SISTEMA DE EVALUACIÓN VIRTUAL

ELIANA MARGARITA LÓPEZ LENGUA

DIRECTOR:

MOISÉS QUINTANA

MSC. CIENCIAS COMPUTACIONALES

UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR

PROGRAMA DE INGENIERÍA DE SISTEMAS

FACULTAD DE INGENIERÍA

CARTAGENA DE INDIAS

2011

SISTEMA DE EVALUACIÓN VIRTUAL

Presentado por:

ELIANA MARGARITA LÓPEZ LENGUA
Estudiante de Ingeniería de Sistemas
Universidad Tecnológica de Bolívar

Revisado por:

MOISÉS QUINTANA
Director de Monografía

Tabla de Contenido

Tabla de Ilustraciones

Objetivos

Explicar de forma clara y concisa en que consiste el Sistema de evaluación virtual, su arquitectura y requerimientos.

Objetivos Específicos

- Explicar en qué consiste el sistema de evaluacion virtual.
- Ilustrar en que consiste la arquitectura usada en el sistema.

Introducción

La tenología JavaServer Faces (JSF) es un nuevo marco de trabajo para interfaces de usuario para aplicación J2EE. Por diseño, es particularmente útil con aplicaciones basadas en la arquitectura MVC (*Model-View-Controller*). Numérosos artículos han presentado ya JSF. Sin embargo, la mayoría toma una apróximación teórica que no cumple los retos del desarrollo empresarial del mundo real. Quedan muchos problemas por resolver. Por ejemplo, ¿dónde entra JSF en la arquiectura general MVC?, ¿Cómo se integra JSF con otros marcos de trabajo?, ¿Existe la lógica de negocio en los beans que hay tras JSF? Y principalmente, ¿cómo se puede construir una aplicación Web del mundo real utilizando JSF?

La primera fase en el diseño de una aplicación Web es descubrir los requerimientos funcionales del sistema. Se utiliza el análisis de los casos de utilización para acceder a los requerimientos funcionales de la aplicación , mediante estos se identifica los actores en un sistema y las operaciones que podrían realizar.

La siguiente fase es el diseño de la arquitectura de alto nivel. Esto implica subdividir la aplicación en componentes funcionales y particionar estos componentes en capas. El diseño de la arquitectura de alto nivel es neutral a las tecnologías utilizadas.

Sistema de Evaluación Virtual

El sistema de evaluación virtual es un sistema pensado para la realización de exámenes en linea, en cual se evalúan las capacidades de los estudiantes en temas respectivos mediante preguntas de respuesta múltiple, única respuesta y verdadero y falso.

El sistema permite crear exámenes aleatorios cada vez que un estudiante toma un examen, esto es posible dado que al crear el examen el profesor elige el porcentaje de preguntas de una tema en especifico.

Ilustración 1 - Pantalla de inicio de sesión

Requerimientos Funcionales

- Gestión de preguntas.
- Gestión de temas.
- Generación de exámenes.
- Consulta de resultados.
- Generación de reportes

Requerimientos No Funcionales

- Disponibilidad.
- Eficiencia.
- Estabilidad.
- Seguridad.

Tecnologías Usadas

JSF

JavaServer Faces (**JSF**) es una tecnología y framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE. **JSF** usa JavaServer Pages (JSP) como la tecnología que permite hacer el despliegue de las páginas, pero también se puede acomodar a otras tecnologías como XUL.

JSF incluye:

- Un conjunto de APIs para representar componentes de una interfaz de usuario y administrar su estado, manejar eventos, validar entrada, definir un esquema de navegación de las páginas y dar soporte para internacionalización y accesibilidad.
- Un conjunto por defecto de componentes para la interfaz de usuario.
- Dos bibliotecas de etiquetas personalizadas para JavaServer Pages que permiten expresar una interfaz JavaServer Faces dentro de una página JSP.
- Un modelo de eventos en el lado del servidor.
- Administración de estados.
- Beans administrados.

El objectivo de la aplicaciones creadas en JSF es:

- Definir un conjunto simple de clases base de Java para componentes de la interfaz de usuario, estado de los componentes y eventos de entrada. Estas clases tratarán los aspectos del ciclo de vida de la interfaz de usuario, controlando el estado de un componente durante el ciclo de vida de su página.
- Proporcionar un conjunto de componentes para la interfaz de usuario, incluyendo los elementos estándares de HTML para representar un formulario. Estos componentes se obtendrán de un conjunto básico de clases base que se pueden utilizar para definir componentes nuevos.

- Proporcionar un modelo de JavaBeans para enviar eventos desde los controles de la interfaz de usuario del lado del cliente a la aplicación del servidor.
- Definir APIs para la validación de entrada, incluyendo soporte para la validación en el lado del cliente.
- Especificar un modelo para la internacionalización y localización de la interfaz de usuario.
- Automatizar la generación de salidas apropiadas para el objetivo del cliente, teniendo en cuenta todos los datos de configuración disponibles del cliente, como versión del navegador.

Hibernate

Hibernate es una herramienta de mapeo objeto-relacional (ORM) para el lenguaje Java, que proporciona un marco para la asignación de un modelo de dominio orientado a objetos para bases de datos relacionales tradicionales. Hibernate resuelven objeto-relacional de los problemas de falta de concordancia mediante la sustitución de directos relacionados con la persistencia de accesos de bases de datos con funciones de alto nivel de objeto de manipulación.

Características:

- Mapea de clases Java a tablas de bases de datos (y de tipos de datos Java a tipos de datos SQL).
- Proporciona persistencia transparente de (POJO). El único requisito estricto para una clase persistente es un constructor sin argumentos, no necesariamente público.
- Ofrece consulta y recuperación de datos.
- Genera las sentencias SQL y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todos los motores de bases de datos con un ligero incremento en el tiempo de ejecución
- Está diseñado para ser flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente. También tiene la funcionalidad de crear la base de datos a partir de la información disponible.
- Ofrece también un lenguaje de consulta de datos llamado **HQL** (*Hibernate Query Language*), al mismo tiempo que una API para construir las consultas programáticamente (conocida como "*criteria*").

ICEfaces

ICEfaces es un framework de código abierto de Ajax que permite a los desarrolladores de aplicaciones Java EE, crear y desplegar aplicaciones de Internet enriquecidas (RIA) utilizando el lenguaje Java.

Ventajas:

- Aprovecha al completo los estándares basados en el ecossistema de herramientas de Java EE y sus entornos de ejecución.
- Las aplicaciones son desarrolladas en Java puro y en un modelo de cliente ligero.
- No necesitan plugins de navegador o applets para ser vistas.
- Las aplicaciones estan basadas en JavaServer Faces (JSF), así que permite el desarrollo de aplicaciones Java EE con la posibilidad de utilizar de forma fácil desarrollos basados en JavaScripts.
- Aisla completamente al desarrollador de AJAX.
- No hacen falta etiquetas especiales: se ponen los controles en la pantalla e ICEFaces se encarga de enviar entre cliente y servidor sólo la información necesaria.

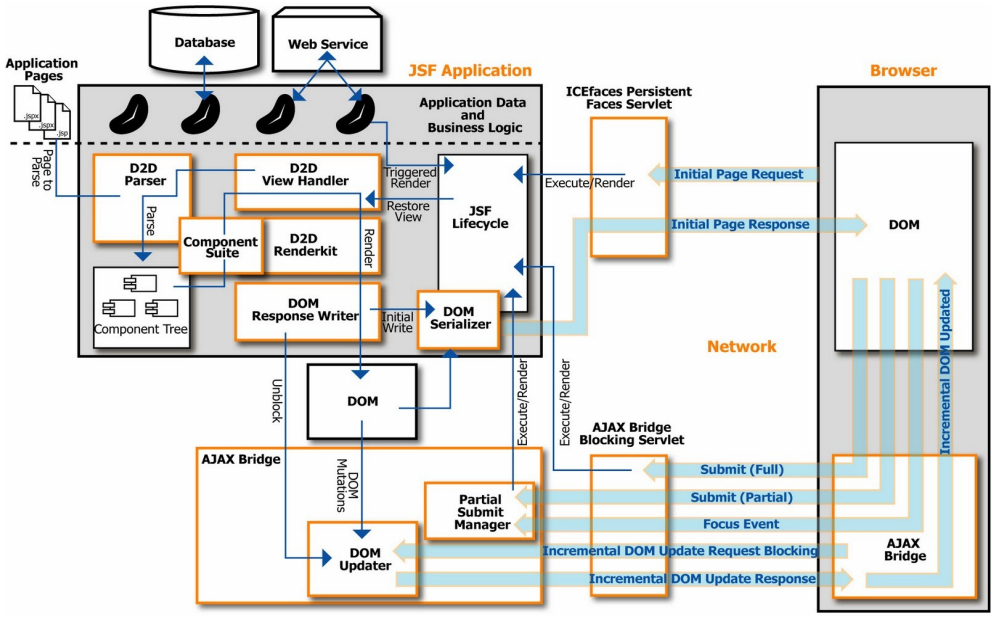

Ilustración 2 – Arquitectura de una aplicación en JSF integrada con ICEFace.

Los principales elementos de la arquitectura ICEfaces incluyen:

- Persistent Faces Servlet: Las URLs con extensión ".iface" son mapeadas por el servlet 'Persistent Faces Servlet'. Cuando se realiza una petición de la página inicial en la aplicación, este servlet se hace responsable de la ejecución del ciclo de vida JSF para petición asociada.
- Blocking Servlet: Se encarga de la gestión de todos las peticiones de bloqueo y no-bloqueo después de las primeras páginas.
- D2D ViewHandler: Se encarga de establecer el Direct-to-DOM, incluyendo la inicialización de la 'DOM Respuesta Writer'. El ViewHandler también invoca al Parser para analizar el árbol de componentes JSF en la página inicial.
- Parseador D2D: Responsable del montaje de un componente de documentos JSP. El Parser ejecuta la etiqueta de JSP de procesamiento del ciclo de vida con el fin de crear el árbol, pero lo hace sólo una vez para cada página. La compilación del estándar JSP y el proceso de análisis no es compatible con ICEfaces.
- DOM Response Writer: Se encarga de la escritura en el DOM. También inicia la serialización DOM para la primera prestación, y desbloquea el DOM Updater para actualizaciones incrementales.
- DOM Serializer: Responsable de la serialización del DOM de la página inicial.
- DOM Updater: Se encarga de conjuntar las de las 'DOM mutations' en una única actualización DOM.
- Component Suite: Ofrece un conjunto de componentes 'rich JSF' con influencia AJAX y características del puente, proporcionando los elementos básicos para aplicaciones ICEfaces.
- Client-side AJAX Bridge: Responsable de la actualización DOM en curso generada por la solicitud y la respuesta del proceso. También es el encargado de centrar la gestión y de presentar el proceso.

Spring

El Spring Framework es un framework de código abiertocontenedor liviano basado en la técnica Inversión de Control (IoC) y una implementación de desarrollo según el paradigma de Orientación a Aspectos (AOP).

A pesar de que Spring Framework no obliga a usar un modelo de programación en particular, se ha popularizado en la comunidad de programadores en Java al considerársele una alternativa y sustituto del modelo de Enterprise JavaBean. Por su diseño el framework ofrece mucha libertad a los desarrolladores en Java y soluciones muy bien documentadas y fáciles de usar para las prácticas comunes en la industria.

