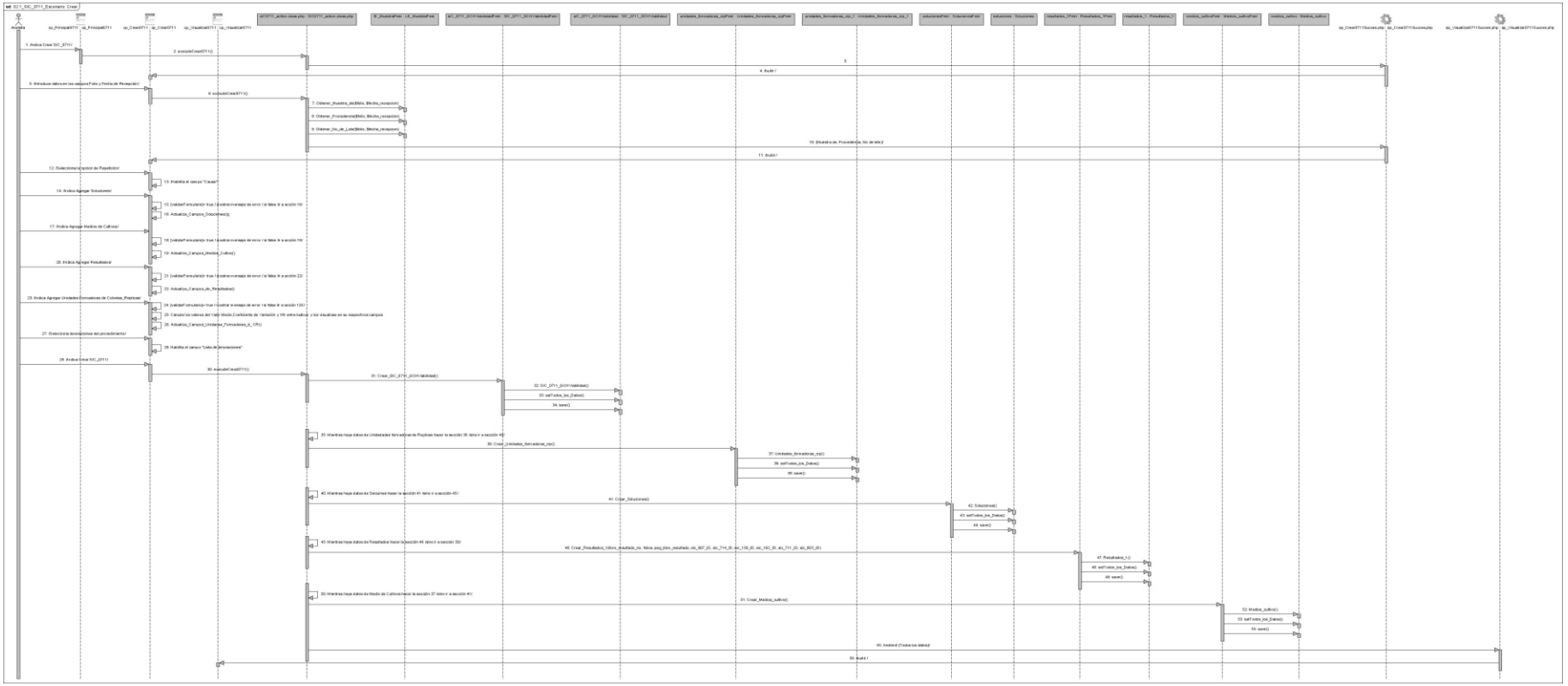

Fig 37: Dsq caso de uso Gestionar Chequeo de Viabilidad: Escenario Crear SIC-0711

2- Escenario: Buscar y Visualizar SIC-0711

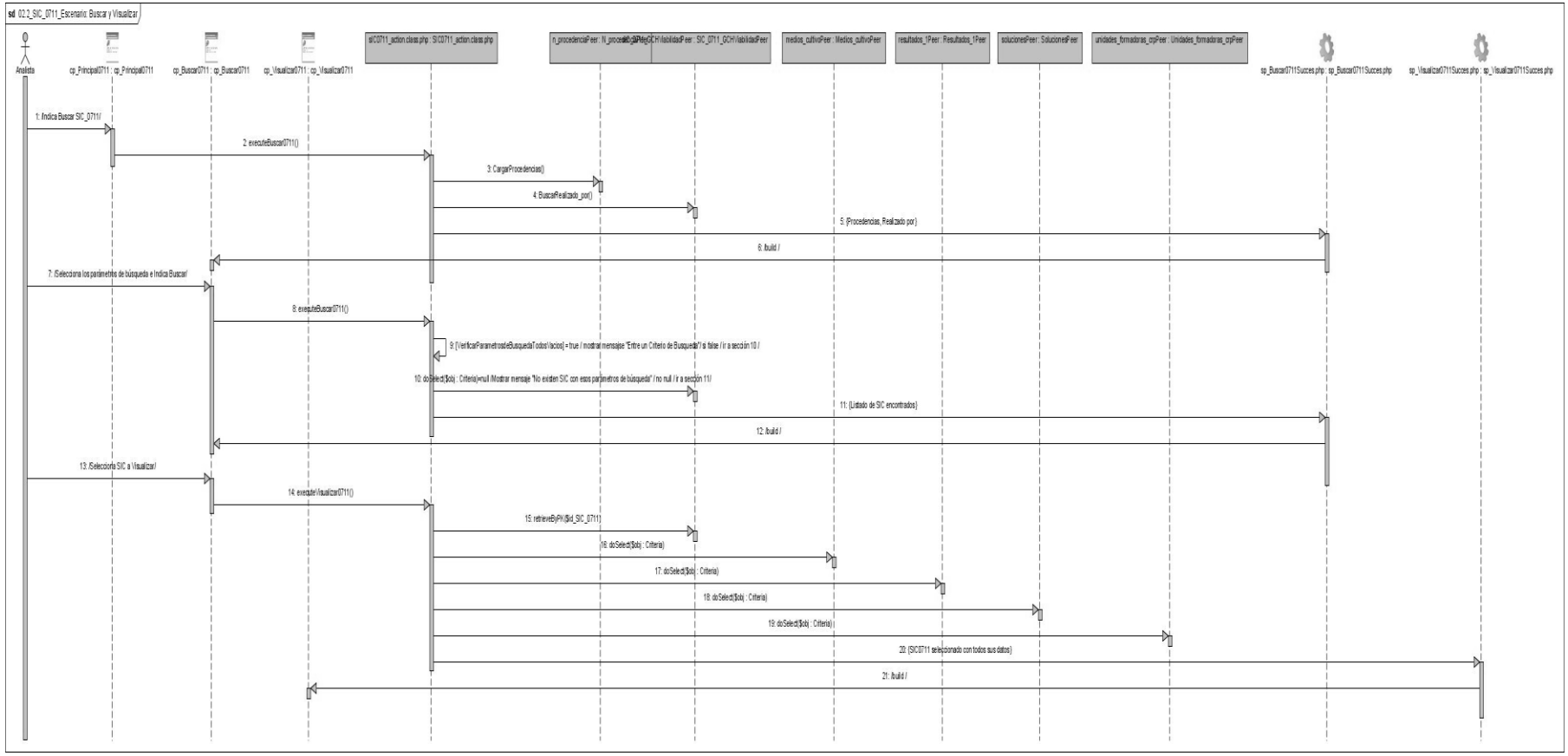

Fig 38: Dsq caso de uso Gestionar Chequeo de Viabilidad: Escenario Buscar y Visualizar SIC-0711

3- Escenario: Modificar SIC-0711

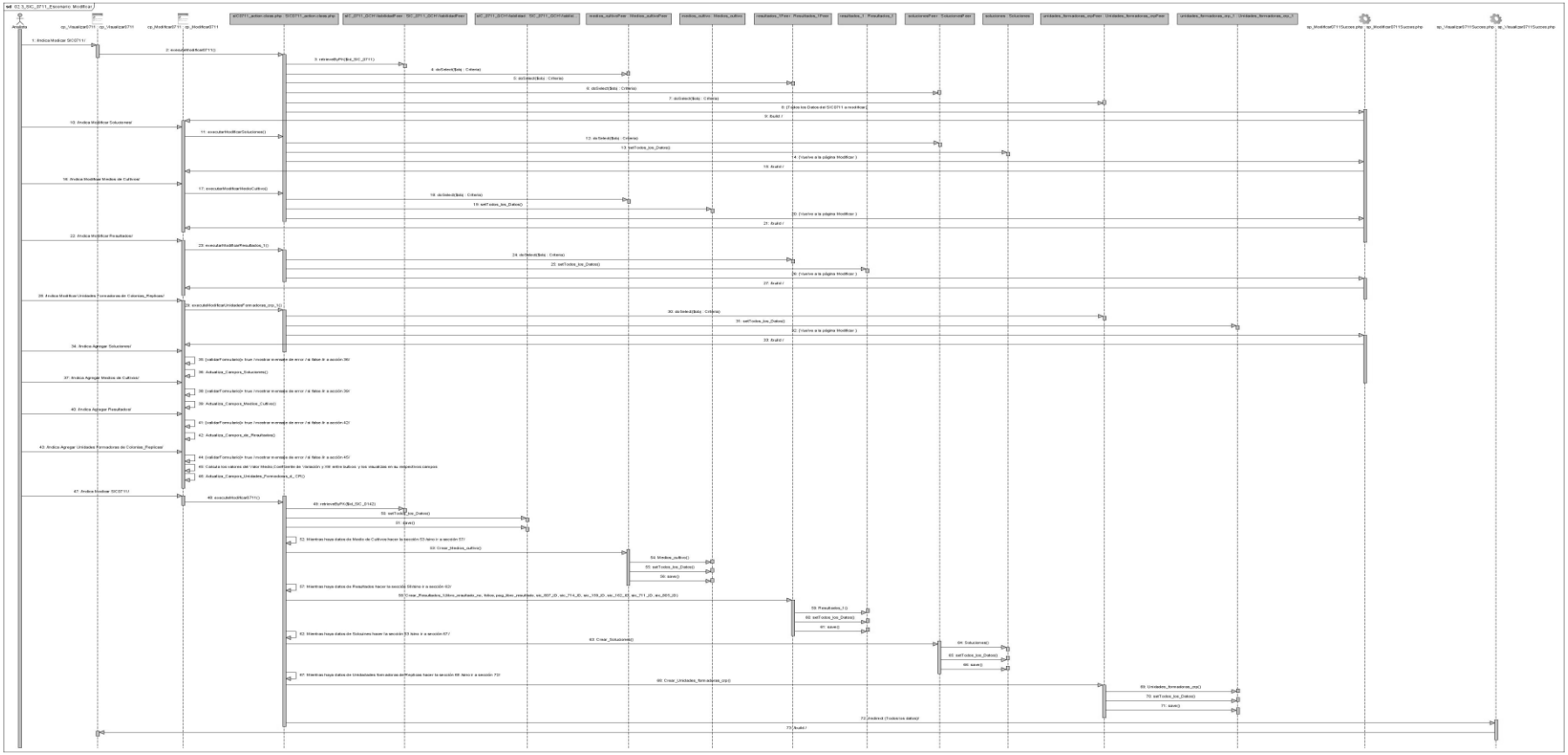

Fig 39: Dsq caso de uso Gestionar Chequeo de Viabilidad Escenario: Escenario Modificar SIC-0711

