VNIVERSIDAD NACIONAL
AVTNOMA DE
MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

REPOSITORIO DE MÉTODOS Y PRÁCTICAS DE PROYECTOS DE SOFTWARE PARA KUALI-BEH

T E S I S

QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN INGENIERÍA (COMPUTACIÓN)

PRESENTA:
RODRIGO ALBERTO BARRERA HERNÁNDEZ

DIRECTORA DE TESIS:
DRA. HANNA JADWIGA OKTABA
FACULTAD DE CIENCIAS – UNAM

MÉXICO, D. F. JUNIO 2014

Dedicatoria

A mis dos ángeles del cielo (Mis papás):

María Belén y Jesús Barrera

Gracias por seguir cuidándome...

Agradecimientos

Agradezco a mi madre: Ma. Soledad, por su siempre apoyo incondicional.

A mis hermanos: Sonia, Raymundo y Arturo.

A Blanca, por sus lecciones de vida.

A la Dra. Hanna, que con su ejemplo me instruyó y motivó, personal y profesionalmente.

A Miguel, por su tiempo y apoyo en la realización de esta tesis.

A todas las personas involucradas en el desarrollo de este trabajo.

A la UNAM y a este Posgrado.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

REPOSITORIO DE MÉTODOS Y PRÁCTICAS DE PROYECTOS DE SOFTWARE PARA KUALI-BEH

RESUMEN

KUALI-BEH es una propuesta mexicana que busca ayudar a redefinir la Ingeniería de Software, de tal manera que se califique como una disciplina rigurosa. Esta propuesta permite respaldar la forma de trabajo de las organizaciones de desarrollo de software. Para brindar un soporte a esta propuesta se creó el proyecto Entorno Computacional para KUALI-BEH, de la cual este trabajo forma parte.

El siguiente trabajo de tesis presenta la creación de un repositorio, basándose en la propuesta de KUALI-BEH, específicamente al apartado correspondiente a la vista estática. El repositorio permitirá el almacenamiento, búsqueda, organización e implementación de los conocimientos de los practicantes de las organizaciones de desarrollo de software, en forma de Métodos y Prácticas.

Este documento además contiene los apéndices: Especificación de Requerimientos, Análisis y Diseño y Pruebas, en los cuales se documentan los requisitos funcionales y no funcionales, los resultados de las actividades de Análisis y Diseño, y las pruebas de funcionalidad y usabilidad aplicadas al sistema.

1. INTRODUCCIÓN

A continuación se presentan algunos antecedentes, el planteamiento del problema y la propuesta de solución a este proyecto. Además se incluye una descripción general del contenido de la tesis.

1.1 Antecedentes y Motivación

Bajo la premisa de que la Ingeniería de Software se encuentra gravemente obstaculizada por prácticas inmaduras, específicamente:

- La prevalencia de tendencias que son más comunes en la industria de la moda que en una disciplina ingenieril.
- La ausencia de bases teóricas ampliamente aceptadas y difundidas.
- La gran cantidad de métodos y variantes de métodos, con diferencias poco entendidas y ampliadas de forma artificial.
- La carencia de evaluación y validación experimental creíble.
- La división entre la práctica en la industria y la investigación académica [1]

Se inició el esfuerzo denominado Software Engineering Method and Theory (SEMAT) [1] fundado por la Troika: Ivar Jacobson, Bertrand Meyer y Richard Soley en diciembre de 2009, a través del cual se busca refundamentar la

Ingeniería de Software de tal manera que se califique como una disciplina rigurosa.

Para impulsar este esfuerzo en Marzo de 2010 en Zurich, Suiza, la comunidad de SEMAT estableció una Declaración de Llamado a la Acción (Call for Action). [1] [2].

Basándose en este Llamado a la Acción, Object Management Group (OMG) [3], quien es un consorcio internacional sin ánimo de lucro dedicado al cuidado y al establecimiento de diversos estándares de tecnologías orientadas a objetos, lanzó el 26 de Junio de 2011, una petición bajo el nombre de: “A Foundation for the Agile Creation and Enactment of Software Engineering Methods (FACESEM) RFP” [4] en la que se invita a la comunidad de la Ingeniería de Software a atender las carencias identificadas por SEMAT.

El OMG está formado por diversas compañías y organizaciones con distintos privilegios dentro de la misma, algunos de ellos son:

- Hewlett-Packard
- HSBC
- IBM
- Eclipse Foundation
- Microsoft Corporation
- Lockheed Martin

Así como diversas universidades, entre las que se encuentra la Universidad Nacional Autónoma de México (UNAM), como miembro Platform [5].

La UNAM, como miembro de este consorcio, y específicamente el grupo de Posgrado de Ciencia e Ingeniería de la Computación (PCIC), liderado por la Dra. Hanna Oktaba, decidió responder a esta solicitud desarrollando la propuesta KUALI-BEH [6]. Esta propuesta se fundamenta en el conocimiento y experiencia obtenidos en múltiples proyectos, que incluyen la participación activa en la

definición de estándares de desarrollo de software [7] [8] [9], tanto a nivel nacional como internacional.

KUALI-BEH se compone de dos vistas: estática y operacional.

La **vista estática** proporciona un marco de trabajo para la definición de las diferentes formas de trabajo de los practicantes. Estas formas de trabajo son organizadas en métodos compuestos por prácticas. Este conocimiento se estructura en una infraestructura de métodos y prácticas que pueden ser aplicadas por los practicantes [6].