Arquitectura:

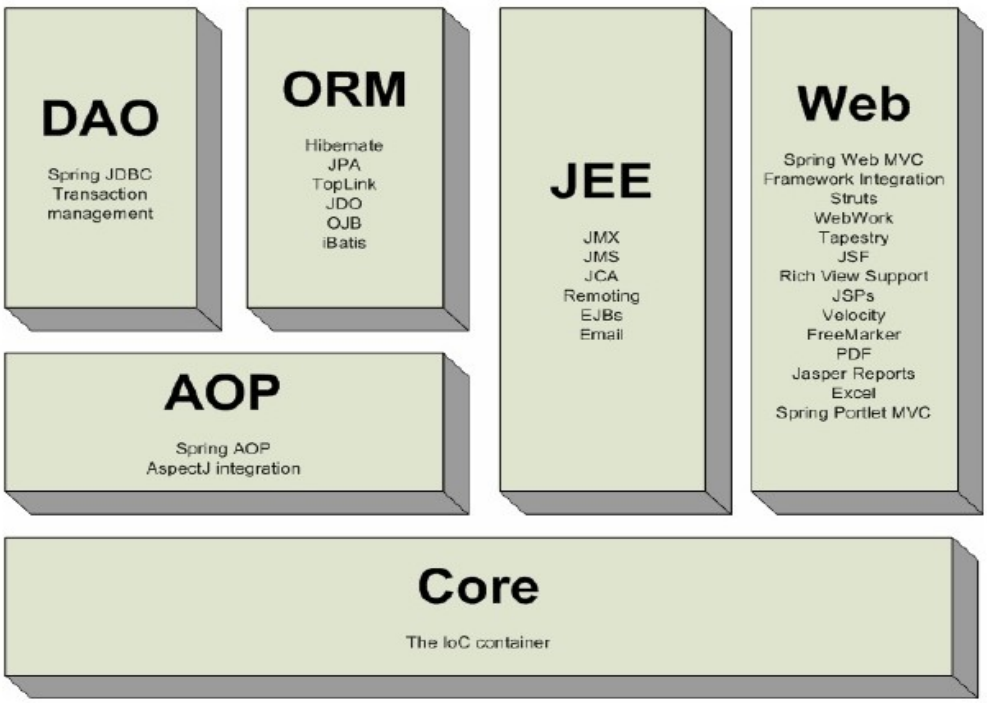

Ilustración 3 – Arquitectura de Spring.

MySQL

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario.

Inicialmente, MySQL carecía de elementos considerados esenciales en las bases de datos relacionales, tales como integridad referencial y transacciones. A pesar de ello, atrajo a los desarrolladores de páginas web con contenido dinámico, justamente por su simplicidad.

Poco a poco los elementos de los que carecía MySQL están siendo incorporados tanto por desarrollos internos, como por desarrolladores de software libre. Entre las características disponibles en las últimas versiones se puede destacar:

- Amplio subconjunto del lenguaje SQL. Algunas extensiones son incluidas igualmente.
- Disponibilidad en gran cantidad de plataformas y sistemas.
- Diferentes opciones de almacenamiento según si se desea velocidad en las operaciones o el mayor número de operaciones disponibles.
- Transacciones y claves foráneas.
- Conectividad segura.
- Replicación.
- Búsqueda e indexación de campos de texto.

Las siguientes características son implementadas únicamente por MySQL:

- Múltiples motores de almacenamiento (MyISAM, Merge, InnoDB, BDB, Memory/heap, MySQL Cluster, Federated, Archive, CSV, Blackhole y Example en 5.x), permitiendo al usuario escoger la que sea más adecuada para cada tabla de la base de datos.
- Agrupación de transacciones, reuniendo múltiples transacciones de varias conexiones para incrementar el número de transacciones por segundo.

Arquitectura del sistema

Una arquitectura multicapa particiona todo el sistema en distintas unidades funcionales: cliente, presentación, lógica-de-negocio, integración, y sistema de información empresarial (EIS). Esto asegura una división clara de responsabilidades y hace que el sistema sea más mantenible y extensible. Los sistemas con tres o más capas se han probado como más escalables y flexibles que un sistema cliente-servidor, en el que no existe la capa central de lógica-de-negocios.

La capa del cliente es donde se consumen y presentan los modelos de datos. Para una aplicación Web, la capa cliente normalmente es un navegador web. Los clientes pequeños basados-en-navegador no contienen lógica de presentación; se trata en la capa de presentación.

La capa de presentación expone los servicios de la capa de lógica-de-negocio a los usuarios. Sabe cómo procesar una petición de cliente, cómo interactúar con la capa de lógica-de-negocio, y cómo seleccionar la siguiente vista a mostrar.

La capa de la lógica-de-negocio contiene los objetos y servicios de negocio de la aplicación. Recibe peticiones de la capa de presentación, procesa la lógica de negocio basada en las peticiones, y media en los accesos a los recursos de la capa EIS. Los componenes de la capa de lógica-de-negocio se benefician de la mayoría de lo servicios a nivel de sistema como el control de seguridad, de transaciones y de recursos.

La capa de integración es el puente entre la capa de lógica-de-negocio y la capa EIS. Encapsula la lógica para interactúar con la capa EIS. Algunas veces a la combinación de las capas de integración y de lógica-de-negocio se le conoce como **capa central**.

Los datos de la aplicación persisten en la capa EIs. Contiene bases de datos relacionales, bases de datos orientadas a objetos, y sistemas antiguos.

La aplicación utiliza una arquitectura multi-capa no-distribuida. La figura 1 muestra el particionamiento de las capas de la aplicación y las tecnologías elegidas para cada capa. También sierve como diagrama de ejemplo de despliegue de la aplicación. Para una arquitectura colocada, las capas de presentación, de lógica-de-negocio y de integración están físicamente localizadas en el mismo contenedor Web. Interfaces bien-definidos aislan las responsabilidades de cada capa. La arquitectura colocada hace que la aplicación sea más simple y escalable.

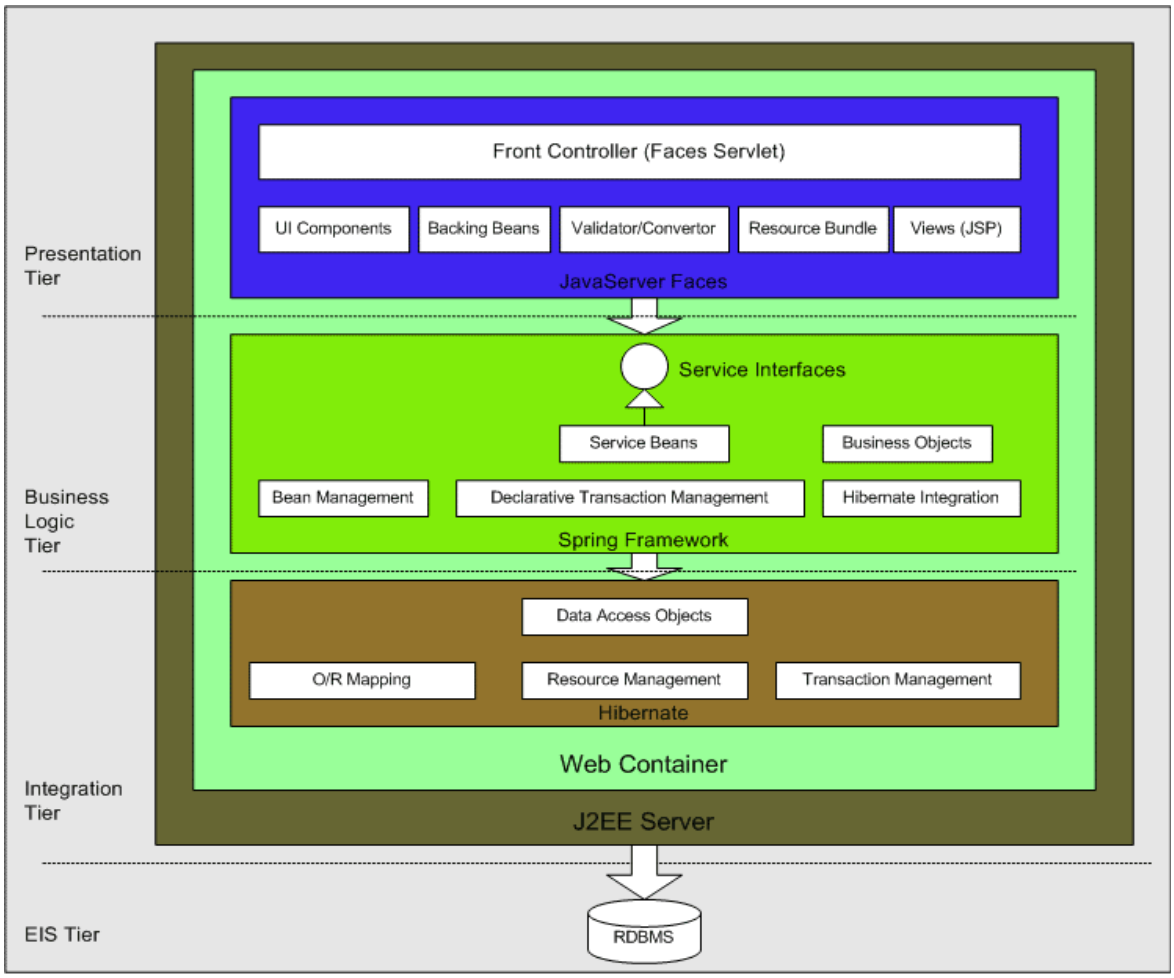

Ilustración 4 - Diseño de la arquitectura de alto nivel.

Para la capa de presentación, la experiencia muestra que la mejor práctica es elegir un marco de trabajo de aplicaciones Web existente y probado en vez de diseñar y construir un marco de trabajo personalizado. Hay varios marcos de trabajo de este tipo entre los que elegir: Struts, WebWork, y JSF. Nosotros utilizaremos JSF para JCatalog.

Se pueden utilizar EJB (*Enterprise JavaBeans*) o POJO (*plain old Java objects*) para construir la capa de lógica-de-negocio. EJB con sus interfaces remotos es una mejor elección si la aplicación es distribuida. Como JCatalog es una aplicación web típica que no requiere acceso remoto, se utilizará POJO con la ayuda del marco de trabajo Spring, para esta capa.

La capa de integración maneja la persistencia de los datos con la base de datos relacional. Se pueden utilizar diferentes aproximaciones para implementar la capa de integración:

- **JDBC puro (*Java DataBase Connectivity)*:** Es la aproximación más flexible; sin embargo es difícil trabajar con JDBC de bajo nivel, y un mal código JDBC no tiene un rendimiento muy bueno.
- **Beans de entidad:** Un Bean de entidad con persistencia manejada por el contenedor (CMP) es una forma costosa de aislar el código de acceso-a-datos y manejar el mapeo O/R (objecto-relacional) para persistencia de datos. Es una aproximación centrada en el servidor de aplicaciones. Un bean de entidad no ata la aplicación a un tipo particular de base de datos, pero si ata la aplicación al contenedor EJB.
- **Marco de Trabajo de Mapeo O/R:** Un marco de trabajo de mapeo O/R toma una aproximación centrada en el objeto para implementar la persistencia de datos. Una aplicación de este tipo es fácil de desarrollar y altamente portable. Existen varios marcos de trabajo bajo este dominio: JDO (Java Data Objects), Hibernate, TopLink, y CocoBase son unos pocos ejemplos. En la aplicación de ejemplo se utiliza Hibernate.

La capa de presentación recoge la entrada del usuario, presenta los datos, controla la navegación por las página y delega la entrada del usuario a la capa de la lógica-de-negocio.

La capa de presentación también puede validar la entrada del usuario y mantener el estado de sesión de la aplicación. En las siguientes secciones, discutiremos las consideraciones de diseño y los patrones de la capa de presentación, y la razón porque la que he elegido JSF para implementar JCatalog.