4- Escenario: Generar Reporte SIC-0711

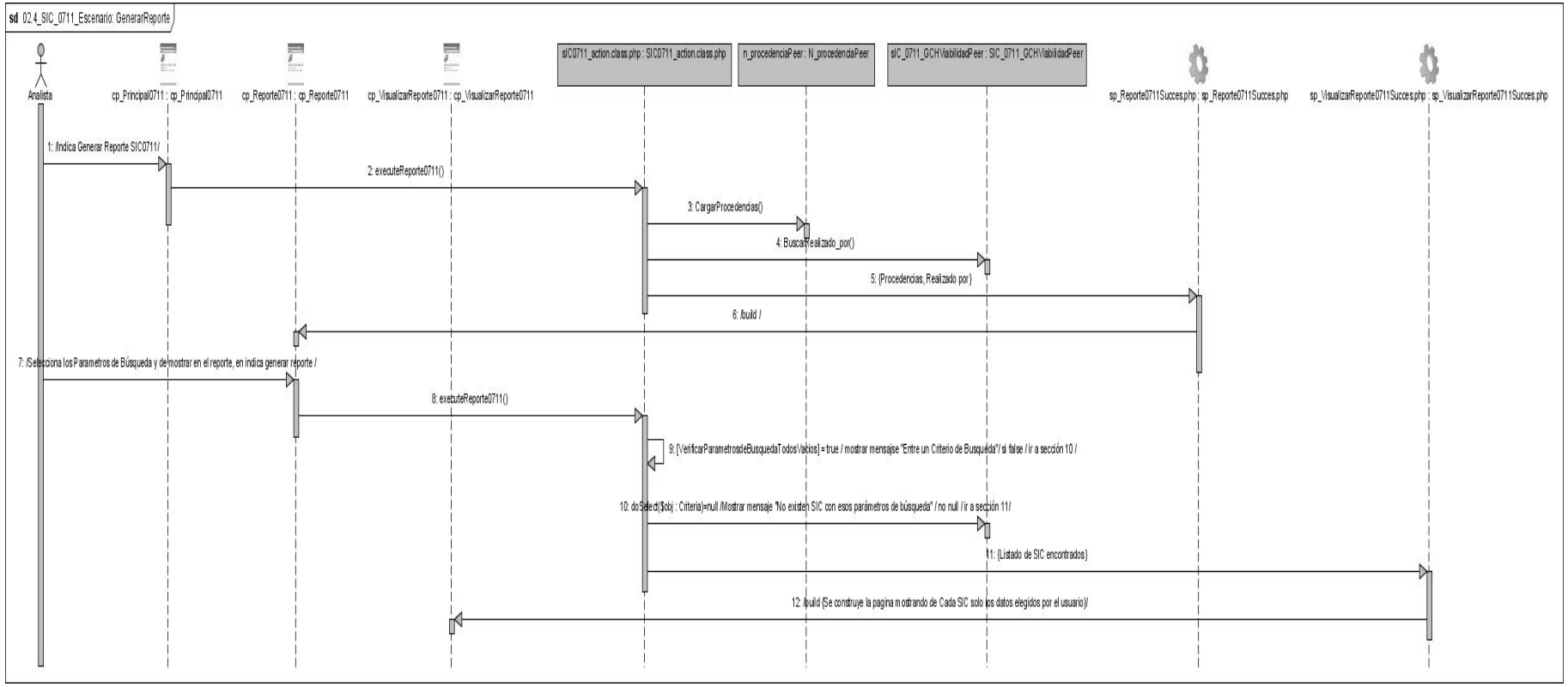

Fig 40: Dsq caso de uso Gestionar Chequeo de Viabilidad Escenario: Escenario Generar Reporte SIC-0711

Caso de Uso: Gestionar Control Ambiental mediante Placa Expuesta.

1- Escenario: Crear SIC-0159

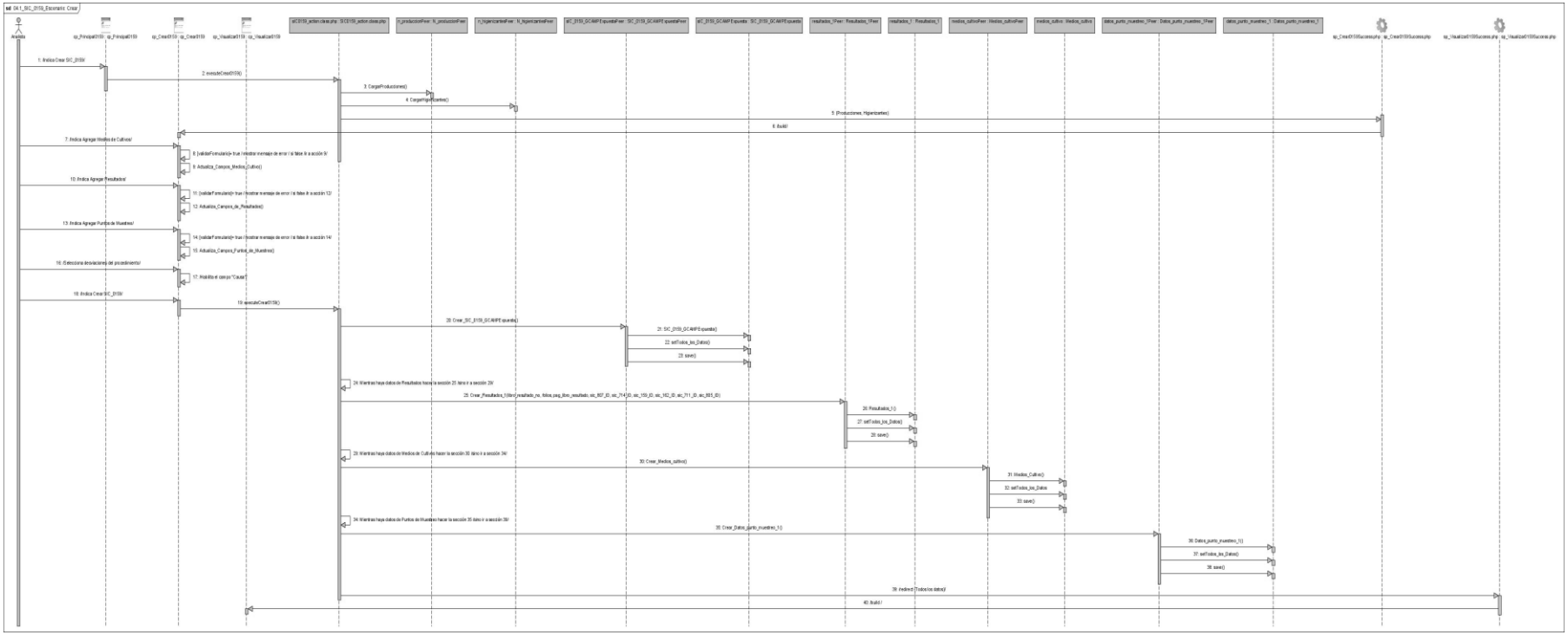

Fig 41: Dsq caso de uso Gestionar Control Ambiental mediante Placa Expuesta: Escenario Crear: SIC-0159