La **vista operacional** está relacionada con la ejecución del proyecto de software. Esta vista proporciona a los practicantes mecanismos para ejecutar y adaptar un método con sus prácticas de acuerdo a las necesidades de los principales interesados y a un contexto específico [6].

1.2 Planteamiento del Problema

A las organizaciones en general, se les dificulta la adopción de una nueva forma de trabajo aun sabiendo que la implementación de la misma les permitirá ser más competitivas y elevar su calidad dentro de la industria. Esto debido a que no cuentan con una herramienta que les facilite esa implementación de forma fácil, rápida y eficiente.

En las pruebas que se han venido realizando para la adopción de KUALI-BEH, se ha observado la necesidad de una herramienta de software que facilite la administración, almacenamiento y ejecución de los conceptos planteados en él.

1.3 Propuesta de Solución

Para brindar un soporte a esta propuesta, se vio la necesidad de crear el proyecto Entorno Computacional para KUALI-BEH, el cual apoyará a las organizaciones que desarrollan software en las siguientes actividades:

1. Expresar sus métodos y prácticas empíricas de manera estructurada.
2. Resguardar esos métodos y prácticas en un repositorio de la organización.
3. Seleccionar y adaptar al contexto de un proyecto de desarrollo de software algún método y prácticas del repositorio de la organización.
4. Aplicar el método y sus prácticas durante la ejecución del proyecto, dándole seguimiento a través de tableros de control.
5. Enriquecer y mejorar el repositorio de la organización a partir de la experiencia en los proyectos y el conocimiento adquirido de fuentes externas.

A fin de construir este entorno computacional, se divide el trabajo en cuatro sub-proyectos que consisten en 4 tesis:

1. Arquitectura de software para el entorno computacional de KUALI-BEH [10].
2. Repositorio de métodos y prácticas de proyectos de software para KUALI-BEH.
3. Selección y adaptación de métodos en proyectos de software aplicando KUALI-BEH [11].
4. Tableros de control para el seguimiento de proyectos de software aplicando KUALI-BEH [12].

Esta tesis se encargará de realizar el punto 2, del conjunto de proyectos de tesis.

1.4 Contenido de la tesis

A continuación se describe la estructura de esta tesis.

En el capítulo 2 se presenta el marco teórico que describe los conceptos sobre los cuales se trabaja en esta tesis. Dentro del marco teórico se describe el proyecto KUALI-BEH, su propósito, las personas a las que va dirigido y se hace énfasis en la descripción de la vista estática, que es el concepto en el que se basa esta tesis para la creación de la herramienta de repositorio. Posteriormente se describe lo que es un repositorio digital, los tipos de repositorios existentes y sus características. Como último punto, se hace un listado de los repositorios existentes en un concepto similar al requerido en esta tesis.

El capítulo 3 describe el proceso de desarrollo del sistema, haciendo mención a las metodologías de desarrollo utilizadas durante su realización. Se documentan los requerimientos del sistema, así como la arquitectura utilizada. Más adelante en este mismo capítulo se describen las herramientas de desarrollo empleadas y el por qué se tomó la decisión de hacer uso de ellas. La construcción y pruebas son otro de los puntos que se describen en este capítulo.

Las conclusiones y trabajo a futuro, son descritas en el capítulo 4.

2. MARCO TEÓRICO

En este capítulo se presentan las bases, conceptos y contexto de los elementos asociados a este trabajo de tesis.

Como punto de partida se explicará el proyecto KUALI-BEH, enfocándose posteriormente a la vista estática ya que esta tesis se basa en los conceptos descritos en ella para la construcción del repositorio.

Posteriormente, se explica a detalle lo que es un repositorio digital y de la misma forma se realiza un estudio de las distintas herramientas y prototipos existentes en el mercado y que son similares al concepto maneado por KUALI-BEH para posteriormente ser comparadas con las características requeridas en la construcción del sistema en esta tesis.

2.1 KUALI-BEH

KUALI-BEH describe los conceptos comunes y sus relaciones presentes en cualquier proyecto de software. Es una propuesta mexicana en respuesta a la solicitud del Object Management Group A Foundation for the Agile Creation and Enactment of Software Engineering Methods [4]. Esta propuesta se fundamenta en el conocimiento y experiencia obtenidos en múltiples proyectos, incluyendo la definición de estándares de desarrollo de software [7] [8] [9].

Los Ingenieros de Software practicantes que están involucrados activamente en proyectos de software son el principal objetivo de la propuesta de KUALI-BEH. Los ingenieros de método, quienes se encargan de definir métodos para proyectos de software, son otro de los grupos que pueden estar en el alcance de esta propuesta.

KUALI-BEH se compone de dos vistas: Estática y Operacional. Ver figura 1.