Model-View-Controller (MVC)

MVC es el patrón de diseño arquitectural recomendado para aplicaciones interactivas Java. MVC separa los conceptos de diseño, y por lo tanto decrementa la duplicación de código, el centralizamiento del control y hace que la aplicación sea más extensible. MVC también ayuda a los desarrolladores con diferentes habiliades a enfocarse en sus habilidades principales y a colaborar a través de interfaces claramente definidos. MVC es el patrón de diseño arquitectural para la capa de presentación.

JavaServer Faces

JSF es un marco de trabajo de componentes de interface de usuario del lado del servidor para aplicaciones Web basadas en Java. JSF contiene un API para representar componentes UI y manejar sus estados, manejar sus eventos, ña validación del lado del servidor, y la conversión de datos, definir la navegación entre páginas, soportar internacionalización y accesibilidad; y proporcionar extensibilidad para todas estas características. Tambiéne contien dos librerías de etiquetas JSP (JavaServer Pages) personalizadas para expresar componentes UI dentro de una página JSP y para conectar componentes a objetos del lado del servidor.

JSF y MVC

JSF encaja bien en la arquitectura de la capa de presentación basada en MVC. Ofrece una clara separación entre el comportamiento y la presentación. Une los familiares componentes UI con los conceptos de la capa-Web sin limitarnos a una tecnología de script o lenguaje de marcas particular.

Los beans que hay tras JSF son la capa de modelo (en una sección posterior hablaremos más de estos beans). También contienen acciones, que son una extensión de la capa del controlador y delegan las peticiones del usuario a la capa de la lógica-de-negocio. Observe, que desde la prespectiva de la arquitectura general de la aplicación, también se puede referir a la capa de lógica-de-negocio como la capa del modelo. Las páginas JSP con

etiquetas JSF personalizadas son la capa de la vista. El Servlet Faces proporciona la funcionalidad del controlador.

JSF no es sólo otro marco de trabajo Web. Las siguientes características diferencian a JSF de otros marcos de trabajo:

- **Desarrollo de una Aplicación Web Orientada a Objetos al Estilo Swing:** El modelo de componentes UI con estado del lado del servidor con oyentes/manejadores de eventos inicia el desarrollo de aplicaciones Web orientadas a objetos.
- **Control de Beans-de-Respaldo:** Los beans de respaldo son componentes JavaBeans asociados con componentes UI utilizados en la página. El control de beans-de-respaldo separa la definición de los objetos componentes del UI de los objetos que realizan el procesamiento específico de la aplicación y que además contienen los datos. La implementación de JSF almacena y maneja estos ejemplares de beans de respaldo en el ámbito apropiado.
- **Modelo de componentes UI extensible:** Los componentes UI de JSF son elementos configurables, reutilizables que componen los intefaces de usuario de aplicaciones JSF. Se puede extender un componentes UI estándar y desarrollar componentes más complejos, como barras de menú y árboles.
- **Modelo de Renderizado Flexible:** Un renderizador separa la vista y la funcionalidad de los componentes UI. Se pueden crear y utilizar varios renderizadores para definir diferentes apariencias del mismo componente para el mismo o diferentes clientes.
- **Modelo de Conversión y Validación Extensible:** Basados en los convertidores y validadores estándar, se pueden desarrollar convertidores y validadores personalizados, que proporcionan un mejor modelo de protección.

A pesar de su potencia, JSF no esta madura aún. Los componentes, convertidores y validadores que vienen con JSF son básicos. Y el modelo de validación por-componente no puede manejar validaciones muchos-a-muchos entre componentes y validadores. Además,

las etiquetas personalizadas de JSF no se pueden integrar con JSTL (*JSP Standard Tag Library*) simultáneamente.

La Capa de Lógica-de-Negocio y el Marco de Trabajo Spring

Los objetos y servicios de negocio existen en la capa de lógica-de-negocio. Un objeto de negocio no sólo contiene datos, también la lógica asociada con ese objeto específico. En la aplicación de ejemplo se han identificado tres objetos de negocio: Product, Category, y User.

Los servicios de negocio interactúan con objetos de negocio y proporcionan una lógica de negocio de más alto nivel. Se debería definir una capa de interface de negocio formal, que contenga los interfaces de servicio que el cliente utilizará directamente. POJO, con la ayuda del marco de trabajo Spring, implementará la capa de lógica-de-negocio de la aplicación JCatalog. Hay dos servicios de negocio: CatalogService contiene la lógica de negocio relacionada con el manejo del catálogo, y UserService contiene la lógica de manejo del usuario.

Spring está basado en el concepto de inversión de control (IoC). Entre las características de Spring utilizadas en la aplicación de ejemplo se incluyen:

- **Manejo de Beans con contexto de aplicación:** Spring puede organizar de forma efectiva nuestros objetos de la capa central y manejar las conexiones por nosotros. Spring puede eliminar la proliferación de solitarios y facilita unas buenas prácticas de programación orientada a objetos, por ejemplo utilizando interfaces.
- **Manejo de Transaciones Declarativo:** Spring utiliza AOP (*aspect-oriented programming*) para ofrecer manejo de transaciones declarativo sin utilizar un contenedor EJB. De esta forma, el control de transaciones se puede aplicar a cualquier POJO. El control de transaciones de Spring no está atado a JTA (*Java Transaction API*) y puede funcionar con diferentes estrategias de transación. En la aplicación de ejemplo se utiliza el manejo de transación declarativo con Hibernate.
- **Árbol de Excepciones de Acceso a Datos:** Spring proporciona un magnífico árbol de excepciones en lugar de SQLException. Para poder utilizar este árbol de

excepciones, se debe definir un traductor de excepciones de acceso a datos dentro del fichero de configuración de Spring:

```
<bean id="jdbcExceptionTranslator"  class=
"org.springframework.jdbc.support.SQLErrorCodeSQLExceptionTranslator">

    <property name="dataSource">

        <ref bean="dataSource"/>

    </property>

</bean>
```

En la aplicación de ejemplo, si se inserta un producto nuevo con el ID duplicado, se lanza una DataIntegrityViolationException. La excepción es capturada y relanzada como una DuplicateProductIdException. De esta forma, la DuplicateProductIdException se puede manejar de forma diferente a cualquier otra excepción de acceso a datos.

- **Integración con Hibernate:** Spring no nos fuerza a utilizar su potente característica de abstracción JDBC. Se integra bien con marcos de trabajo de mapeo O/R, especialmente con Hibernate. Spring ofrece un manejo seguro y eficiente de sesiones Hibernate, maneja la configuración de la SessionFactorie de Hibernate y las fuentes de datos JDBC en el contexto de la aplicación, y hace que la aplicación sea más fácil de testear.

La Capa de Integración e Hibernate

Hibernate es un marco de trabajo de mapeo O/R Open Source que evita la necesidad de utilizar el API JDBC. Hibernate soporta la mayoría de los sistemas de bases de datos SQL. El *Hibernate Query Language*, diseñado como una extensión mínima, orientada a objetos, de SQL, proporciona un puente elegante entre los mundos objeto y relacional. Hibernate ofrece facilidades para recuperación y actualización de datos, control de transaciones, repositorios de conexiones a bases de datos, consultas programáticas y declarativas, y un control de relaciones de entidades declarativas.

Hibernate es menos invasivo que otros marcos de trabajo de mapeo O/R. Se utilizan Reflection y la generación de *bytecodes* en tiempo de ejecución, y la generación del SQL ocurre en el momento de la arrancada. Esto nos permite desarrollar objetos persistentes siguiendo el lenguaje común de Java: incluyendo asociación, herencia, polimorfismo, composición y el marco de trabajo Collections de Java. Los objetos de negocio de la aplicación de ejemplo son POJO y no necesitan implementar ningún interface específico de Hibernate.

Data Access Object (DAO)

En JCatalog se utiliza el patrón DAO. Este patrón abstrae y encapsula todos los accesos a la fuente de datos. La aplicación tiene dos interfaces DAO: CatalogDao y UserDao. Sus clases de implementación, HibernateCatalogDaoImpl y HibernateUserDaoImpl contienen lógica específica de Hibernate para manejar los datos persistentes.

Implementación

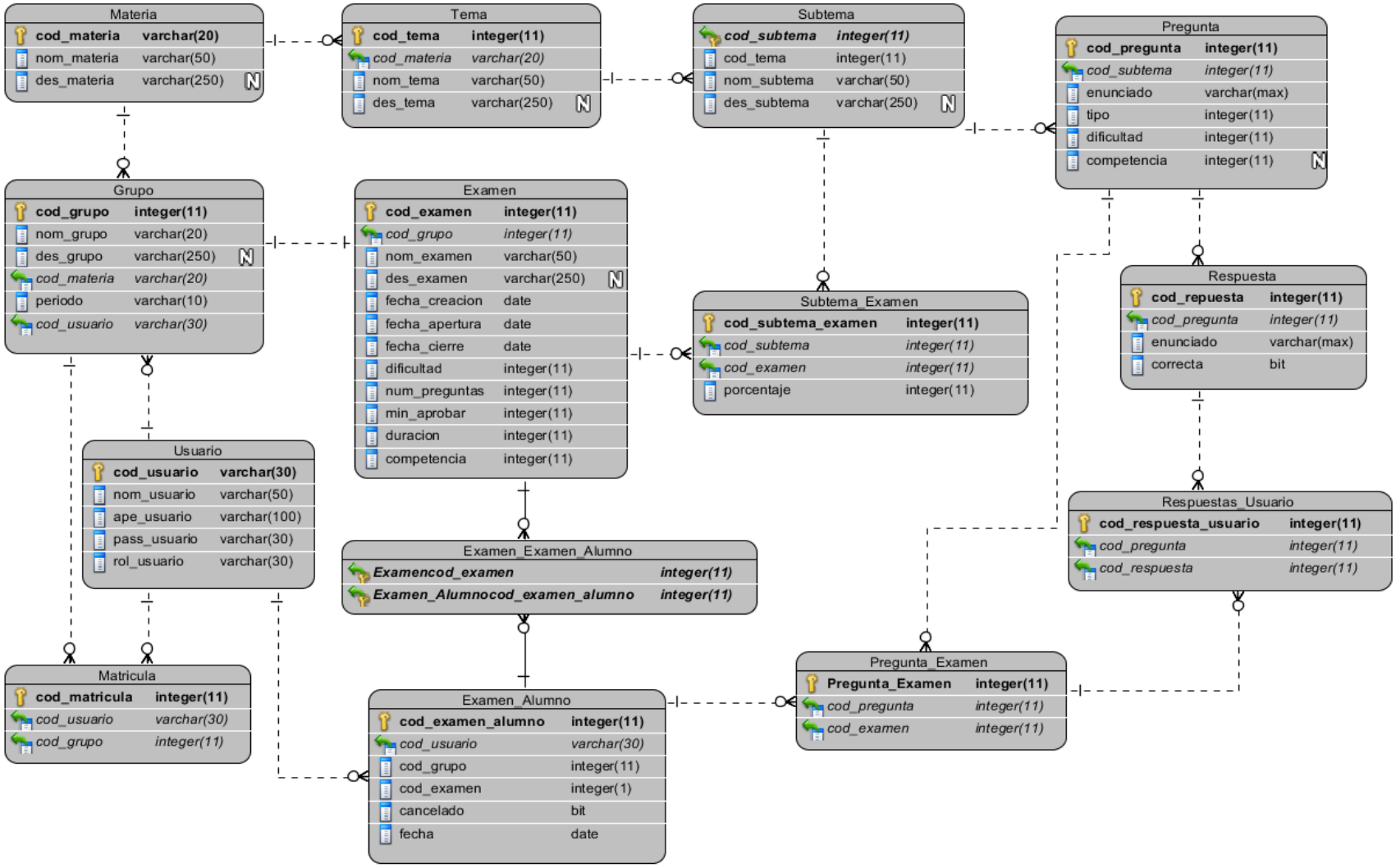

Ilustración 5 – Diseño de la base de datos.