2- Escenario: Buscar y Visualizar SIC-0159

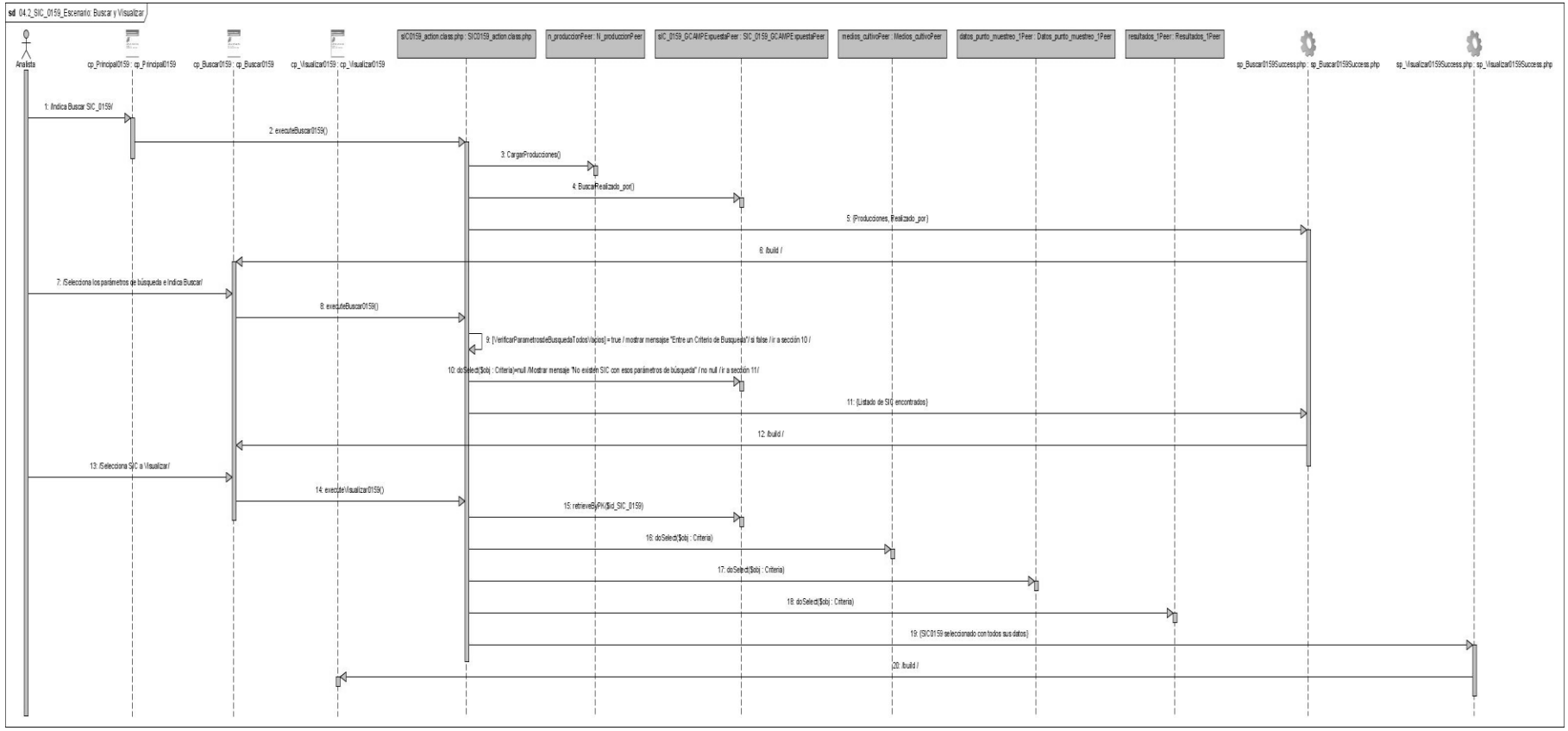

Fig 42: Dsq caso de uso Gestionar Control Ambiental mediante Placa Expuesta: Escenario Buscar y Visualizar SIC-0159

3- Escenario: Modificar SIC-0159

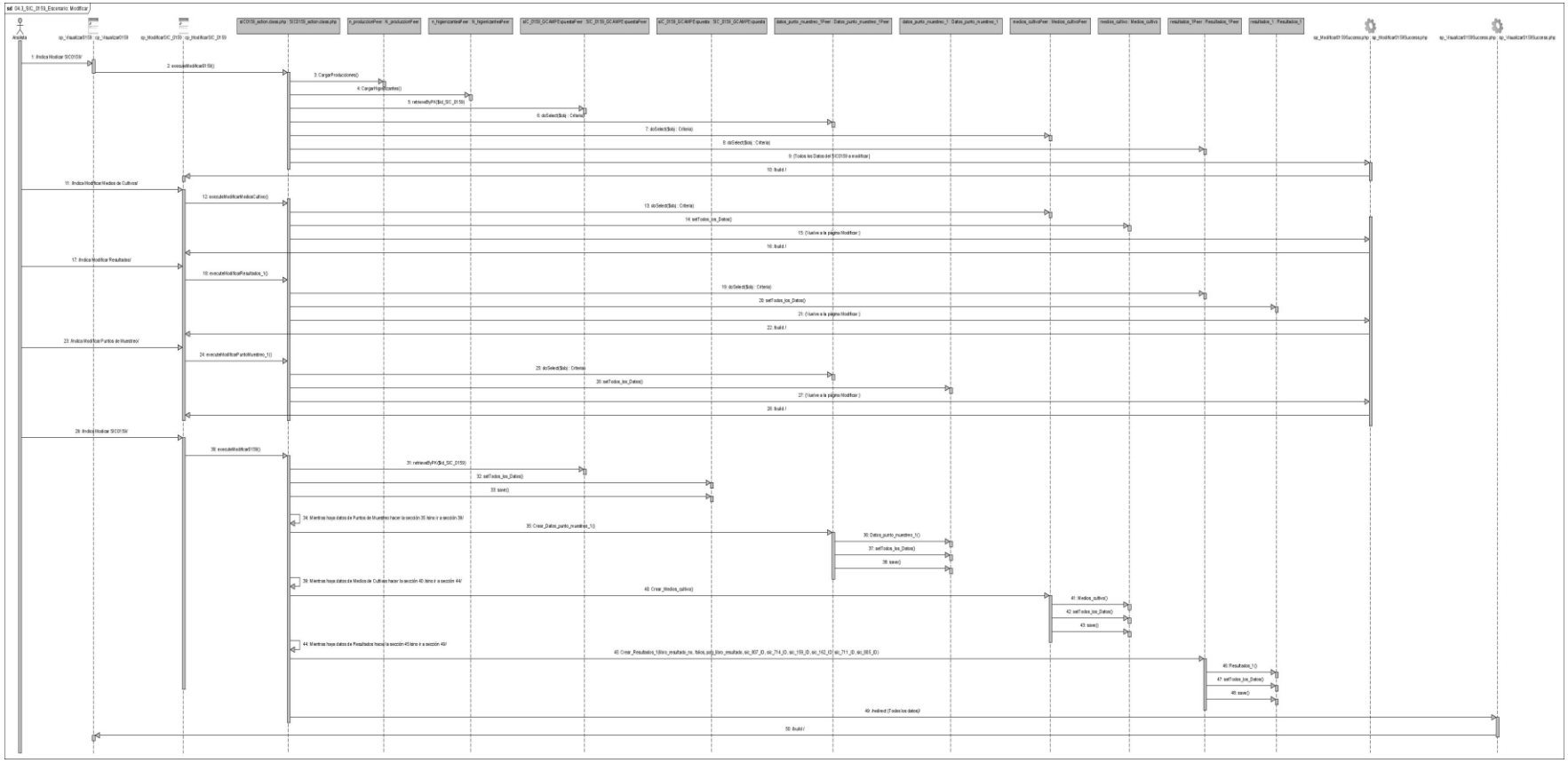

Fig 43: Dsq caso de uso Gestionar Control Ambiental mediante Placa Expuesta: Escenario Modificar SIC-0159

4- Escenario: Generar Reporte SIC-0159

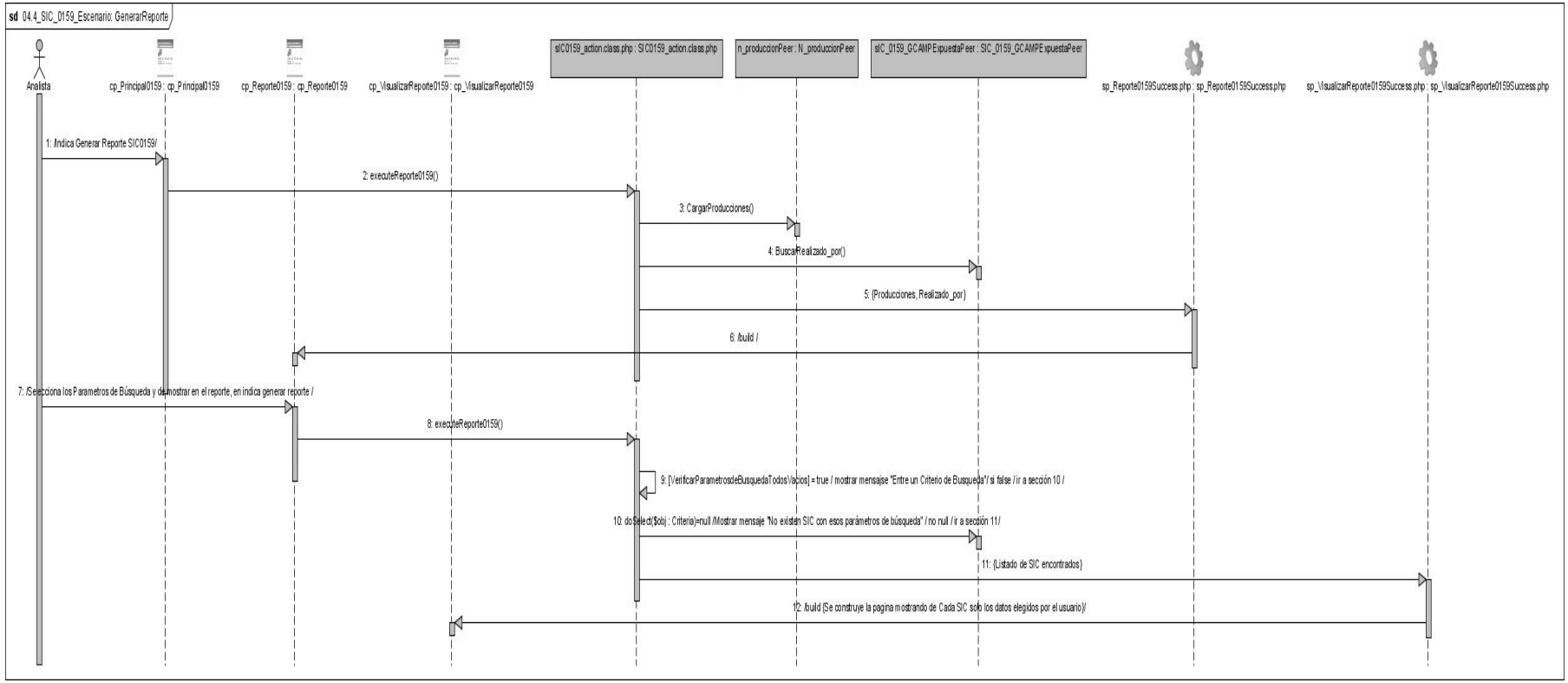

Fig 44: Dsq caso de uso Gestionar Control Ambiental mediante Placa Expuesta: Escenario Generar Reporte SIC-0159

2.5 Validación del Diseño

Después de culminado el diseño de cualquier software, es importante la evaluar la calidad del mismo para comprobar si el objetivo propuesto fue cumplido y si el sistema puede ser implementado sin ambigüedades.

Existen diferentes formas de validar el diseño después de culminado: se puede generar código a través de la herramienta CASE utilizada y comprobar si el mismo es de utilidad para los programadores que implementarán el sistema. En el caso de que el sistema ya se esté implementando, una variante para comprobar la calidad del diseño es comparar el diseño con el código fuente y analizar si existe correspondencia entre ellos, también se pueden realizar encuestas a los programadores para saber en que medida les fue de utilidad el diseño para desarrollar la implementación.