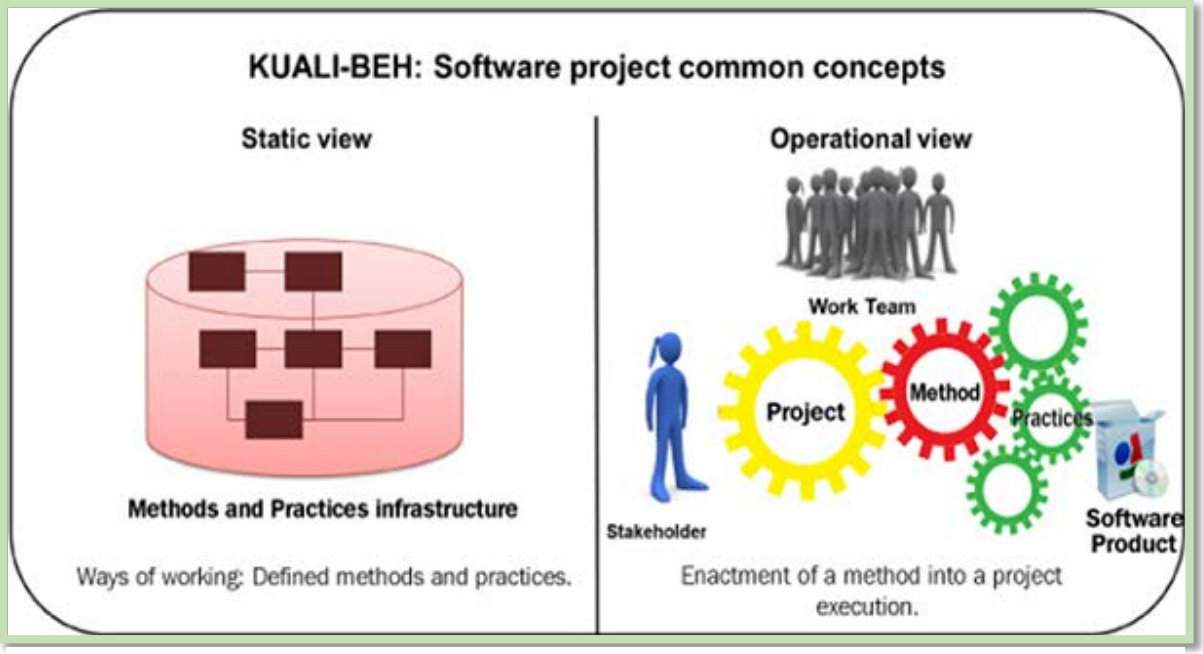


Figura 1. Vistas de KUALI-BEH [6].

A continuación se describe la estructura y conceptos contenidos en la vista estática.

2.1.1 Vista Estática

Un **proyecto de software** es un esfuerzo temporal que realiza un grupo de **practicantes** de Ingeniería de Software y que tiene como finalidad la creación, mantenimiento, integración o realización de pruebas de un **producto de**

software. Los proyectos de software nacen de las necesidades de un individuo u organización (***Stakeholders***) en obtener algún **producto de software**.

La **vista estática** se encarga de resguardar las diversas formas de trabajo que utilizan los **practicantes** de Ingeniería de Software, normalmente **conocimientos** y **habilidades** que los mismos practicantes poseen y que utilizan para la realización de los distintos **proyectos de software** en los que participan.

Estas formas de trabajo se almacenan dentro de una infraestructura de **métodos** compuestos por **prácticas**. El objetivo de realizar esta infraestructura es la de permitir a los mismos practicantes facilitar la administración, consulta y reutilización de todo este conocimientos como unidades de trabajo.

Una **práctica** es un conjunto de actividades y tareas que se utilizan en la ejecución de los **proyectos de software**, estas contienen la forma de trabajo de los practicantes y permite la reutilización de estos conceptos en proyectos posteriores. Una práctica posee un objetivo, el cual se describe al momento de su creación, también permite definir entradas y uno o varios resultados, los cuales, van acorde al objetivo planteado en su inicio. Las entradas y resultados pueden ser **productos de trabajo** o **condiciones.**

A su vez, las prácticas se encuentran compuestas por una o varias **guías** de trabajo, las cuales contienen diferentes **actividades.** Cada actividad almacena **tareas**, **métricas**, **criterios de verificación, herramientas** y **conocimientos y habilidades** que tienen la finalidad de transformar las diferentes entradas en los resultados previamente establecidos dentro de la práctica.

Los **métodos** son la unión de un conjunto de prácticas, estos tienen sus propios objetivos, sin embargo estos objetivos o propósitos deben estar acorde con el conjunto de prácticas que contiene y viceversa. En general, el conjunto de prácticas que comprende un método debe ser coherente, consistente y completo. Un conjunto de prácticas es coherente si el objetivo de cada práctica contribuye con el propósito del método. Es consistente si cada una de sus entradas y

resultados están interrelacionados y son útiles. Finalmente, es completa si el logro de todos los objetivos de las prácticas, cumplen totalmente el propósito del método y se produce un producto de software esperado.

Los conceptos de KUALI-BEH se pueden aplicar para definir métodos y prácticas de cualquier proyecto de software, independientemente del tamaño, complejidad, modelo de ciclo de vida utilizado o tecnología. Estos métodos y prácticas son definidos por los propios practicantes de ingeniería de software, KUALI-BEH permite adaptar y actualizar cada uno de ellos a los proyectos de software en los que se necesite.

En la Figura 2 se muestran los conceptos comunes de un proyecto de software mediante un diagrama de clases UML. Cada concepto se representa como una clase y sus conexiones lógicas como las relaciones.