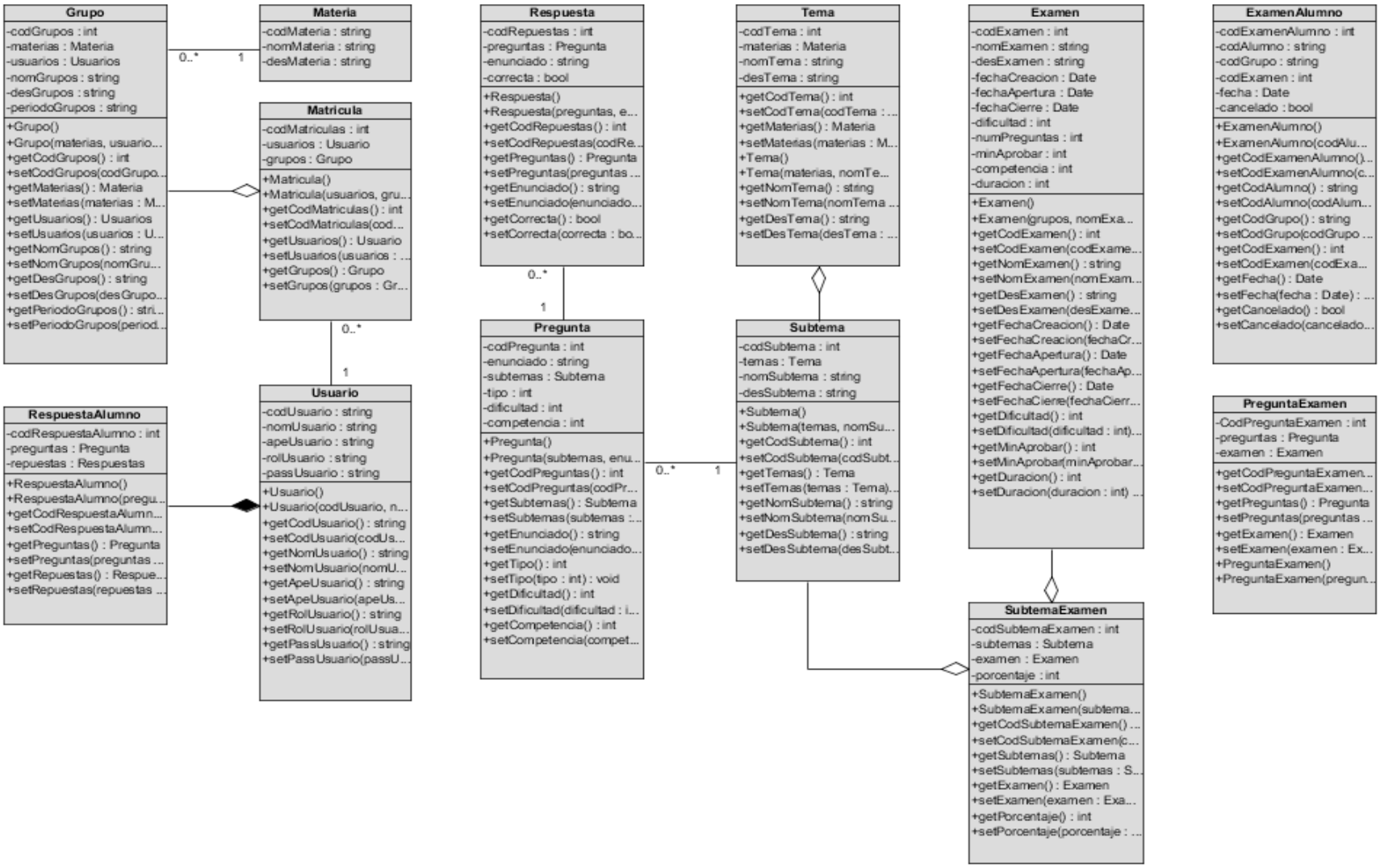

Ilustración 6 – Diseño de la base de datos.

Diagramas de Actividad

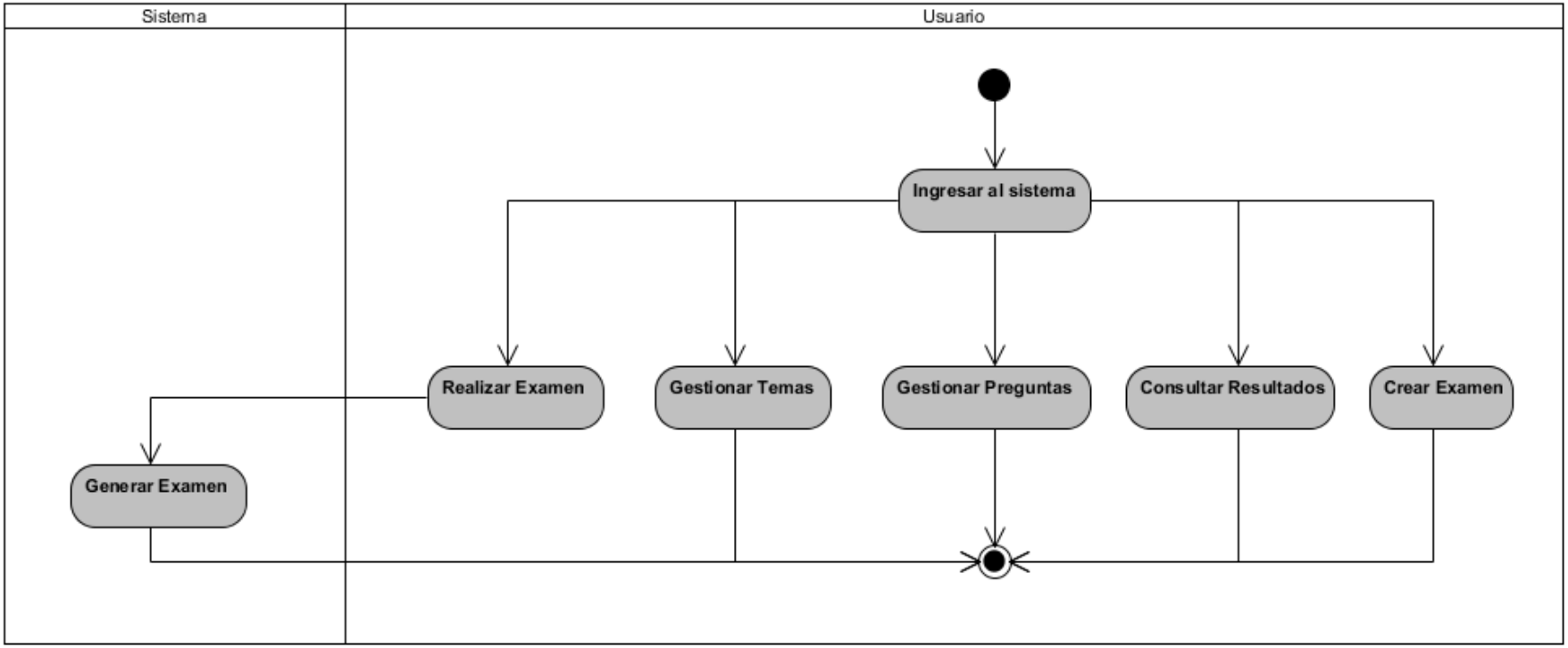

Ilustración 7 – Sistema de Evaluación Virtual.