Cómo ya para este módulo se está programando para la validación del diseño se comparará el mismo con fragmentos de códigos ya programado y veremos si existe correspondencia entre ellos.

A continuación se muestra los diagramas de secuencia y parte del código de la aplicación Web referente a los escenarios del CU Gestionar Control Microbiológico de Vapor Puro.

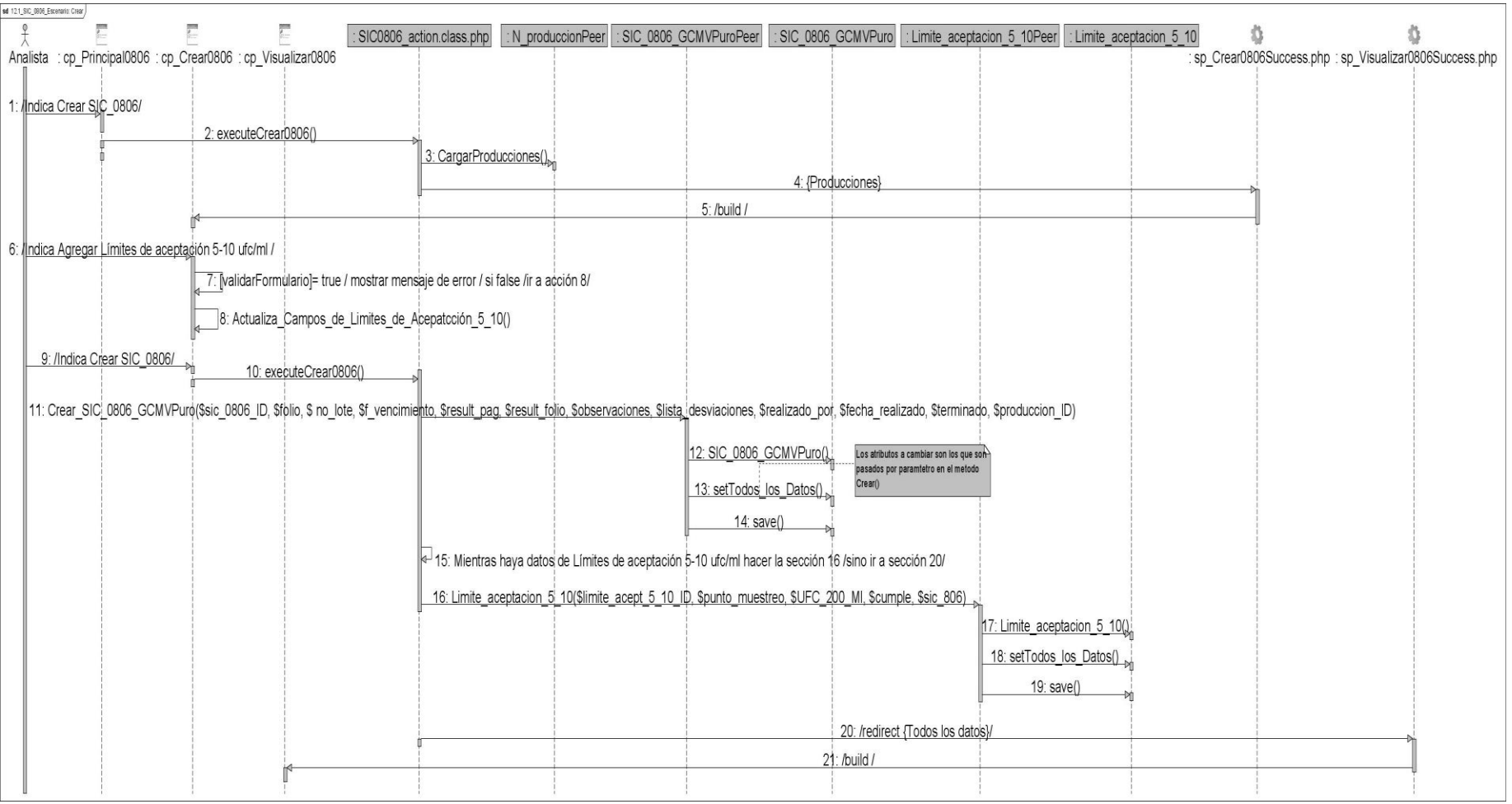

Fig 45: Dsq caso de uso Gestionar Control Microbiológico de Vapor Puro: Escenario Crear: SIC-0806

Función Crear0806() de la clase SIC0806_Action.php del módulo SIC-0806.

```
public function executeCrear0806()
      {
            $this->prod=NProduccionPeer::CargarProd();

            if ($this->getRequest()->getMethod() != sfRequest::POST)
            {
                  $this->mensaje='';
                  return sfView::SUCCESS;

            }
            else
            {
                  $folio=$this->getRequestParameter('crear_folio');
                  $lote=$this->getRequestParameter('crear_lote');
                  $fecha=$this->getRequestParameter('fecha');
                  $fecha1=$this->getRequestParameter('fecha1');
                  $resultado_pag=$this->getRequestParameter('resultado_pag');
                  $folio1=$this->getRequestParameter('folio1');
                  $observaciones=$this->getRequestParameter('observaciones');
                  $determine_ufc=$this->getRequestParameter('determine_ufc');
                  $desv=$this->getRequestParameter('desv');
                  $lista_desv=$this->getRequestParameter('lista_desv');
                  $nombre=$this->getRequestParameter('nombre');
                  $fecha2=$this->getRequestParameter('fecha2');
                  $terminado=$this->getRequestParameter('crear_terminado');
                  $produccion=$this->getRequestParameter('produccion');


      $id=Sic0806GcmvpuroPeer::Crear($folio,$lote,$fecha,$fecha1,$resultado_pag,$folio
1,$observaciones,$determine_ufc,
                  $desv,$lista_desv,$nombre,$fecha2,$terminado,$produccion);

                  $muestreo=$this->getRequestParameter('muestreo');
                  $arr_muestreo= explode('*', $muestreo);

                  $ufc=$this->getRequestParameter('ufc');
                  $arr_ufc= explode('*', $ufc);

                  $cumple=$this->getRequestParameter('cumplex[]');
                  $arr_cumple= explode('*', $cumple);

                  for($i=0; $i< count($arr_muestreo)-1; $i++)

      LimiteAceptacion510Peer::CrearLimitesAceptacion510($id,$arr_muestreo[$i],$arr_uf
c[$i],$arr_cumple[0],0);

                  LimiteAceptacion510Peer::InsertarLimitesAceptacion510($id);
```

```
                $this->redirect('SIC0806/Visualizar0806?enviar='.$id);
        }
}
```

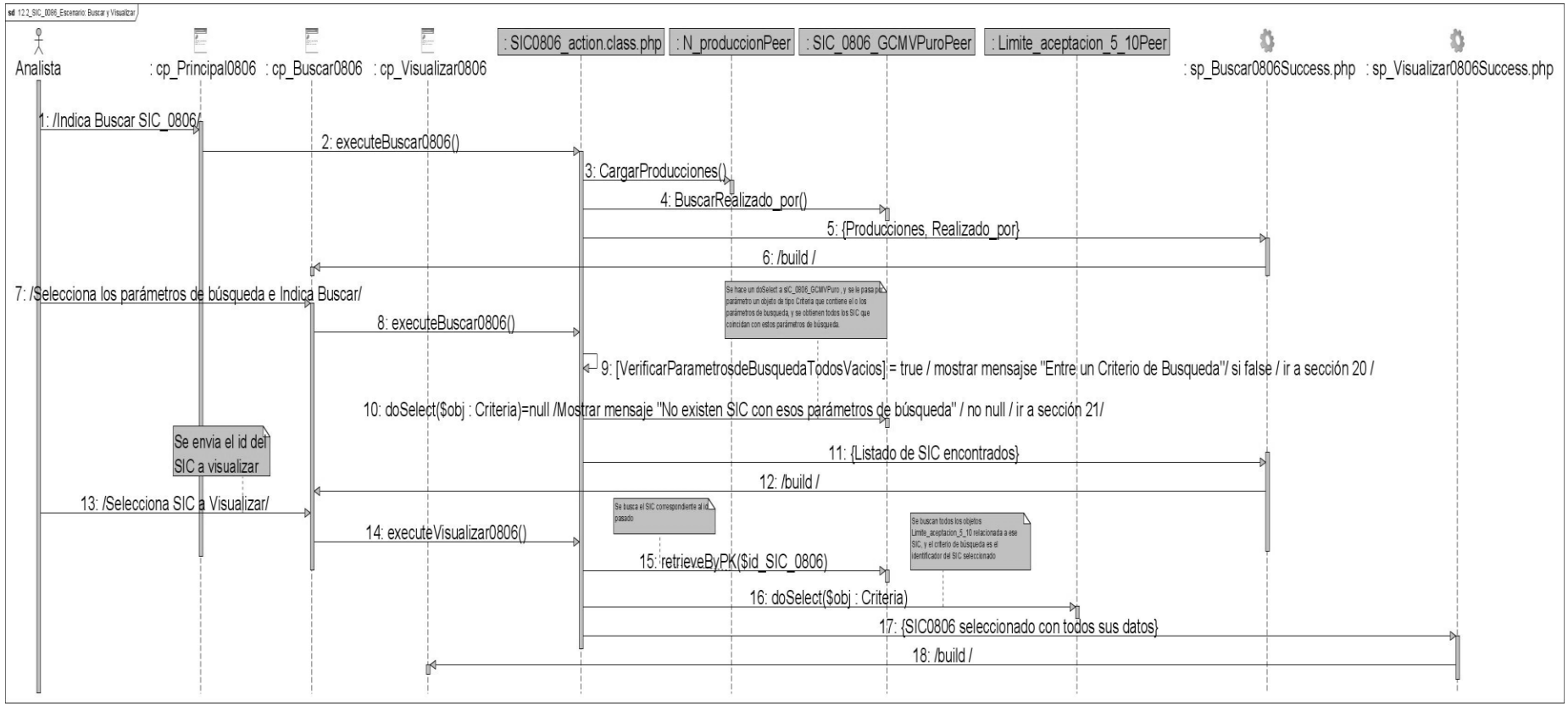

Fig 46: Dsq caso de uso Gestionar Control Microbiológico de Vapor Puro: Escenario Buscar y Visualizar SIC-0806