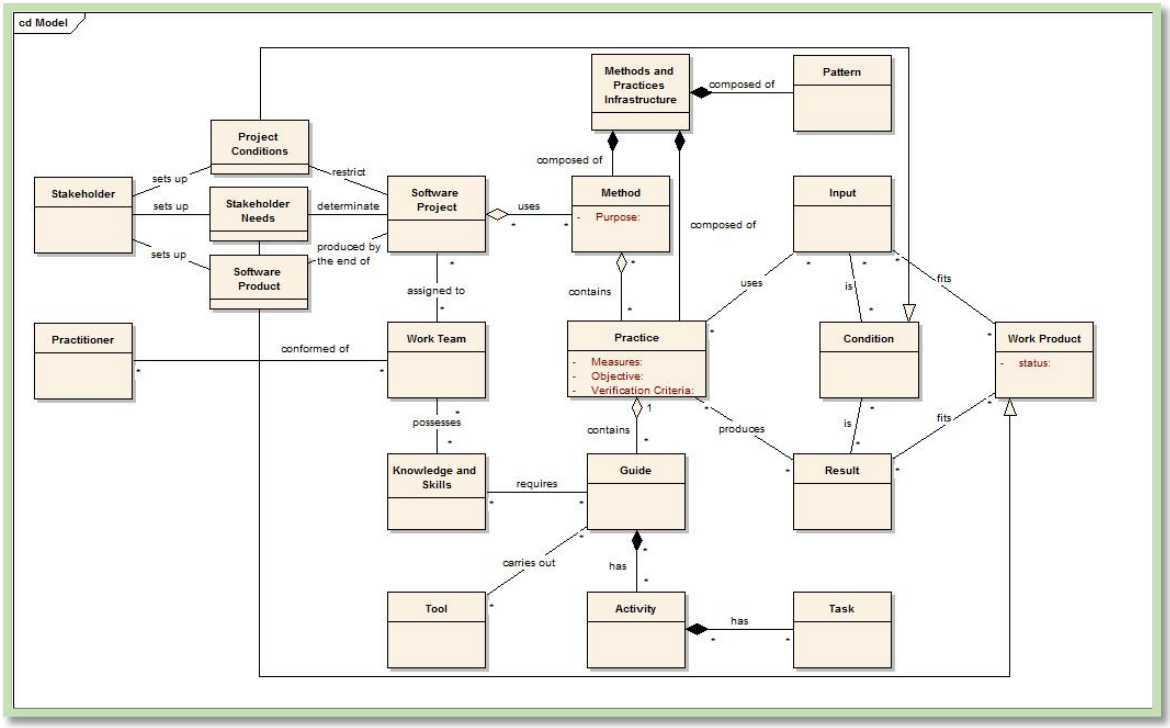


Figura 2. Proyecto de software. Conceptos comunes, sus relaciones y atributos [6].

Glosario de Conceptos

El siguiente glosario de conceptos mostrado en la tabla 1, lista algunos de los conceptos mencionados en el diagrama de clases y que son utilizados en este proyecto [6].

Esta tesis se basa en construir la herramienta para la vista estática, el repositorio permitirá almacenar y gestionar los conocimientos de los practicantes de ingeniería de software para posteriormente llevar a cabo su ejecución y adaptación a cada proyecto en particular.

Tabla 1. Glosario de Conceptos.

NOMBRE	DEFINICIÓN
Proyecto de Software (Software Project)	Es un esfuerzo temporal que tiene como finalidad la realización de un producto de software que satisfaga las necesidades planteadas de los Stakeholders. Este esfuerzo es realizado por un equipo de trabajo integrado por practicantes activos de la ingeniería de software.
Producto de Trabajo (Work Product)	Es el recurso generado como resultado o utilizado como entrada de una práctica, actividad o método.
Equipo de Trabajo (Work Team)	Es un grupo de practicantes de ingeniería de software que trabaja en conjunto para la realización de los objetivos planteados a lo largo de un proyecto de software
Producto de Software (Software Product)	Un producto de software es el resultado de la ejecución del método. Puede contener un conjunto de programas informáticos, procedimientos y documentación asociada. Esta es una especialización de un Producto de Trabajo [1].
Actividad (Activity)	Es un conjunto de conocimientos y habilidades, tareas, métricas, criterios de verificación y herramientas que contribuyen a la realización del objetivo de una práctica. Un conjunto de actividades

	conforman una guía.
Practicante **(Practitioner)**	Persona que forma parte del equipo de trabajo y que se encuentra participando activamente en un proyecto de software. El practicante debe contar con los conocimientos y habilidades requeridos para poder desempeñar las tareas y actividades contenidas en las prácticas en las que participa.
Condición **(Condition)**	Una condición es una situación o circunstancia de algo o alguien que tiene relación con el proyecto de software que se está realizando. Estas condiciones, pueden ser entradas o resultadas de una práctica, método o una actividad.
Guía **(Guide)**	Es un conjunto de actividades que unidas sirven de apoyo para lograr el objetivo de una práctica. Una práctica puede tener una o más guías que ayudan a transformar las entradas en resultados.
Entrada **(Input)**	Se define como las características esperadas por un producto de trabajo y / o las condiciones necesarias para iniciar la ejecución de una práctica, método o actividad.
Resultado **(Result)**	Se define como las características esperadas de un producto de trabajo y / o condiciones requeridas como salidas después de la ejecución de una práctica, guía o actividad.
Conocimientos y Habilidades **(Knowledge and Skills)**	Conjunto de habilidades, aptitudes y destrezas que son requeridas por el practicante de ingeniería de software para poder ejecutar una práctica.
Método **(Method)**	Es un conjunto **coherente**, **consistente** y **completo** de prácticas. Un conjunto de prácticas es **coherente** si el objetivo de cada práctica contribuye con el propósito del método. Es **consistente** si cada una de sus entradas y resultados están interrelacionados y son útiles. Es

	completa si el logro de todos los objetivos de las prácticas, cumplen totalmente el propósito del método.
Práctica **(Practice)**	Es un conjunto de guías, actividades y tareas que sirven para asesorar la producción de un resultado(s) a partir de una entrada(s). Una práctica contiene criterios de verificación asociados a cada una de sus actividades que ayudan a determinar que los resultados que se obtuvieron cumplen con el objetivo de la práctica. También cuenta con métricas que sirven para evaluar el rendimiento de cada una de las actividades que contiene y por lo tanto evalúan el rendimiento de la práctica en general. Cada práctica cuenta con un objetivo particular y el conjunto de guías y actividades contenidas en ella deberá servir de apoyo para su cumplimiento.
Stakeholder	Es un individuo u organización que tiene interés en la realización de un producto de software que satisfaga sus necesidades.
Herramienta **(Tool)**	Una herramienta es un documento, un software o cualquier dispositivo que ayude a los practicantes de ingeniería de software y al equipo en general a realizar una tarea.
Tarea **(Task)**	Una tarea es la mínima acción que se realiza dentro de una actividad.