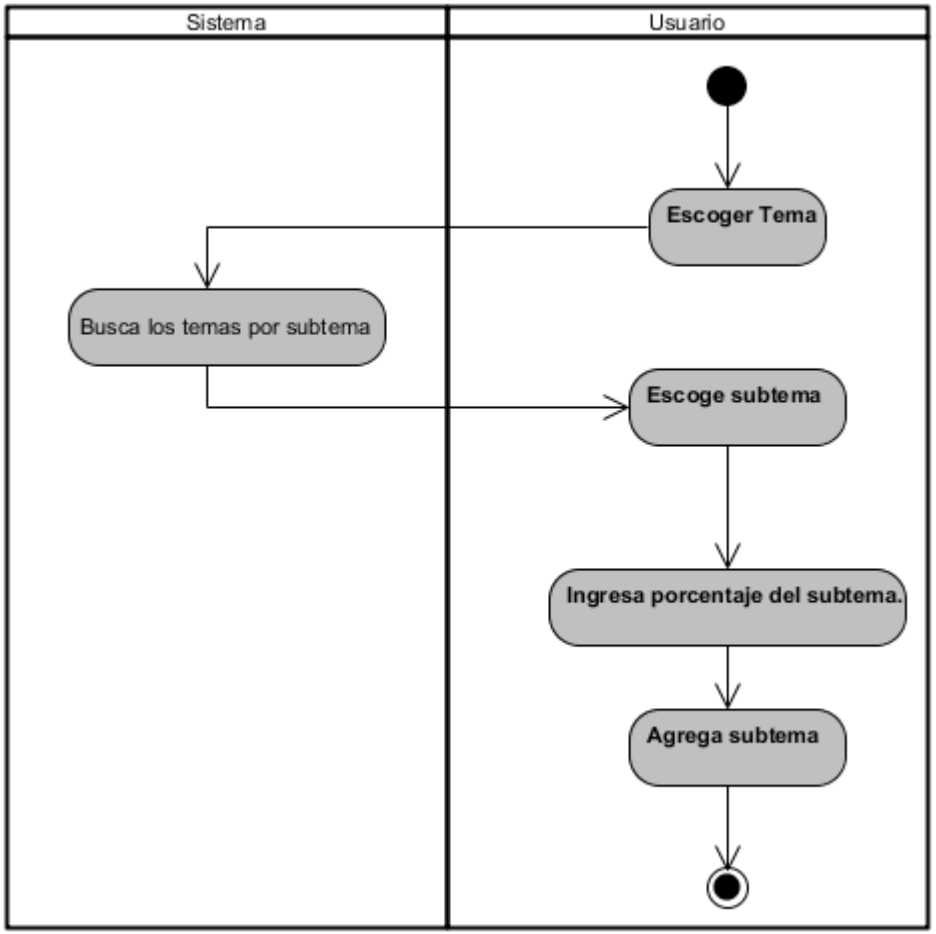

Ilustración 8 – Agregar subtema a examen

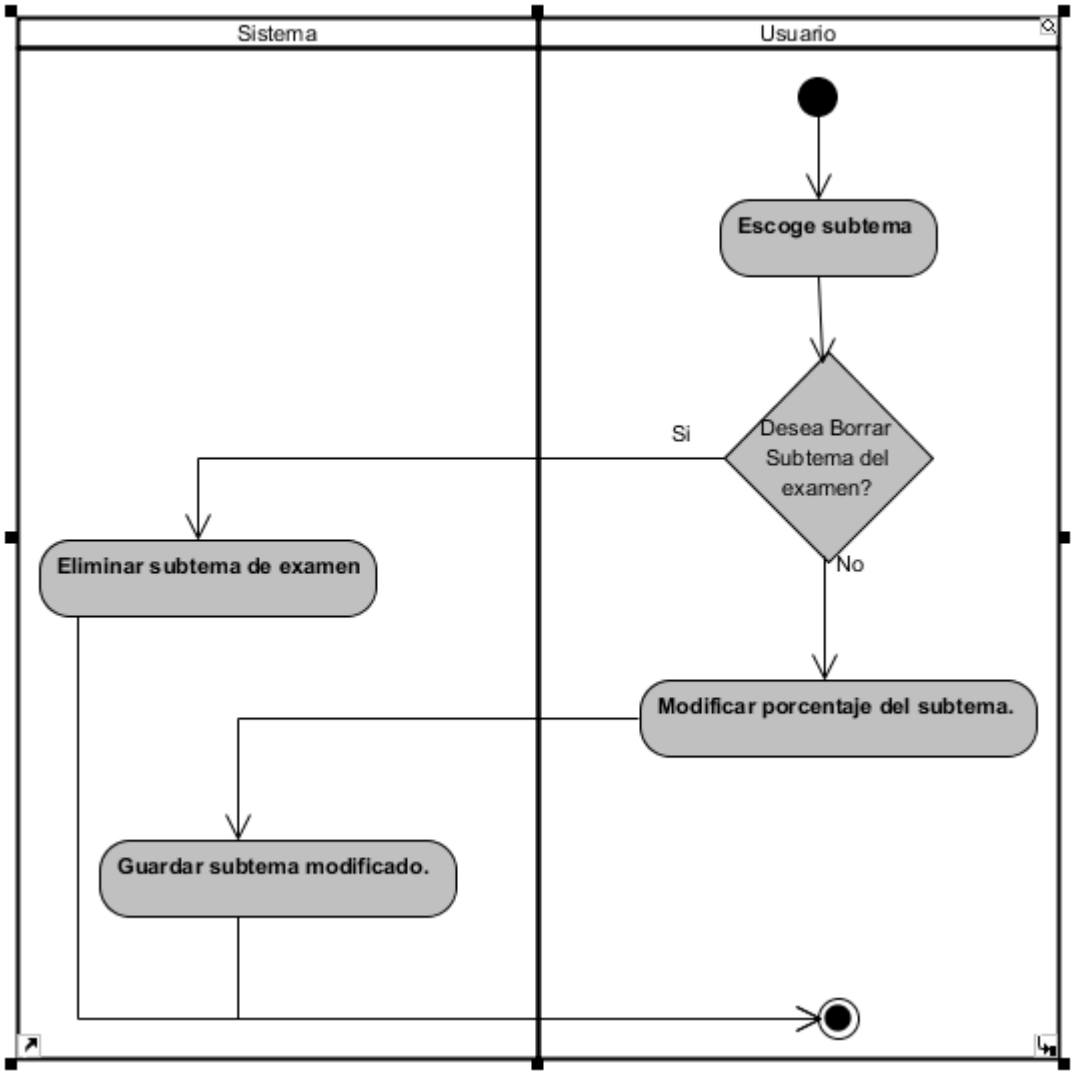

Ilustración 9– Modificar subtemas agregados a un examen

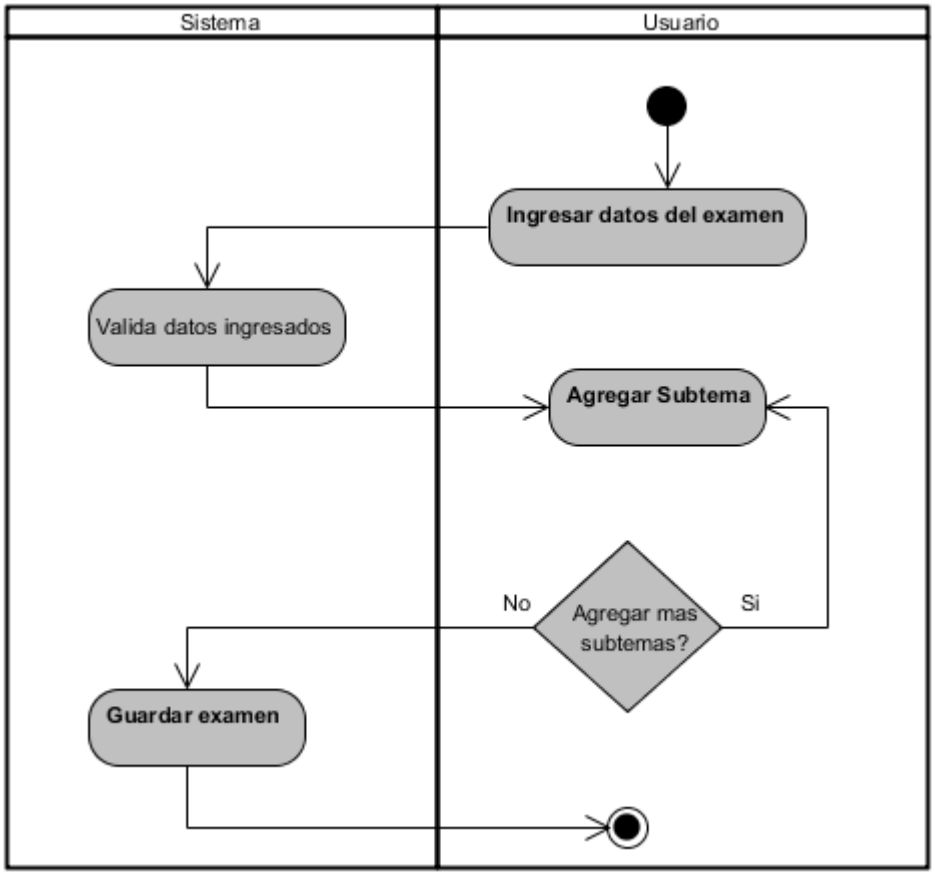

Ilustración 10 – Creación de examen.

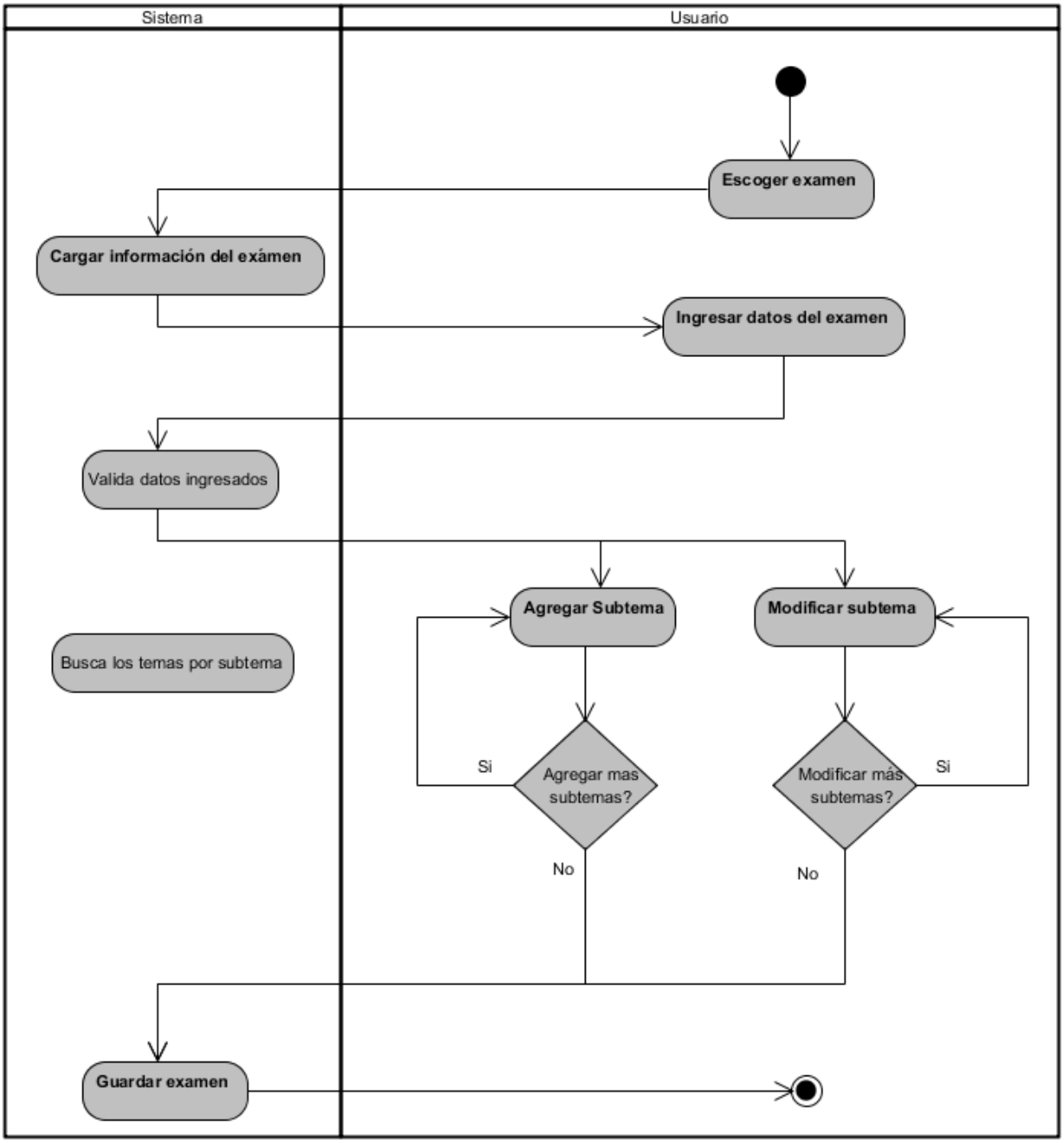

Ilustración 11 – Modificación de examen.

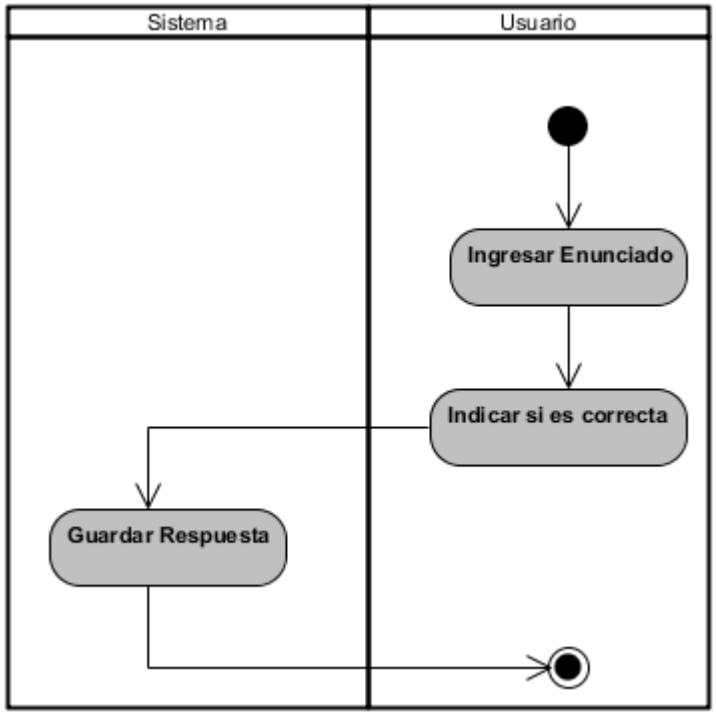

Ilustración 12 – Agregar respuesta a una pregunta.