Función Buscar0806() de la clase SIC0806_Action.php del módulo SIC-0806.

```
public function executeBuscar0806()
      {
            $verdep=array();
            $verrealizado=array('Select...');

            $lista_dep=NProduccionPeer::CargarProd();
            $lista_realpor=Sic0806GcmvpuroPeer::BuscarRealizadoPor();

            for($i=0;$i<count($lista_dep);$i++)
            array_push($verdep,$lista_dep[$i]);

            for($i=0;$i<count($lista_realpor);$i++)
            array_push($verrealizado,$lista_realpor[$i]);

            $this->select_dep=$verdep;
            $this->select_reali=$verrealizado;

            $prod=$this->getRequestParameter('produccion');
            $real=$this->getRequestParameter('realizado_por');

            $f_i = $this->getRequestParameter('fecha_inicial');
            $f_f= $this->getRequestParameter('fecha_final');

            if($prod==0 && $real==0)
            {
                  $this->mensaje='Entre un Criterio de Busqueda';
                  $this->ver=-1;
            }
            else
            {
                  $dev=array();
                  $dev_arreglo=array();
                  $dev1=array();

                  $c= new Criteria();
                  $this->mensaje='';
                  if($prod!=0)
                  {
                        $produccion=$verdep[$prod];
                        $c->add(NProduccionPeer::DESCRIPCION,$produccion);
                  }
                  if($real!=0)
                  {
                        $realiza=$verrealizado[$real];
                        $c->add(Sic0806GcmvpuroPeer::REALIZADO_POR,$realiza);
                  }
                  $sic0806=Sic0806GcmvpuroPeer::doSelect($c);
                  $dev1=Sic0806GcmvpuroPeer::BuscarArregPrincipal1($sic0806);
```

```
                  if($f_f!='' && $f_i!='')
                  {
                        $fecha_i=str_replace('-','',$f_i);
                        $fecha_f=str_replace('-','',$f_f);


                        $fecha=array();

                        for($i=0;$i<count($dev1);$i++)
                        {
                              $arr_aux=array();
                              $fecha=$dev1[$i]->getFecha();
                              if($fecha>=$fecha_i && $fecha<=$fecha_f)
                              array_push($arr_aux,$dev1[$i]);
                        }
                        $dev_arreglo=$arr_aux;

                  }

                  else
                  $dev_arreglo=$dev1;

                  $this->mensaje='';
                  $this->ver=$dev_arreglo;
            }

      }
```

Función Visualizar0806() de la clase SIC0806_Action.php del módulo SIC-0806.

```
public function executeVisualizar0806()
      {
            $id=$this->getRequestParameter('enviar');
            $this->pepe=$id;
            $arreglo=Sic0806GcmvpuroPeer::retrieveByPK($id);
            if(count($arreglo)>0)
            $this->ver=Sic0806GcmvpuroPeer::BuscarArregPrincipal($arreglo);
            else
            $this->ver=$id;

            $arre=ReactivosPeer::BuscarUltFolioReacticos($id);


            $this-
>verr=LimiteAceptacion510Peer::BuscarArregloLimitesAceptacion510($arre);
```

```
        if($this->getRequestParameter('crear_terminado')==1)
        $this->mensaje="Esta Terminado";
        else
        $this->mensaje="No esta Terminado";
    }
```

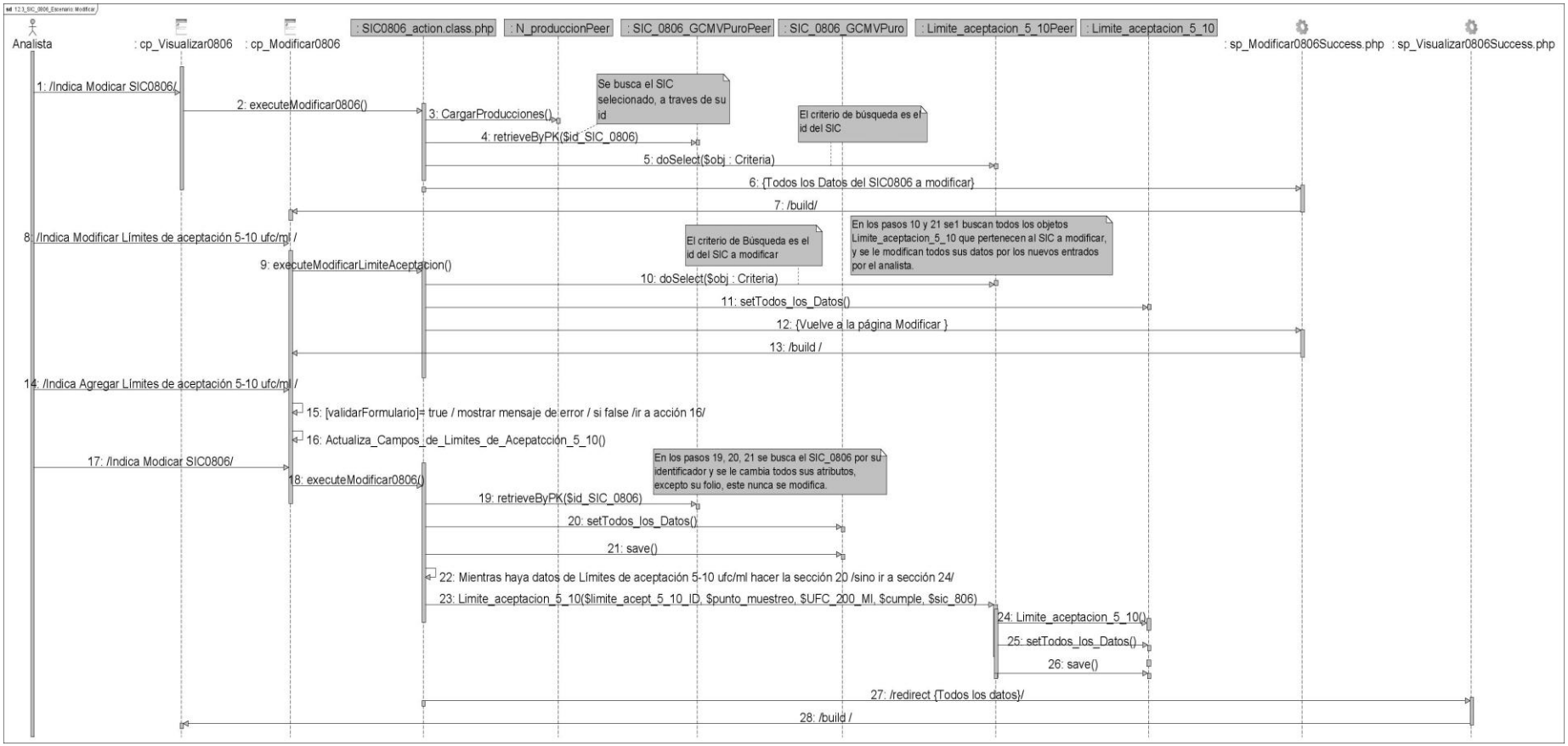

Fig 47: Dsq caso de uso Gestionar Control Microbiológico de Vapor Puro: Escenario Modificar SIC-0806

Función Modificar0806() de la clase SIC0806_Action.php del módulo SIC-0806.

```
public function executeModificar0806()
      {
            if ($this->getRequest()->getMethod() != sfRequest::POST)
            {
                  $id=$this->getRequestParameter('enviar');

                  $this->ver=$id;

      $var=LimiteAceptacion510Peer::BuscarUltFolioLimitesAceptacion510($id);

                  $this-
>dev=LimiteAceptacion510Peer::BuscarArregloLimitesAceptacion510($var);

                  $this->dev1=Sic0806GcmvpuroPeer::BuscarPorFolio($id);
                  return sfView::SUCCESS;
            }
            else
            {
                  $folio=$this->getRequestParameter('crear_folio');


                  $muestreo=$this->getRequestParameter('');
                  $arr_muestreo= explode('*', $muestreo);

                  $ufc_=$this->getRequestParameter('');
                  $arr_ufc= explode('*', $ufc);

                  $cumple=$this->getRequestParameter('');
                  $arr_cumple= explode('*', $cumple);


                  for($i=0; $i< count($arr_muestreo)-1; $i++)

      ReactivosPeer::CrearReactivos($folio,$arr_muestreo[$i],$arr_ufc[$i],$arr_cumple[
$i],1);


                  $no_lote=$this->getRequestParameter('crear_lote');
                  $fecha_muestreo=$this->getRequestParameter();
                  $f_vencimiento=$this->getRequestParameter('fecha');
                  $result_pag=$this->getRequestParameter('resultado_pag');
                  $result_folio=$this->getRequestParameter('folio1');
                  $observaciones=$this->getRequestParameter('observaciones');
                  $ufc_100mi=$this->getRequestParameter('determine_ufc');
                  $desv_proced=$this->getRequestParameter('desv');
                  $lista_desviaciones=$this->getRequestParameter('lista_desv');
                  $realizado_por=$this->getRequestParameter('nombre');
                  $fecha_realizado=$this->getRequestParameter('fecha2');
```

```
                $terminado=$this->getRequestParameter('crear_terminado');

                $SIC0806=Sic0806GcmvpuroPeer::retrieveByPK($folio);

                $SIC0806->setNoLote($no_lote);
                $SIC0806->setFechaMuestreo($fecha_muestreo);
                $SIC0806->setFVencimiento($f_vencimiento);
                $SIC0806->setResultPag($result_pag);
                $SIC0806->setResultFolio($result_folio);
                $SIC0806->setObservaciones($observaciones);
                $SIC0806->setUfc100mi($ufc_100mi);
                $SIC0806->setDesvProced($desv_proced);
                $SIC0806->setListaDesviaciones($lista_desviaciones);
                $SIC0806->setRealizadoPor($realizado_por);
                $SIC0806->setFechaRealizado($fecha_realizado);
                $SIC0806->setTerminado($terminado);

                $SIC0806->save();

                if($this->getRequestParameter('crear_terminado')!=1)
                $this->mensaje="Esta Terminado";
                else
                $this->mensaje="No esta Terminado";
                $this->redirect('SIC0806/Visualizar0806?enviar='.$ver);

        }
```