2.2. Repositorios Digitales

Con la finalidad de tener cierto conocimiento sobre las herramientas de repositorios existentes en el mercado y que manejen un concepto similar a KUALI-BEH, se realizó un estudio sobre las mismas.

Un repositorio es un sistema de software que almacena objetos digitales tales como: tesis, libros, imágenes, artículos, reportes técnicos, además de

proporcionar algún tipo de interfaz de búsqueda de los mismos, o bien para la interacción con otros sistemas [17].

Algunas de las características que debe tener un repositorio dependiendo de su funcionalidad, son la posibilidad de almacenar distintos tipos de archivos, como contar con un sistema de búsqueda que agilice y facilite la localización de la información u objetos dentro del mismo, también pueden permitir la distribución y visualización del contenido.

Tipos de Repositorios:

Temático: Solo almacena información de un tema en específico sin importar si pertenece a una persona o institución.

Institucional: Lo ofrece una institución o comunidad para la difusión de los contenidos generados por ellos mismos.

Los repositorios digitales tienen como objetivos principales, poder difundir el conocimiento, dar visibilidad a la investigación que se ha realizado y conservar la documentación y el conocimiento generado.

En los puntos posteriores se presentan algunas herramientas para manejar.

2.2.1 Herramientas para Repositorios de Proyectos de Software

En este apartado se describen las características de algunas herramientas de repositorio y prototipos que existen en el mercado y son comparadas con las características requeridas en esta tesis. Dichas herramientas son utilizadas para gestionar los programas de mejora de procesos software y la ejecución de proyectos.

Las funcionalidades estudiadas en cada una de las herramientas fueron las siguientes:

- gestión de conocimiento.
- reutilización del know-how.
- gestión de proyectos.
- despliegue de modelos de procesos software.
- plataforma colaborativa.
- capacidad de búsquedas de conocimiento.

CodeBeamer

CodeBeamer es una plataforma de desarrollo colaborativo soportada por tecnología J2EE que ofrece aplicaciones con características de gestión de ciclo de vida de una aplicación [14]. Esta herramienta permite a los miembros de un grupo de trabajo visualizar el progreso de un proyecto en tiempo real y de manera colaborativa. Proporciona información sobre los avances y el estado de los proyectos en sus distintas etapas. Gracias a los informes que genera esta herramienta se puede mantener la gestión del proyecto incluso de forma externa.

Sin embargo esta herramienta no tiene un componente formal para la gestión de conocimiento que le proporcione capacidad para reutilizar el conocimiento de una forma automática y sencilla a las organizaciones e incluso poder modificar dicho conocimiento para su utilización en proyectos futuros.

IRIS Process Author

IRIS Process Author [15] es un sistema de gestión de proceso visual que permite la personalización de activos de proceso de forma colaborativa. Esta herramienta también permite la administración de proyectos exportando el contenido a otras herramientas como Microsoft Project.

Aunque esta herramienta tiene algunas características de gestión de conocimiento, carece de poder reutilizar el conocimiento de la organización y proyectos en trabajos futuros y de poder actualizar dichos conocimientos y adaptarlos.

Microsoft Visual Studio Team System

Microsoft Visual Studio Team System [16], es una plataforma para herramientas del ciclo de vida del desarrollo de software extensible, integrado y productivo que ayuda a los equipos de desarrollo de software mediante la mejora de las comunicaciones y la colaboración durante todo el proceso de desarrollo.

Sin embargo esta herramienta no maneja la gestión de conocimiento, lo que conlleva el no poder reutilizar el conocimiento en proyectos futuros.

Building an Organization Repository of Experiences

Building an Organization Repository of Experiences (BORE) [24] es una herramienta creada con la finalidad de mejorar la calidad en el desarrollo de software utilizando las experiencias y conocimiento de los desarrolladores.

La herramienta BORE pretende utilizar las lecciones aprendidas en proyectos anteriores, utilizando soluciones para resolver proyectos similares en los que se este trabajando. De esta manera se agiliza el desarrollo del proyecto y se mejora la calidad del producto final. Para lograr lo anterior, esta herramienta cuenta con un repositorio en donde se almacenan las experiencias de los desarrolladores. El repositorio almacena el problema, las actividades a realizar para su solución y la información documental generada. En cada proyecto o nuevo problema, la información almacenada se puede adaptar completamente, permitiendo de esta

manera que la información se incremente y los proyectos nuevos no sufran atrasos o inconvenientes.

Una vez que se ha creado un proyecto, aquellos miembros del equipo que tengan permisos podrán modificar las tareas a realizar, borrarlas o crear otras nuevas. Se pide al encargado de realizar estos cambios explicar la razón de estas decisiones, es decir, si se desea realizar algún cambio, este debe estar completamente justificado. Una vez justificada esta decisión y haber comprobado los beneficios de la misma, los cambios se aprueban y quedan registrados en el sistema.