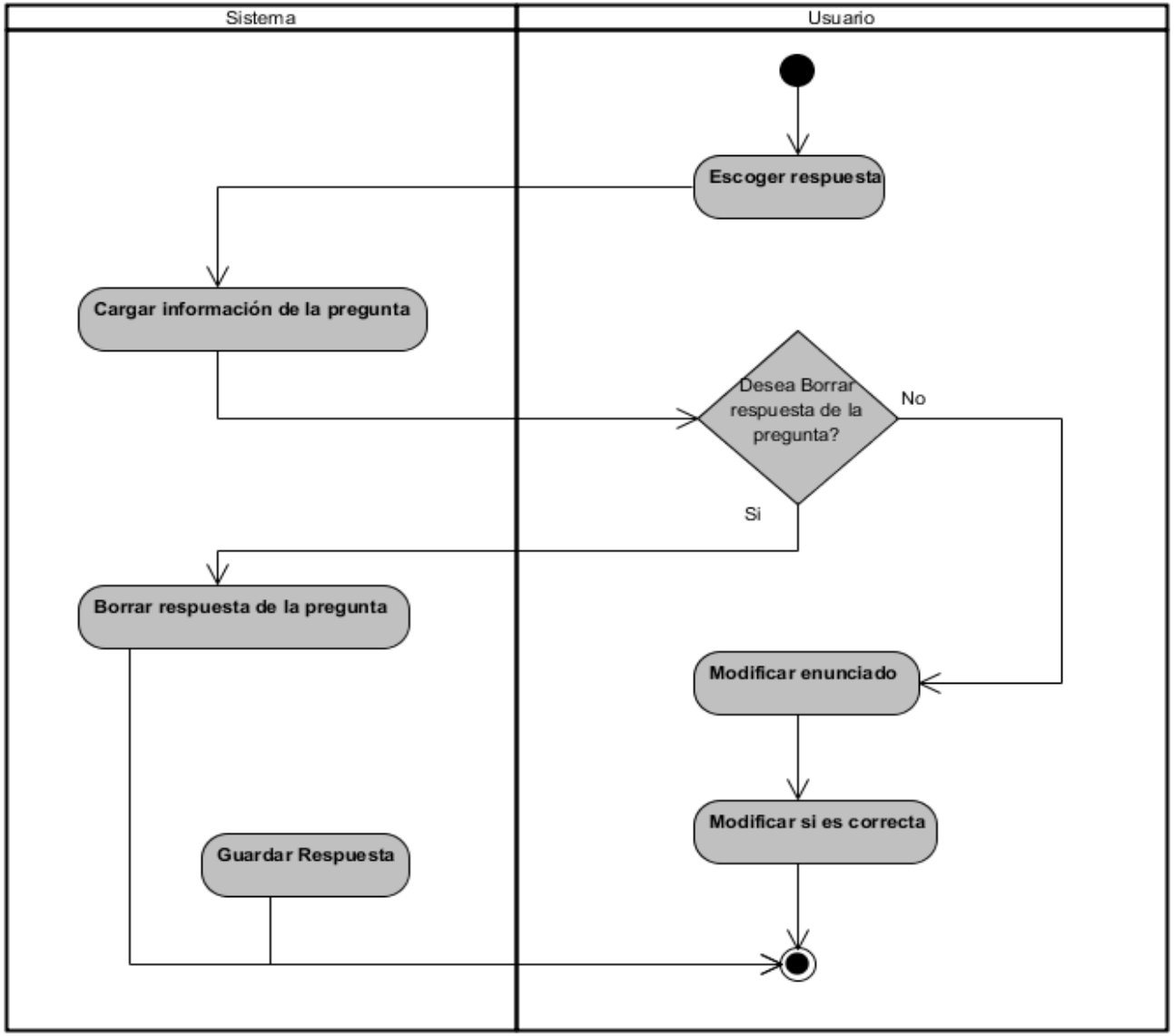

Ilustración 13 –Modificar una respuesta de una pregunta.

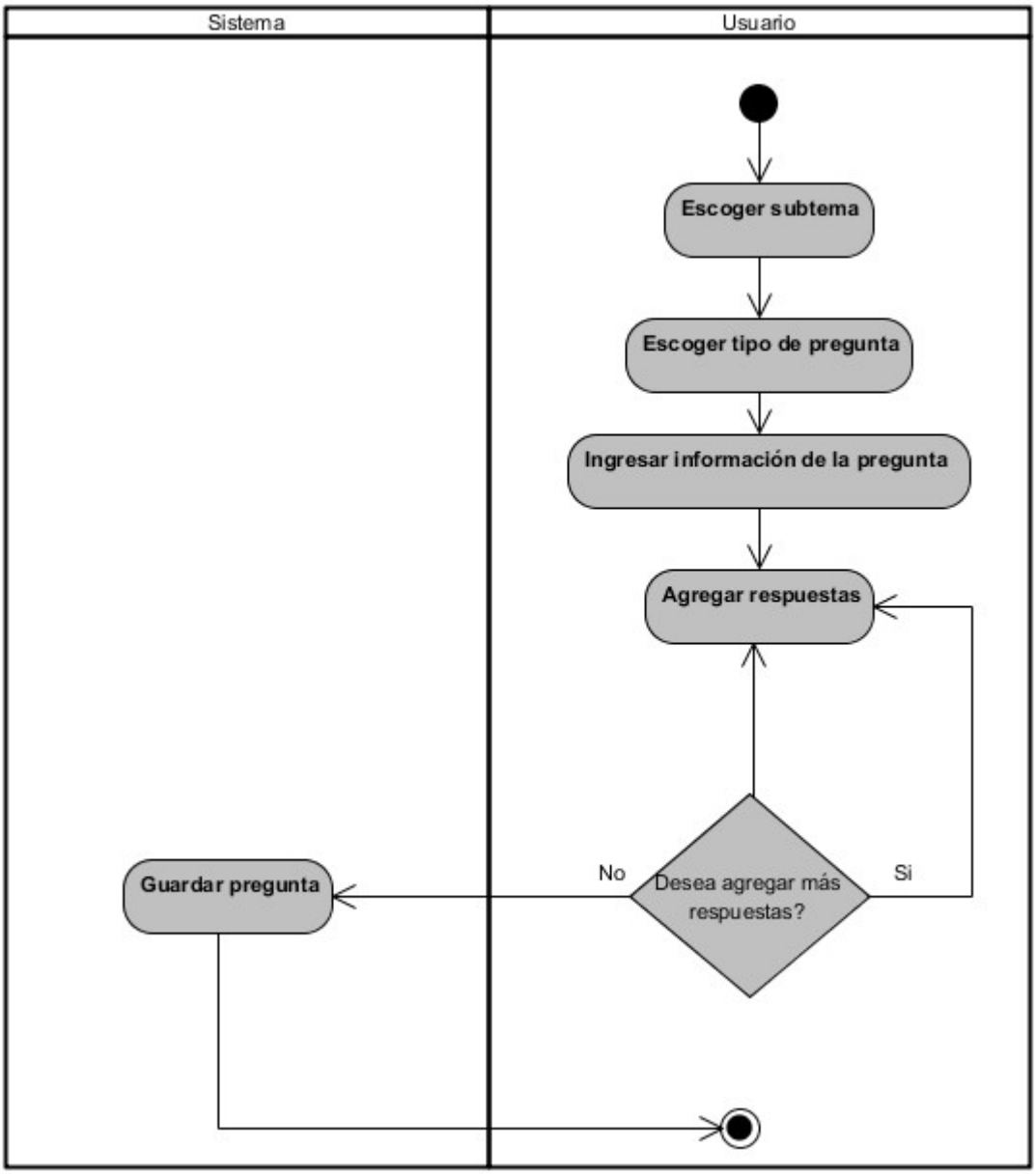

Ilustración 14 –Crear Pregunta.

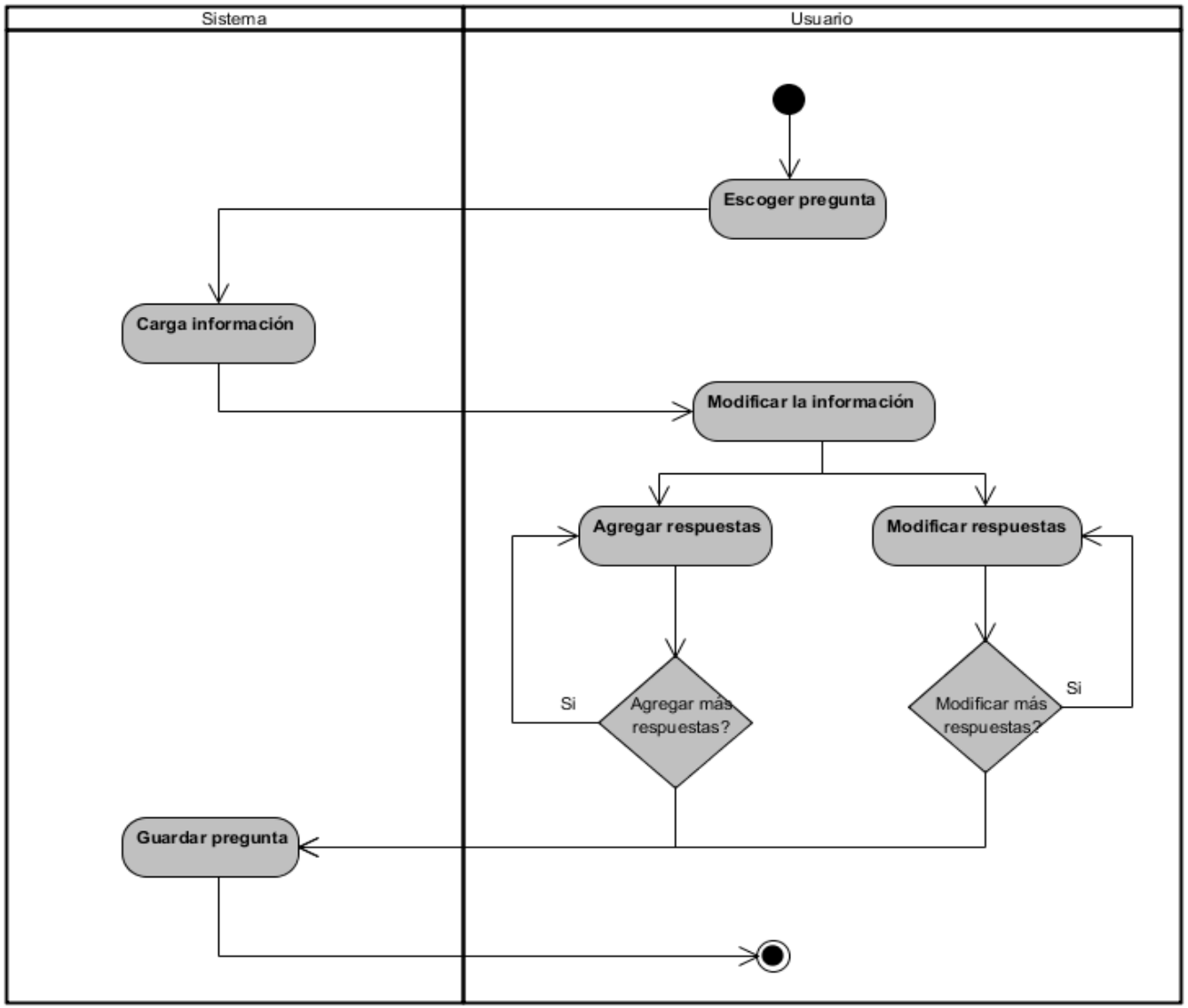

Ilustración 15 –Modificar Pregunta.