Función ModificarLimiteAceptacion510() de la clase SIC0806_Action.php del módulo SIC-0806.

```
public function executeModificarLimiteAceptacion510()
      {
            $ban=false;
            $id=$this->getRequestParameter('folio');

      $arr=LimiteAceptacion510Peer::BuscarparaVisualizarLimitesAceptacion510($id);
            for($i=0;$i<count($arr);$i++)
            {
                  $punto_muestreo=$this->getRequestParameter('pto_muestreo'.$i);
                  $ufc_200_ml=$this->getRequestParameter('ufc200'.$i);
                  $cumple=$this->getRequestParameter('cumple'.$i);


                  if($punto_muestreo=='' || $ufc_200_ml=='' || $cumple=='')
                  $ban=true;
            }
            if(!$ban)
            {
                  $this->ver_pruebas='Modificados Correctamente';
```

```
                $i=0;
                while($this->getRequestParameter('pto_muestreo'.$i))
                {
                      $punto_muestreo=$this->getRequestParameter('pto_muestreo'.$i);
                      $ufc_200_ml=$this->getRequestParameter('ufc200'.$i);
                      $cumple=$this->getRequestParameter('cumple'.$i);


                      $sic0806 =LimiteAceptacion510Peer::retrieveByPK($this-
>getRequestParameter('lolo'.$i));

                      $sic0806->setPuntoMuestreo($punto_muestreo);
                      $sic0806->setUfc200Ml($ufc_200_ml);
                      $sic0806->setCumple($cumple);

                      $sic0806->save();
                      $i++;
                }
          }
          else
          $this->ver_pruebas='Ente Todos Los Datos Bien';}
```

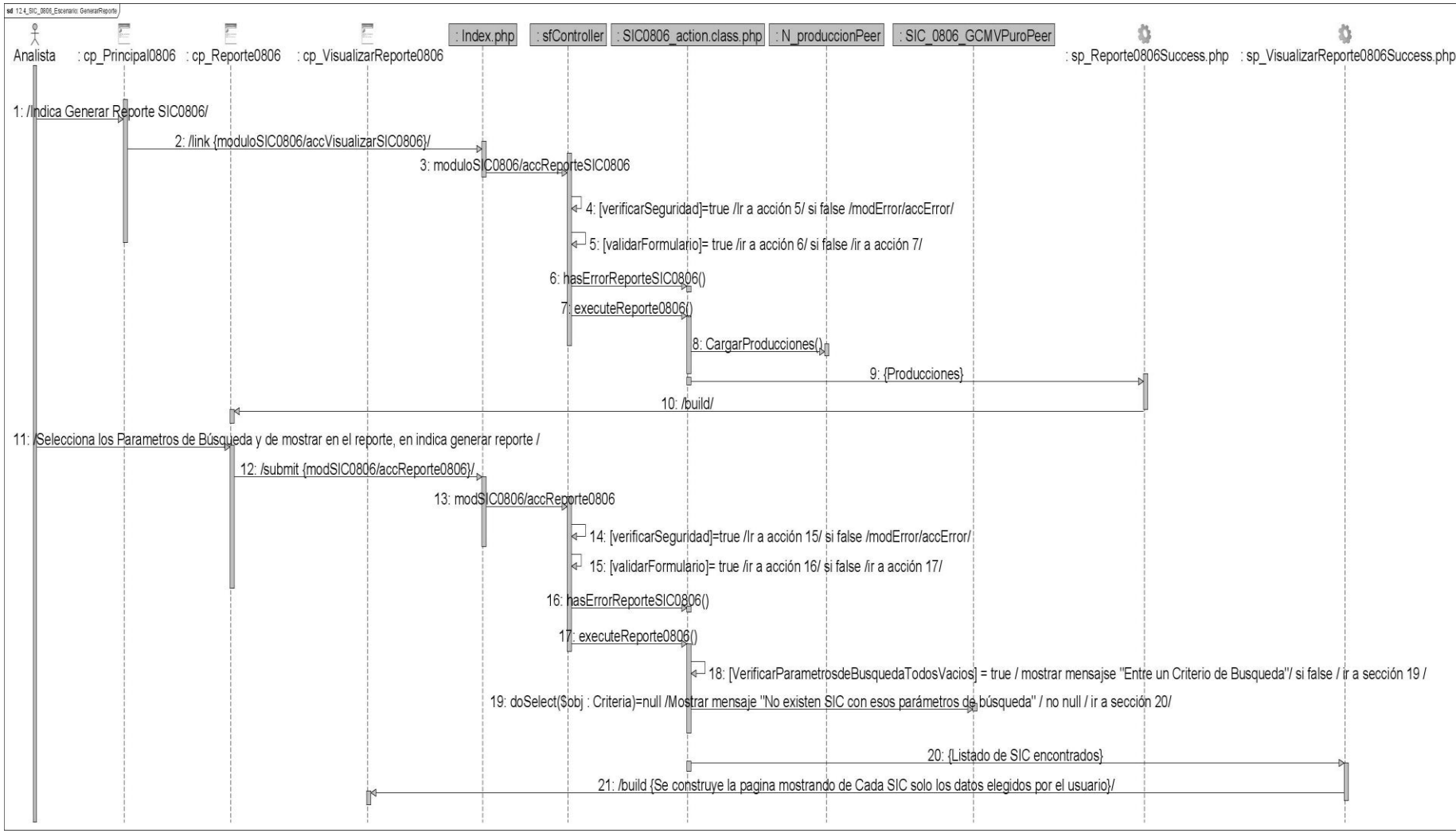

Fig 48: Dsq caso de uso Gestionar Control Microbiológico de Vapor Puro: Escenario Generar Reporte SIC-0806

Función Reporte0806() de la clase SIC0806_Action.php del módulo SIC-0806.

```
public function executeReporte0806()
      {
            $prod=NProduccionPeer::CargarProd();


            $nombre_realizador=array('Select...');

            //$lista_realpor=Sic0159GcampexpuestaPeer::BuscarRealizadoPor();


            for($i=0;$i<count($lista_realpor);$i++)
            array_push($lista_realpor[$i]);

            $this->lista_produccion=$prod;
            //$this->lista_nomb=$nombre_realizador;

            //$nueva=$this->getRequestParameter('Nombre');
            $nueva1=$this->getRequestParameter('Produccion');

            if($nueva==0 && $nueva1==0)
            {
                  $this->mensaje='Usted Debe Seleccionar la Producción';
            }
            else
            {
                  $c= new Criteria();
                  $this->mensaje='';
            }
      }
```

A continuación se muestran las funciones utilizadas en los diagramas de secuencia que pertenecen a las clases “Peer” implicadas en este caso de uso.

Sic0806GcmvpuroPeer

```
class Sic0806GcmvpuroPeer extends BaseSic0806GcmvpuroPeer
{
      public static function
Crear($folio,$lote,$fecha,$fecha1,$resultado_pag,$folio1,$observaciones,
      $determine_ufc,$desv,$lista_desv,$nombre,$fecha2,$terminado,$produccion)
      {
                  $SIC0806=new Sic0806Gcmvpuro();
```

```
	$SIC0806->setFolio($folio);
	$SIC0806->setNoLote($lote);
	$SIC0806->setFechaMuestreo($fecha);
	$SIC0806->setFVencimiento($fecha1);
	$SIC0806->setResultPag($resultado_pag);
	$SIC0806->setResultFolio($folio1);
	$SIC0806->setObservaciones($observaciones);
	$SIC0806->setUfc100mi($determine_ufc);
	$SIC0806->setDesvProced($desv);
	$SIC0806->setListaDesviaciones($lista_desv);
	$SIC0806->setRealizadoPor($nombre);
	$SIC0806->setFechaRealizado($fecha2);
	$SIC0806->setProduccion($produccion);

	if($terminado==1)
	$SIC0806->setTerminado('1');
	else
	$SIC0806->setTerminado('0');

	$SIC0806->save();
	return $SIC0806->getSic0806Id();

}


public static function BuscarRealizadoPor()
{
	$arr=array();
	$arr1=array();
	$c = new Criteria();
	$sic0806=Sic0806GcmvpuroPeer::doSelect($c);
	for($i=0;$i<count($sic0806);$i++)
	array_push($arr,$sic0806[$i]->getRealizadoPor());

	array_push($arr1,$arr[0]);
	for($i=1;$i<count($arr);$i++)
	{   $aux=true;
	for($j=0;$j<count($arr1);$j++)
	{
		$aux=true;
		if($arr[$i]==$arr1[$j])
		break;
		else
		$aux=false;
	}
	if($aux==false)
	array_push($arr1,$arr[$i]);

	}
	return $arr1;
```

```
	}

}
```

NProduccionPeer

```
class NProduccionPeer extends BaseNProduccionPeer
{
	public static function CargarProd()
	{

	 $arre=array();
	 $arre1=array();

	 $re=NProduccionPeer::doSelect(new Criteria());

	 for($i=0;$i<count($re);$i++)
	 array_push($arre,$re[$i]->getDescripcion());

	 array_push($arre1,$arre[0]);

	 for($i=1;$i<count($arre);$i++)
	 {   $aux=true;
	 for($j=0;$j<count($arre1);$j++)
	 {
	      $aux=true;
	      if($arre[$i]==$arre1[$j])
	      break;
	      else
	      $aux=false;
	 }
	 if($aux==false)
	 array_push($arre1,$arre[$i]);

	 }
	 return $arre1;
	}
}
```

LimiteAceptacion510Peer

```
class LimiteAceptacion510Peer extends BaseLimiteAceptacion510Peer
{
	public static function
CrearLimitesAceptacion510($folio,$punto_muestreo,$ufc_200_ml,$cumple)
	{
```

```
            $sic0806 = new LimitesAceptacion510();
            $sic0806->setPuntoMuestreo($punto_muestreo);
            $sic0806->setUfc200Ml($ufc_200_ml);
            $sic0806->setCumple($cumple);
            $sic0806->setSic0806Id($folio);

            $sic0806->save();

      }
}
```

Como se pudo apreciar en lo anteriormente expuesto existe correspondencia entre el diseño realizado y el código implementado. Por lo que se puede concluir que los programadores pudieron implementar la aplicación Web sin ambigüedades a partir del diseño realizado.