Esta herramienta maneja ideas muy similares a KUALI-BEH, sin embargo existen discrepancias muy marcadas entre los conceptos manejados por cada uno, como lo son las prácticas y métodos manejados por KUALI-BEH, por dar un ejemplo. Aunque BORE maneja una amplia funcionalidad de gestión de conocimiento, no tiene una visión del proyecto como un todo.

No se han encontrado seguimientos a la herramienta BORE, la información recopilada data de alrededor del año 2003, sin embargo por su similitud con esta tesis se consideró de interés anexarlo para su conocimiento.

Eclipse Process Framework Composer (EPF Composer)

Eclipse Process Framework Composer (EPF Composer) [23], es una herramienta gratuita desarrollada dentro del entorno ECLIPSE, que sirve para editar fragmentos de métodos, procesos o metodologías, y generar automáticamente la documentación adecuada en formato para la web. EPF Composer es un editor de procesos SPEM 2, que incluye características particulares que permiten publicar sitios web.

EPF Composer fue creado para ser utilizado por todos aquellos grupos de trabajo que son responsables de mantener y llevar a cabo procesos en organizaciones dedicadas al desarrollo de software o incluso para proyectos individuales.

EPF Composer tiene dos propósitos principales:

- Proporcionar a los desarrolladores una base de conocimiento que les permita administrar y desplegar su contenido. Esta base de conocimiento puede ser usada como referencia para los procesos de desarrollo futuro.
- Proveer aptitudes de ingeniería de procesos para la selección y adecuación de procesos para proyectos de desarrollo específicos.

EPF Composer pretende dar solución a algunos de los problemas que presentan los líderes y equipos de desarrollo como por ejemplo:

- Los equipos de desarrollo necesitan tener acceso a la información manejada en los proyectos en todo momento.
- Se necesita que los contenidos de los procesos de desarrollo estén en formatos estándar que permitan una fácil integración.
- Se necesita una base de conocimiento para que los grupos de desarrollo aprendan sobre las prácticas que emplean en la resolución de sus problemas.
- Se necesita soporte para que los equipos de desarrollo puedan medir la magnitud de sus procesos de forma correcta.
- Los equipos de desarrollo requieren la habilidad de estandarizar prácticas y procesos dentro de las organizaciones.
- Se necesita emplear términos similares para unir a la ingeniería de procesos, con los procesos empleados en las organizaciones de desarrollo.

Sin embargo, como ya mencionamos anteriormente, todas estas características que distinguen a EPF Composer se las debe a SPEM.

A continuación se dará un breve resumen de lo que es SPEM.

SPEM 2.0

SPEM 2.0 es un metamodelo para ingeniería de procesos y un marco de trabajo conceptual que provee los conceptos necesarios para modelar, documentar, presentar, publicar, gestionar, intercambiar y realizar métodos y procesos de software [22].

Este es un estándar del Object Management Group (OMG), que se encuentra dentro de la ingeniería de procesos, la cual es un área de la Ingeniería de Software dedicada a la “definición, implementación, medición y mejora de los procesos de Ingeniería de Software”.

SPEM 2.0 define procesos de desarrollo de software. Su trabajo es definir los elementos necesarios para definir estos procesos, sin involucrar características de una disciplina en particular.

SPEM 2.0 maneja 3 elementos básicos que son: rol, producto de trabajo y tarea. La finalidad básicamente consiste en decir quién (rol) realiza qué (tarea) para, a partir de unas entradas (productos de trabajo), obtener unas salidas (productos de trabajo).

Al trabajar con SPEM 2.0 existen 4 escenarios fundamentales:

- Crear un repositorio de “Contenidos de método” reutilizables, es decir, una colección organizada de roles, tareas, productos de trabajo, guías, fragmentos de método y procesos, etc.
- Dar soporte al desarrollo, gestión y crecimiento de procesos software. Esto implica combinar, reutilizar y extender los elementos de métodos anteriores para configurar los procesos que sirven para guiar los proyectos.
- Establecer un marco de trabajo general de la organización a partir de los procesos y los elementos definidos anteriormente. Para ello, SPEM 2.0 permite dar soporte al despliegue del contenido de método y proceso que

justo se necesita en cada caso, teniendo en cuenta que ningún proyecto es exactamente igual. SPEM 2.0 incorpora conceptos para:

- Reutilización de procesos o patrones de procesos.
- Variabilidad, procesos que incluyen partes alternativas configurables.
- Particularización, los usuarios definen sus propias extensiones, omisiones y puntos de variabilidad sobre procesos estándares reutilizados.

- Generar plantillas para planes de proyecto concretos. Con el objetivo de darles auténtico valor, las definiciones de los procesos deben ser desplegadas en formatos que permitan su realización automática (sistemas de gestión de proyectos y recursos, motores de flujos de trabajo, etc). Para ello, SPEM 2.0 incluye estructuras de definición de procesos que permiten expresar cómo un proceso será realizado de forma automática con estos sistemas.

ELEMENTOS BASICOS DE SPEM 2.0

En SPEM 2.0 se distinguen dos grupos fundamentales: Method Content y Process. Ver figura 3.

Method Content (Contenido de Método). En este grupo se incluyen los elementos primarios o constructores básicos (Content Elements). Su objetivo es organizar distintos pasos para definir una tarea y especificar cuáles son los productos de trabajo de entrada-salida de cada una de ellas y marcar quién realizará dicha tarea.