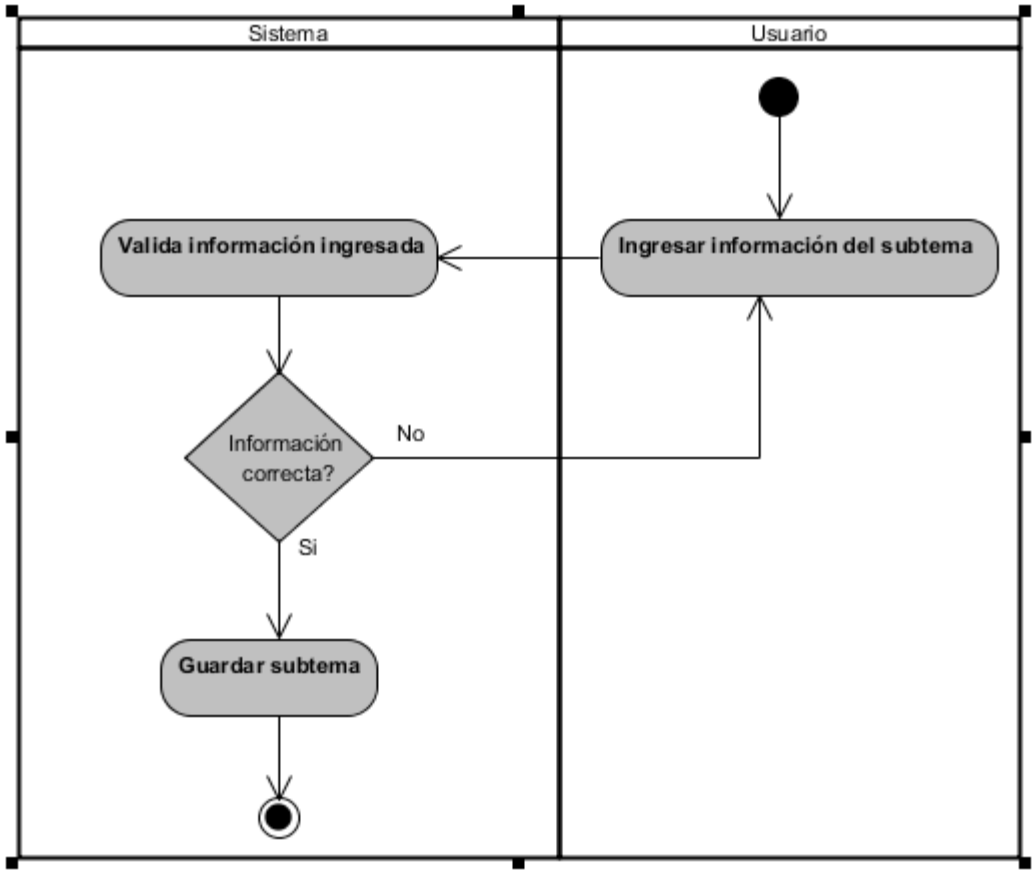

Ilustración 16 –Agregar subtema.

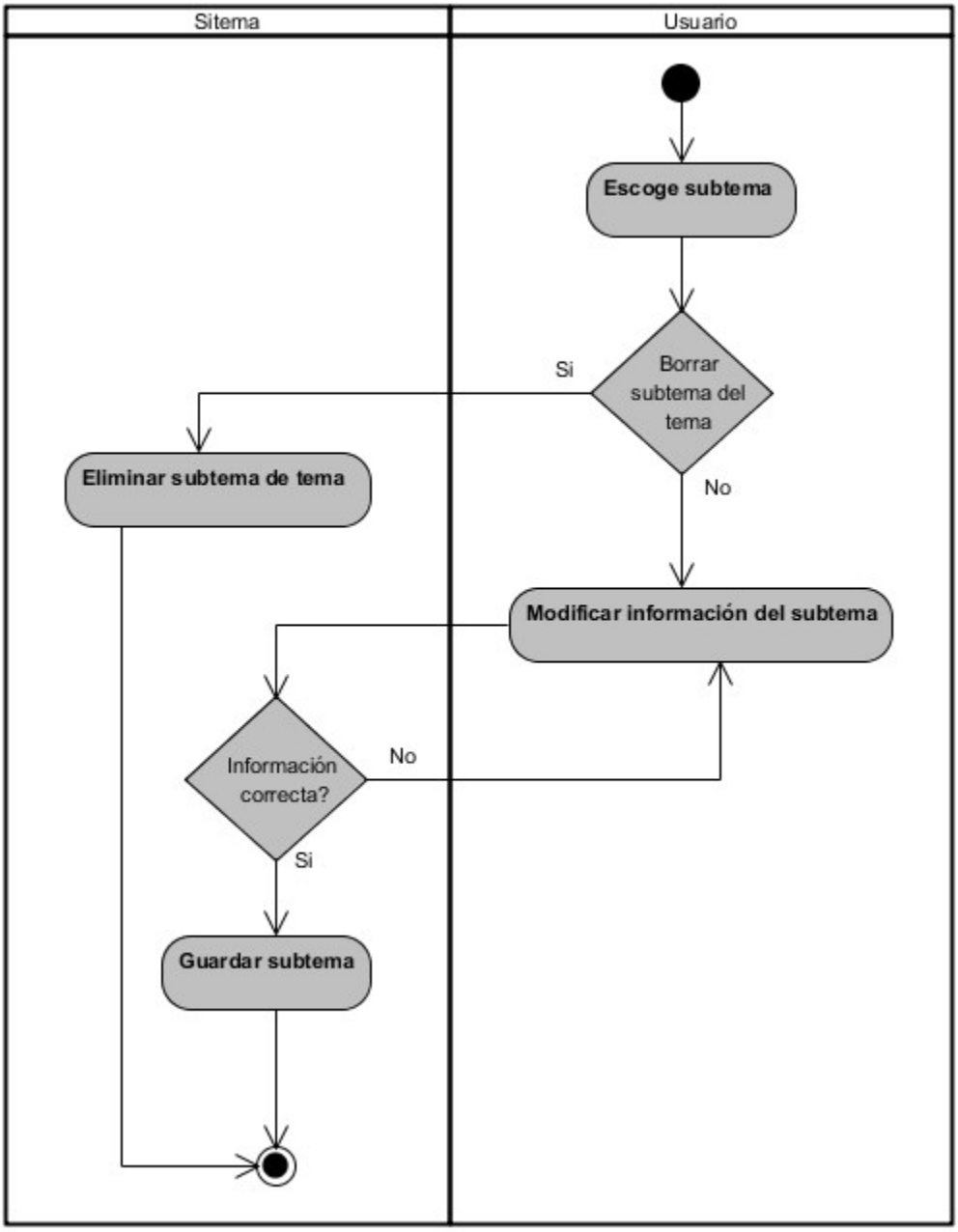

Ilustración 17 –Modificar subtema.