2.6 Conclusiones

En este capítulo se definieron las clases que se corresponden con el diseño de la aplicación Web, se realizaron los diagramas de clases para todos de los casos de uso identificados así como sus realizaciones del diseño, de acuerdo al patrón arquitectónico utilizado, Modelo-Vista-Controlador.

Conclusiones

Se realizó el diseño del Módulo de Microbiología para el Sistema de Gestión de la Información de los Laboratorios de la Dirección de Calidad del CIGB, identificando los elementos y clases, así como sus relaciones en correspondencia con las especificaciones que exige el uso del framework Symfony el cual se rige bajo el patrón arquitectónico Modelo-Vista-Controlador, facilitando la comunicación entre los analistas y el equipo de desarrollo.

Recomendaciones

- ✓ Debido al constante cambio existente en las capacidades y cualidades que debe cumplir el módulo, sería satisfactorio actualizar el diseño elaborado, generando nuevas versiones del mismo, en correspondencia con los cambios realizados.

- ✓ Durante la implementación del módulo sería recomendable realizar auditorías y revisiones para evaluar el grado de correspondencia que debe existir entre esta y el diseño propuesto en este trabajo.

- ✓ En el desarrollo de los módulos del proyecto, que actualmente no se les ha realizado el diseño, sería propicio emplear como guía, el estudio realizado durante la confección del presente trabajo de diploma, facilitando la elaboración y estandarización de la documentación de estos módulos.

Referencias Bibliográficas

1. **Contreras Días, Yimian de Liz y Rivero Amador, Soleydi.** *Diseño del Sistema de Gestión de Información del Centro de Estudios de Medio Ambiente y recursos Naturalez (CEMARNA).* Universidad de Pinar del Río : s.n., 2007. [Citado el: 5 de Mayo de 2008.]

2. **Lacoste Ricardo, Yoendris y Sánchez Luna, Jorge Daniel.** *LIMS de Calidad del Centro de Ingeniería Genética y Biotecnología: Análisis del Módulo de Microbiología.* Universidad de las Ciencias Informáticas : s.n., 2007. [Citado el: 29 de Mayo de 2008.]

3. **Rodríguez Perojo, Keily y Ronda León, Rodrigo.** El Web como sistema de información. [En línea] 2006. [Citado el: 29 de Mayo de 2008.] http://bvs.sld.cu/revistas/aci/vol14_1_06/aci08106.htm.

4. **Franco Noriega, Yaritza y López Corría, Yulieimis.** *LIMS de Calidad del Centro de Ingeniería Genética y Biotecnología: Análisis de los laboratorios Sistemas Críticos y Cromatografía-Electroforesis.* Universidad de las Ciencias Informáticas : s.n., 2007. [Citado el: 12 de Diciembre de 2007.]

5. **Rumbaugh, Jacobson y Booch.** *EL LENGUAJE UNIFICADO DE MODELADO. La referencia definitiva de UML escrita por sus autores.* [Citado el: 12 de Diciembre de 2007.]

6. **Larman, Craig.** *UML y PATRONES. Introducción al análisis y diseño orientado a objetos.* México : s.n., 1999.

7. Prácticas Ingeniería del Software 3ra. Una Herramienta CASE para ADOO: Visual Paradigm. Análisis y Diseño Orientado a Objetos. [En línea] [Citado el: 13 de Febrero de 2007.] http://alarcos.inf-cr.uclm.es/per/fgarcia/isoftware/doc/LabTr1_VP.pdf.

8. Teleconferencia: Fase de Elaboración. Análisis y Diseño. [En línea] [Citado el: 12 de Diciembre de 2007.] mms://ucimedia.uci.cu/teleclases/1er_Semestre/3er/Ingenieria_de_Software_I/conf5.

9. **Potencier, Fabien y Zaninotto, Francois.** *Symfony, la guía definitiva, Editorial.* s.l. : Apress (ISBN-13: 978-1590597866).

10. **Oktaba, Hanna.** Introducción a Patrones. [En línea] [Citado el: 24 de Abril de 2008.] http://www.mcc.unam.mx/~cursos/Algoritmos/javaDC99-2/patrones.html.

11. Paradigma MVC. [En línea] [Citado el: 27 de Abril de 2008.] http://aldeacafe.com.mx/programacion/patrones/mvc.htm. [Citado el: 24 de Abril de 2008.]

12.Patrón Decorator. [En línea] [Citado el: 27 de Abril de 2008.] Disponible en: http://agamenon.uniandes.edu.co/~pfiguero/soo/PatronesDiseno/Decorator/Decorator.htm.

13. Mapa de Navegacion. [En línea] [Citado el: 30 de Mayo de 2008.] http://www.arquitecturadeinformacion.cl/como/mapa.html .

Bibliografías

1. CENTRO DE INGENIERÍA GENÉTICA Y BIOTECNOLOGÍA. *Dirección de Calidad.* [En línea] 2003. http://www.cigb.edu.cu/pages/calidad.htm.
2. CENTRO DE INGENIERÍA GENÉTICA Y BIOTECNOLOGÍA. *Departamento de Control de la Calidad.* [En línea] 2003. http://www.cigb.edu.cu/pages/ccalidad.htm.
3. **Contreras Días, Yimian de Liz y Rivero Amador, Soleydi.** *Diseño del Sistema de Gestión de Información del Centro de Estudios de Medio Ambiente y recursos Naturalez (CEMARNA).* Universidad de Pinar del Río : s.n., 2007.
4. Entorno Virtual de Aprendizaje. *Teleconferencias de Ingeniería de Software.* [En línea] teleformacion.uci.cu.
5. **Franco Noriega, Yaritza y López Corría, Yulieimis.** *LIMS de Calidad del Centro de Ingeniería Genética y Biotecnología: Análisis de los laboratorios Sistemas Críticos y Cromatografía-Electroforesis.* Universidad de las Ciencias Informáticas : s.n., 2007.
6. **Jacobson, Ivar y Booch, Grady.** *El Proceso Unificado de Desarrollo de Software.* 1999.
7. **Lacoste Ricardo, Yoendris y Sánchez Luna, Jorge Daniel.** *LIMS de Calidad del Centro de Ingeniería Genética y Biotecnología: Análisis del Módulo de Microbiología.* Universidad de las Ciencias Informáticas : s.n., 2007.
8. **Larman, Craig.** *UML y PATRONES. Introducción al análisis y diseño orientado a objetos.* México : s.n., 1999.
9. Mapa de Navegacion. [En línea] [Citado el: 30 de 05 de 2008.] http://www.arquitecturadeinformacion.cl/como/mapa.html.
10. **Oktaba, Hanna.** Introducción a Patrones. [En línea] [Citado el: 24 de Abril de 2008.] http://www.mcc.unam.mx/~cursos/Algoritmos/javaDC99-2/patrones.html.
11. Paradigma MVC. [En línea] [Citado el: 27 de Abril de 2008.] http://aldeacafe.com.mx/programacion/patrones/mvc.htm.

12. Patrón Decorator. [En línea] [Citado el: 27 de 04 de 2008.] Disponible en: http://agamenon.uniandes.edu.co/~pfiguero/soo/PatronesDiseno/Decorator/Decorator.htm.

13. Prácticas Ingeniería del Software 3ra. Una Herramienta CASE para ADOO: Visual Paradigm. Análisis y Diseño Orientado a Objetos. [En línea] [Citado el: 13 de Febrero de 2007.] http://alarcos.inf-cr.uclm.es/per/fgarcia/isoftware/doc/LabTr1_VP.pdf.

14. **Potencier, Fabien y Zaninotto, Francois.** *Symfony, la guía definitiva, Editorial.* s.l. : Apress (ISBN-13: 978-1590597866).

15. Rational Unified Process. . [En línea] Disponible en: https://pid.dsic.upv.es/C1/Material/Documentos%20Disponibles/Introducci%C3%B3n%20a%20RUP.doc.

16. **Rodríguez Perojo, Keily y Ronda León, Rodrigo.** El Web como sistema de información. [En línea] 2006. [Citado el: 29 de Mayo de 2008.] http://bvs.sld.cu/revistas/aci/vol14_1_06/aci08106.htm.

17. **Rumbaugh, Jacobson y Booch.** *EL LENGUAJE UNIFICADO DE MODELADO. La referencia definitiva de UML escrita por sus autores.*

18. **Sarmiento, Alieski y Cutiño, Elián.** *LIMS de Calidad del Centro de Ingeniería Genética y Biotecnología: Análisis del Grupo de Recepción de Muestras y Manipulación de Expedientes.* Instituto Superior Politécnico “José Antonio Echeverria" : s.n., 2006.

19. Teleconferencia: Fase de Elaboración. Análisis y Diseño. [En línea] [Citado el: 12 de Diciembre de 2007.] mms://ucimedia.uci.cu/teleclases/1er_Semestre/3er/Ingenieria_de_Software_I/conf5.

20. Visual Paradigm for UML. [En línea] http://www.visual-paradigm.com.

Anexos

Anexo 1: Estructura jerárquica del área de Calidad del CIGB.