Process (Procesos). Se combinan y reutilizan los elementos primarios para dar como resultado los procesos.

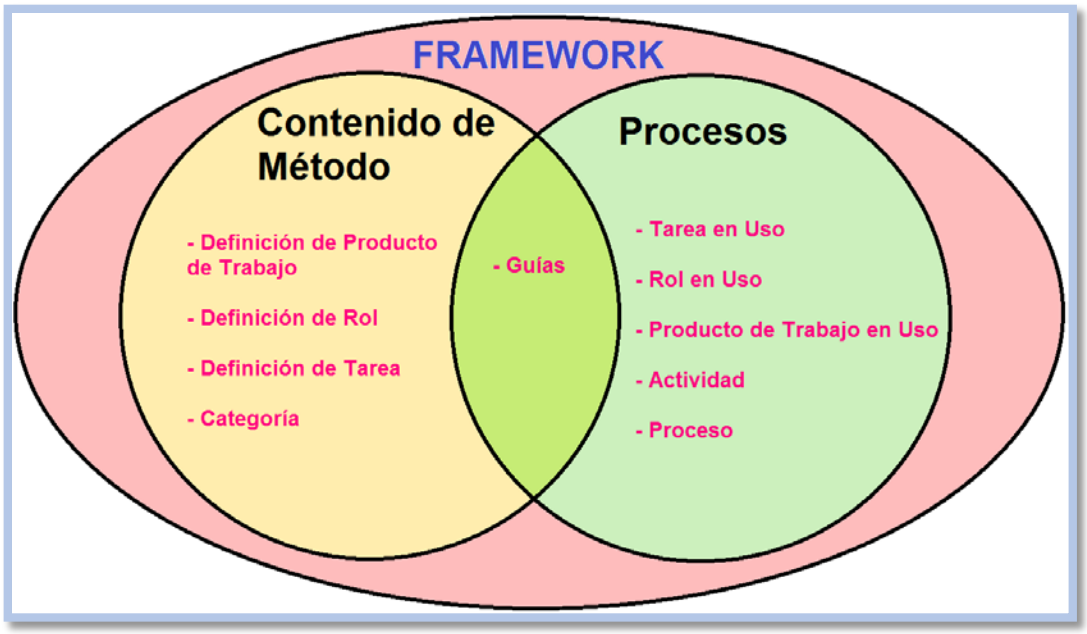


Figura 3. Elementos Básicos y Procesos SPEM

ESTRUCTURA DE UN REPOSITORIO EN SPEM 2.0.

Un repositorio o biblioteca de métodos y procesos en SPEM 2.0 se distribuye en una colección de uno o más plugins y una o más configuraciones. Un ejemplo de toda esta estructura se muestra en la figura 4.

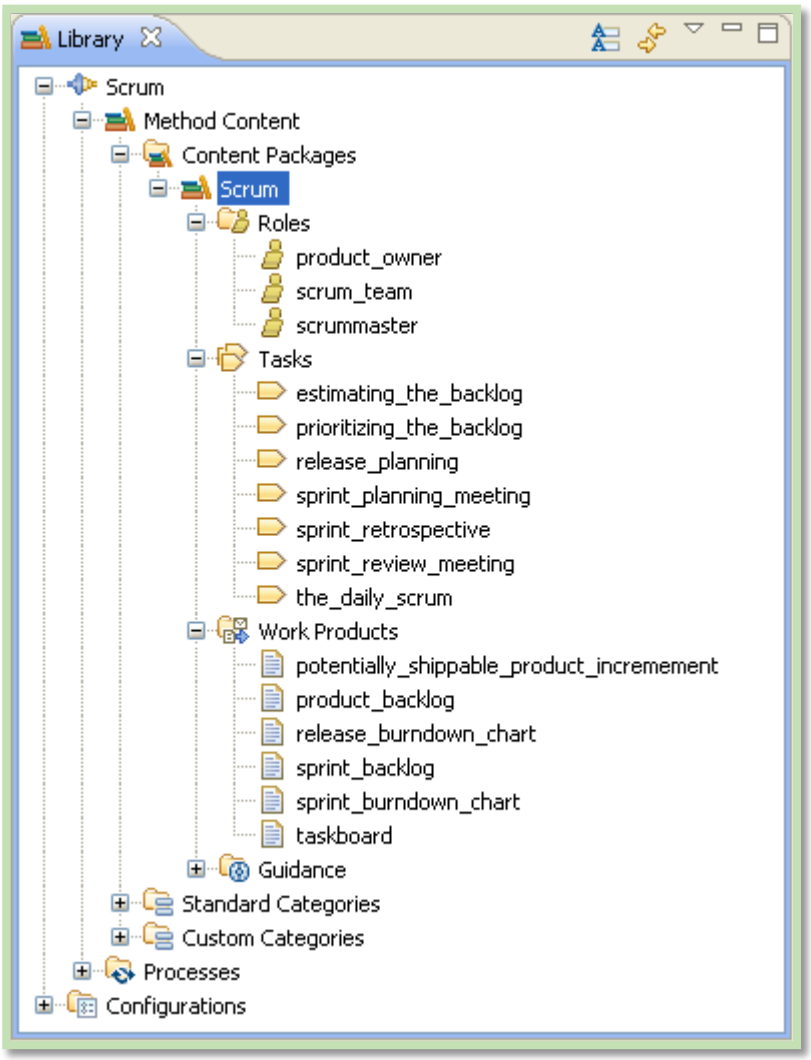


Figura 4. Organización de un repositorio SPEM 2.0

Como se puede observar SPEM 2.0 y KUALI-BEH 1.1 son muy similares, a continuación se presenta una tabla comparativa de manera general que describe las semejanzas y discrepancias entre los distintos conceptos manejados por cada uno.