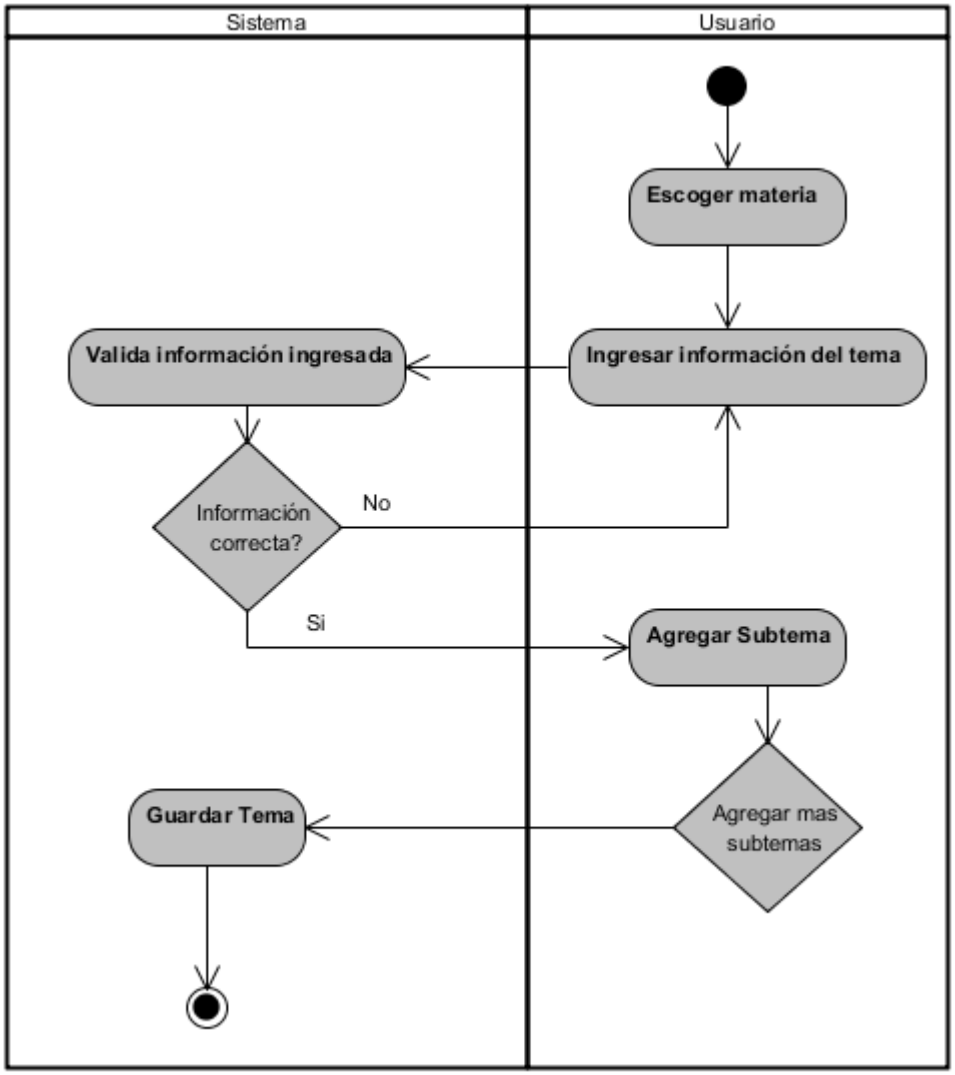

Ilustración 18 –Agregar Tema

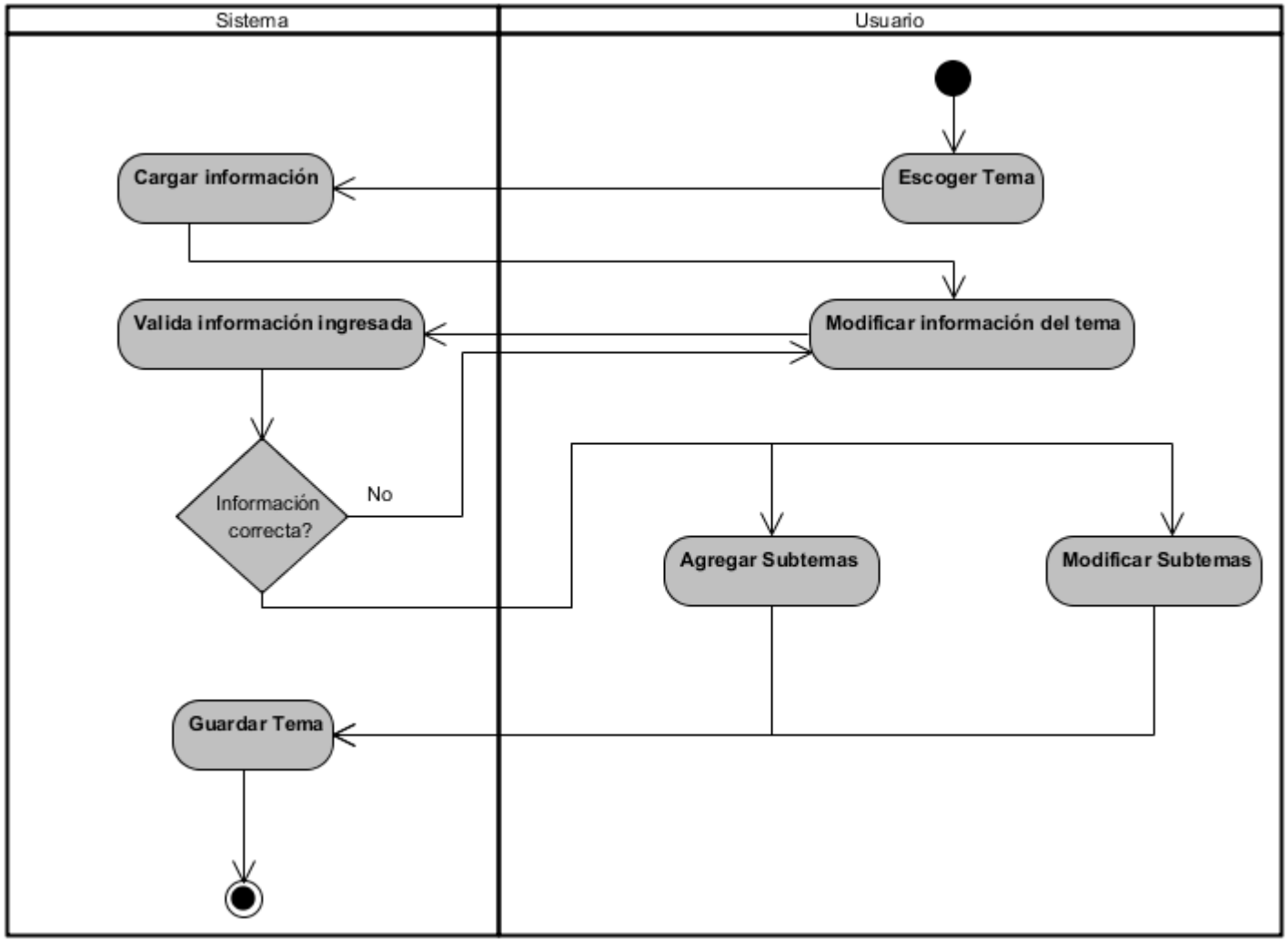

Ilustración 19 – Modificar Tema

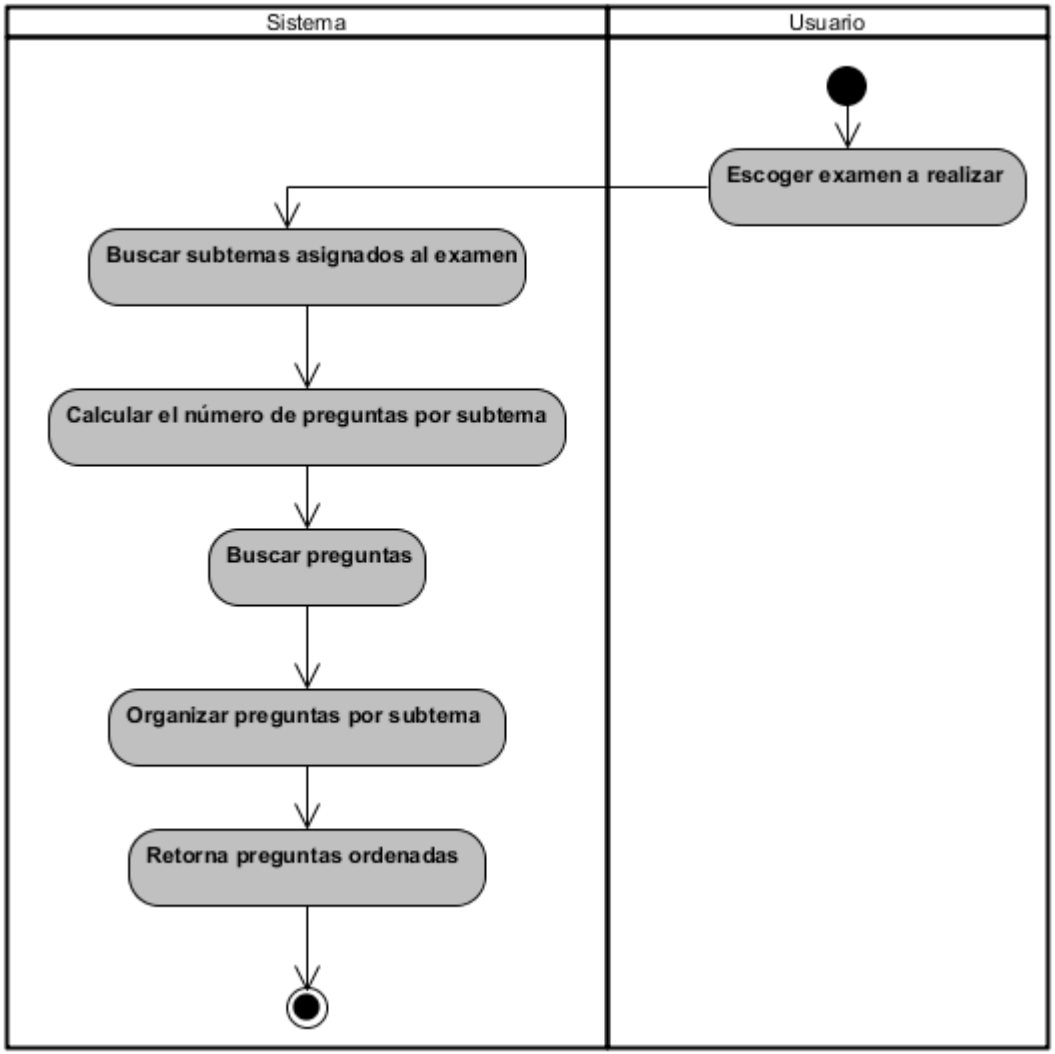

Ilustración 20 – Generar Examen.

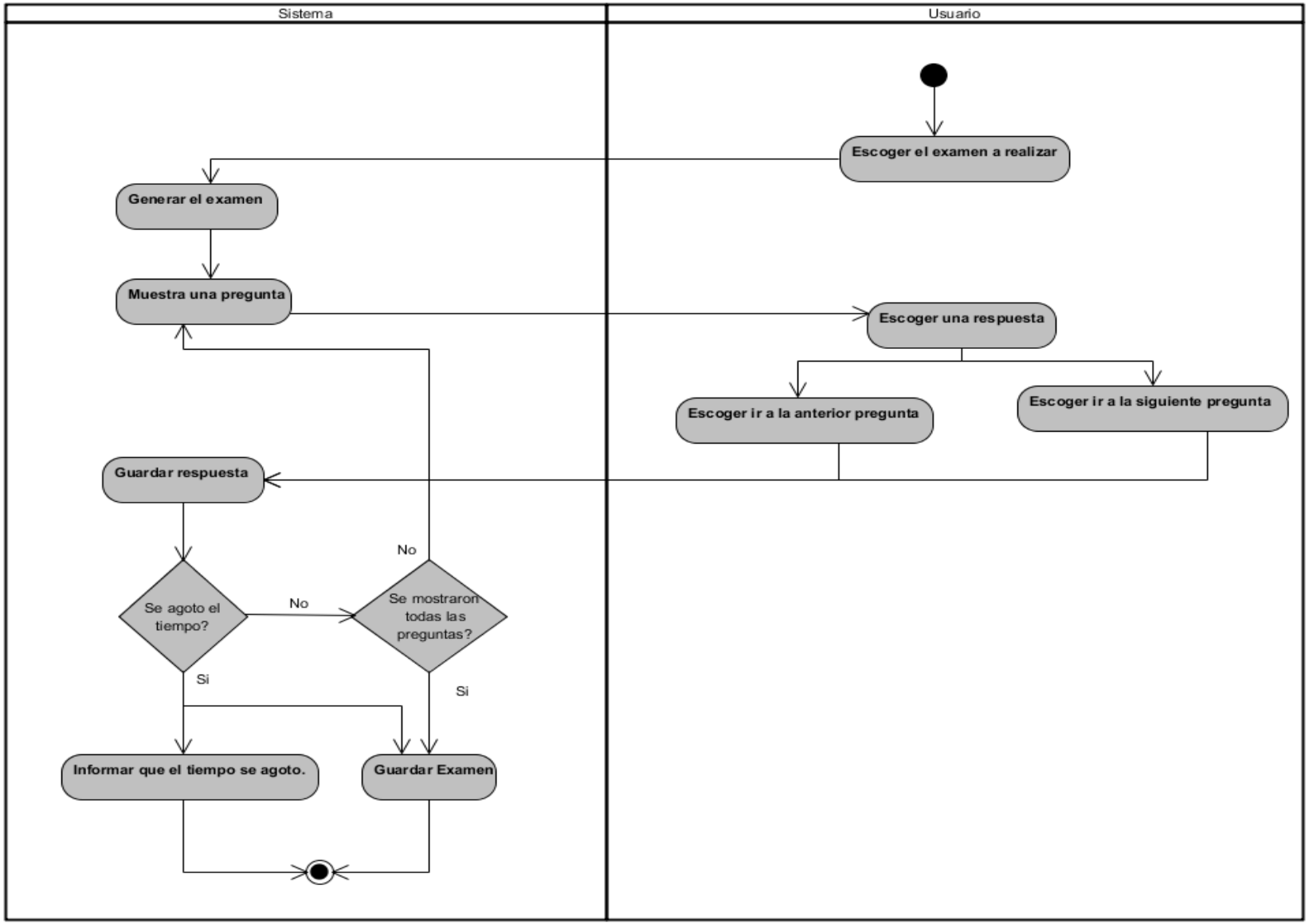

Ilustración 21 – Realizar examen.

Conclusiones

Se ha mostrado cómo integrar JSF con el marco de trabajo Spring e Hibernate para construir una aplicación Web del mundo real. La combinación de estas tres tecnologías proporciona un sólido marco de trabajo para el desarrollo de aplicaciones Web. Para este tipo de aplicaciones se debería utilizar una arquitectura multi-capa como arquitectura de alto nivel. JSF se acopla muy bien en el patrón de diseño MVC y se puede utilizar para implementar la capa de presentación. El marco de trabajo Spring se puede utilizar en la capa de lógica-de-negocio para manejar los objetos de negocio, y porporcionar control de transaciones declarativoo y control de recursos. Spring se integra muy bien con Hibernate. Hibernate es un poderoso marco de trabajo de mapeo O/R y puede proporcionar el mejor servicio dentro de la capa de integración.

Particionando toda la aplicación Web en capas y programando contra interfaces, se pueden reemplazar las tecnologías usadas para cada capa de la aplicación. Por ejemplo, Struts puede ocupar el lugar de JSF para la capa de presentación, y JDO puede reemplazar a Hibernate en la capa de integración. La integración entre las capas no es trivial. El uso de la inversión de control y del patrón de diseño *Service Locator* puede hacerlo más sencillo. JSF proporciona funcionalidades que no poseen otros marcos de trabajo como Struts. Sin embargo, esto no significa que usted tenga que tirar Struts y empezar a utilizar JSF ahora mismo. Si JSF debe utilizarse o no como marco de trabajo Web para sus aplicaciones, depende del estado de sus proyectos, de los requerimientos funcionales, y de la experiencia de su equipo.

Bibliografia

- Bellows, Jeannie, Castek (2000). Activity Diagrams and Operation Architecture. Technologies Group Inc..
- Linwood, Jeff; Minter, Dave (May 28, 2010), *Beginning Hibernate* (Second ed.), Apress, pp. 400, ISBN 1430228504
- Linwood, Jeff; Minter, Dave (August 25, 2006), *Beginning Hibernate: From Novice to Professional* (Third ed.), Apress, pp. 360, ISBN 1590596935
- http://www.hibernate.org/
- Juntao Yuan, Michael; Heute, Thomas (May 6, 2007). *JBoss Seam: Simplicity and Power Beyond Java EE* (1st ed.). Prentice Hall. pp. 432. ISBN 978-0131347960.
- Chris Schalk, Ed Burns, James Holmes: JavaServer Faces: The Complete Reference, McGraw-Hill Osborne Media, ISBN 0-07-226240-0