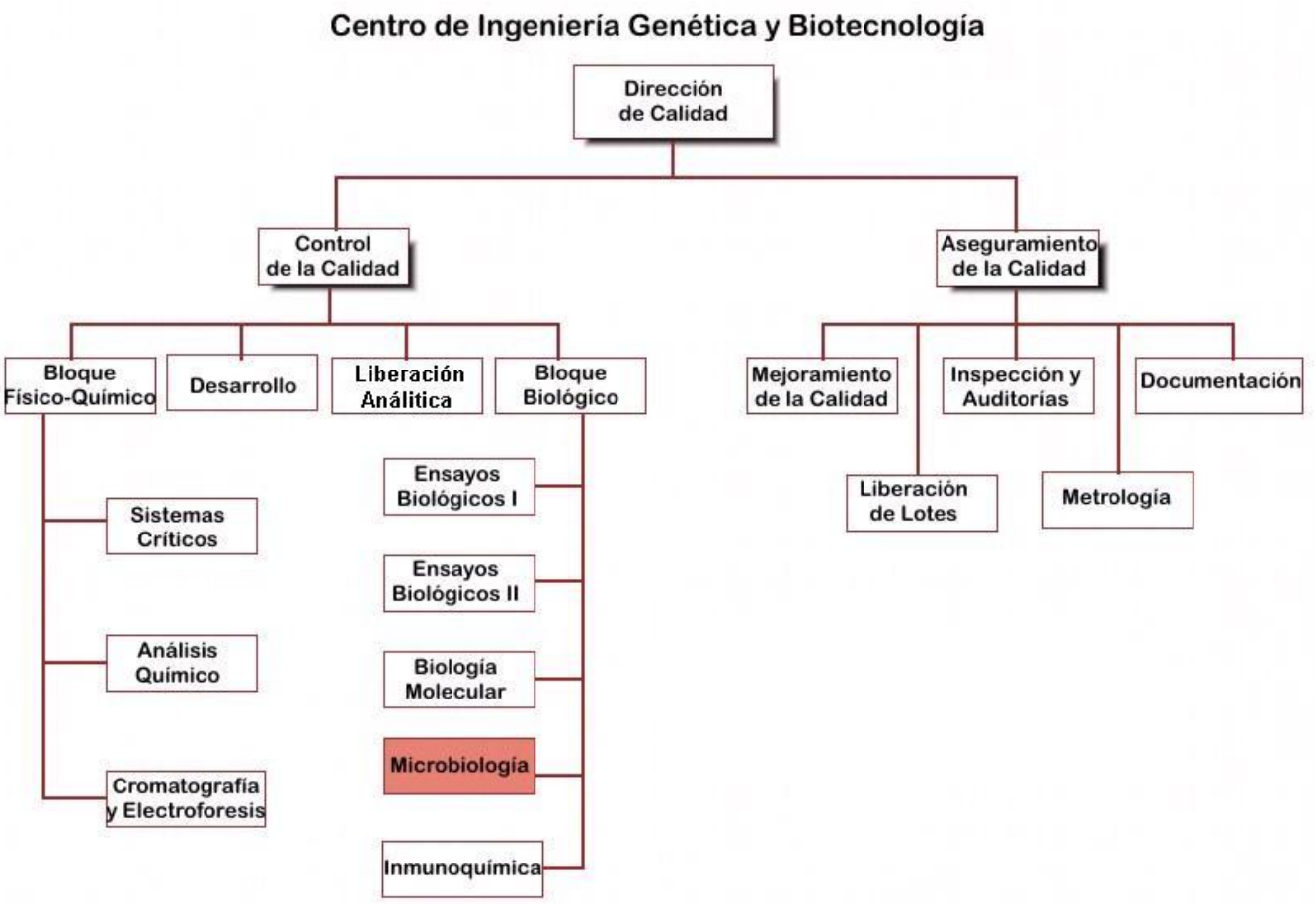

Fig 49: Estructura jerárquica del área de Calidad del CIGB.

Anexo 2: Lenguaje Unificado de Modelado (UML)

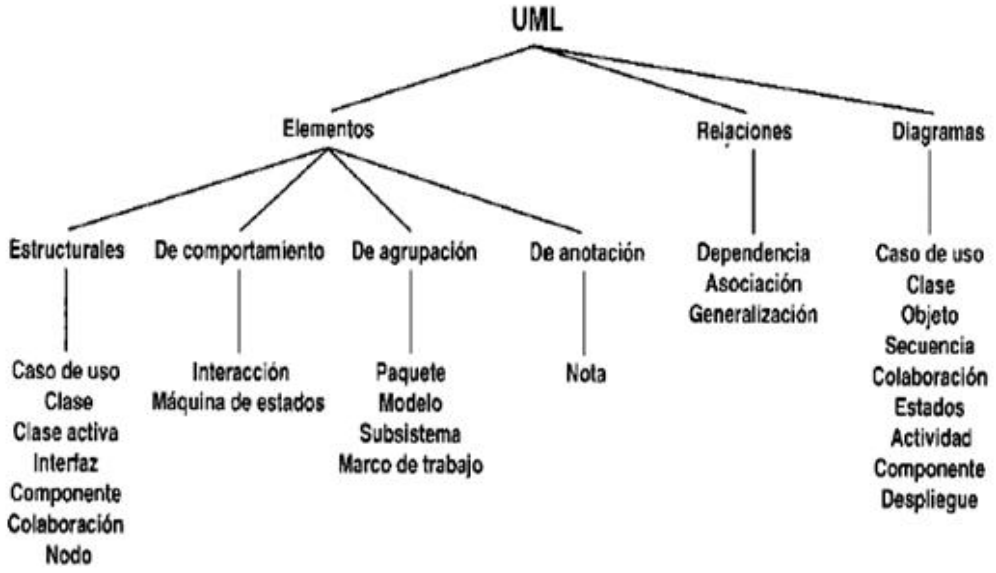

Fig 50: Lenguaje Unificado de Modelado (UML)

Anexo 3: Funcionamiento del patrón MVC

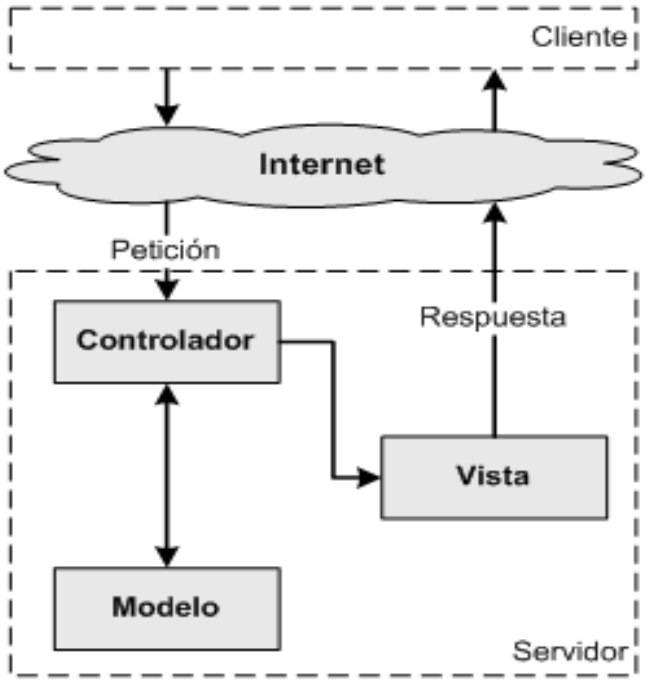

Fig 51: Funcionamiento del patrón MVC

Anexo 4: Estructura del patrón Decorator

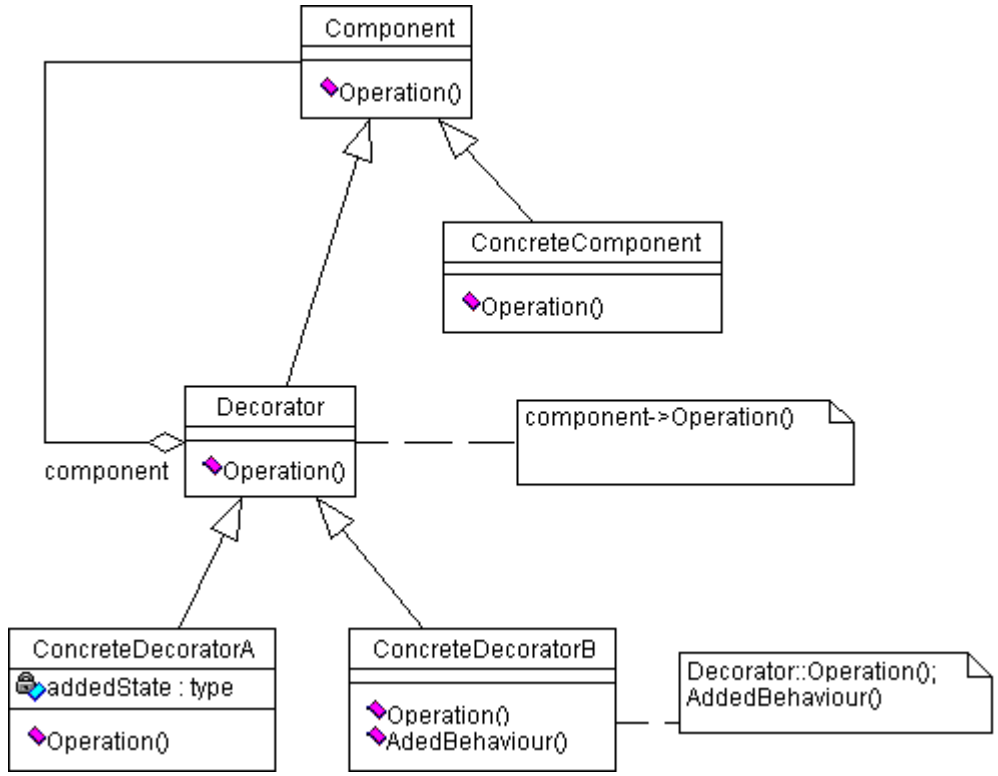

Fig 52: Estructura del patrón Decorator

Anexo 5: Plantilla decorada con un layout.

Fig 53: Plantilla decorada con un layout.

Glosario de Términos

Artefacto: Pieza de información utilizada o producida por un proceso de desarrollo de software, como un documento externo o el producto de un trabajo. Un artefacto puede ser un modelo, una descripción o un software.

Aseguramiento de la Calidad: Conjunto de acciones planificadas y sistemáticas que son necesarias para proporcionar la confianza de que un producto o servicio satisface los requisitos de calidad preestablecidos.

Caso de uso: Especificación de las secuencias de acciones, incluyendo secuencias variantes y una descripción de un conjunto de secuencias de acciones, incluyendo variaciones, que un sistema lleva a cabo y que conduce a un resultado observable de interés para un actor determinado.

Control de la Calidad: Técnicas y actividades de carácter operativo utilizadas para satisfacer los requisitos de calidad.

Desviación: Alteración no prevista, resultado de variaciones accidentales negligentes o aleatorias que afecta o puede afectar potencialmente la calidad de un producto o proceso.

GRASP: es un acrónimo que significa General Responsibility Asignment Software Patterns (patrones generales de software para asignar responsabilidades).

Metodología: Un sistema de principios y normas generales de organización y estructuración teórico-práctica de actividades.

SIC: Sistema de Información de Calidad.

Validación: Acción documentada que demuestra, de acuerdo con los principios de las Buenas Prácticas de Fabricación, que cualquier procedimiento, proceso, equipo, material, actividad o sistema realmente brinda los resultados esperados.