COMPARACIÓN DE SPEM 2.0 y KUALI-BEH 1.1

La tabla 2 muestra una breve comparativa entre SPEM 2 [22] y KUALI-BEH 1.1 [6]

La comparación se realiza en base al nivel de abstracción y al objetivo de cada uno de los conceptos que se manejan, así como al producto recibido o arrojado por la ejecución de cada uno de los conceptos.

Tabla 2. Comparación SPEM 2.0 y KUALI-BEH 1.1.

SPEM 2.0	KUALI-BEH 1.1	Similitudes Encontradas	Diferencias Identificadas
Activity	Activity	Son prácticamente parecidas en el nivel de abstracción	
Artifact	Work Product	Son conceptos similares en ambos casos	
Deliverable	Result	Ambos son producto de la ejecución de una Tarea en el caso de SPEM y de una Actividad, Práctica o Método en el caso de KUALI-BEH.	Todos los entregables caben dentro del concepto de resultado, pero no al revés.
Deliverable	Work Product	Ambos son producto de la ejecución de una Tarea en el caso de SPEM y de una Actividad, Práctica o Método en el caso de KUALI-BEH.	Un producto de trabajo puede ser o no un entregable
Deliverable	Software Product	Ambos son	

		entregados al cliente	
Guidance	Guide	Son conceptos similares	
Milestone	Objective/Purpose	Son conceptos similares	KUALI-BEH utiliza dos términos para marcar diferencia entre los conceptos del método y los de la práctica
Metric	Measure	Son conceptos similares	
Outcome	Result		Un Result puede ser tangible o intangible. Result es intangible.
Role Definition	Knowledge and Skills	Ambos definen un conjunto de habilidades, competencias y responsabilidades de un individuo	Los roles definen a un tipo de personas. Los Conocimientos y Habilidades abarcan a los roles y dan mayor flexibilidad
Step	Task	Ambos definen la acción más pequeña realizada.	
Task Definition	Practice	Similares en el nivel de abstracción	
Tool Definition	Tool	Son conceptos muy similares haciendo referencia a las herramientas que se utilizan.	
Work Product	Input/Result	Ambos son consumo	

		o producto de la ejecución de una Tarea en el caso de SPEM y de una Actividad, Práctica o Método en el caso de KUALI-BEH.	
Method Library	Methods and Practices Infrastructure		La Infraestructura de Métodos y Prácticas puede contener la Biblioteca Método y más elementos.

Con base a esta tabla podemos concluir que además de abarcar los conceptos manejados por KUALI-BEH, el Entorno Computacional para KUALI-BEH también logra abarcar los conceptos manejados por SPEM.

La tabla 3 es una tabla comparativa resultado del análisis de todas las herramientas estudiadas en este trabajo.

Tabla 3. Comparación de Herramientas.

	Gestión formal del Conocimiento y Experiencias Adquiridas	Reutilización del Conocimiento y Experiencias Adquiridas	Desglose del proyecto en tareas mínimas	Gestión del Proyecto	Simplicidad de Uso
CodeBeamer	NO	NO	NO	SI	MEDIO

IRIS Process Author	NO	NO	NO	SI	MEDIO
Microsoft Visual Studio Team System	NO	NO	NO	SI	MEDIO
Building an Organization Repository of Experiences (BORE)	MEDIO	MEDIO	SI	MEDIO	MEDIO
Eclipse Process Framework Composer (EPF Composer)	SI	SI	SI	SI	MEDIO
Repositorio de métodos y prácticas de proyectos de software para KUALI-BEH.	SI	SI	SI	Si	SI

2.3 Proceso de Desarrollo

Para el desarrollo de este proyecto se contó con la colaboración de distintas personas, por lo que fue necesario incluir un modelo de desarrollo de software para elaborar el trabajo. Analizando las necesidades que se tenían y los resultados que se deseaba obtener, se decidió por una metodología ágil.

A continuación se describirá lo que es una metodología ágil y lo que es Scrum [13], que fue la metodología ágil en la cual se basó la forma de trabajo para este proyecto.

2.3.1 Metodología Ágil

Los métodos ágiles son aquellos métodos de Ingeniería de Software que apoyan el desarrollo iterativo e incremental. Estas metodologías requieren la participación de pequeños grupos de trabajo activos y auto-organizados.

Los métodos ágiles están descubriendo nuevas formas de desarrollar software basándose en la valoración entre los diferentes conceptos involucrados en el desarrollo, la cual queda plasmada en el denominado manifiesto ágil:

*"**Individuos e interacciones** sobre procesos y herramientas*
***Software funcionando** sobre documentación extensiva*
***Colaboración con el cliente** sobre negociación contractual*
***Respuesta ante el cambio** sobre seguir un plan*

Esto es, aunque valoramos los elementos de la derecha,
valoramos más los de la izquierda." [18]

Los métodos ágiles ponen como prioridad la satisfacción del cliente mediante la entrega temprana y continua de software, minimizando riesgos y desarrollando porciones pequeñas de software en lapsos cortos, aproximadamente de una a cuatro semanas. Cada lapso de tiempo, es llamado iteración y al tener como resultado la entrega de software, cada iteración incluye la planificación, análisis de requerimientos, diseño, codificación, revisión y documentación. El equipo de trabajo realiza todas las iteraciones necesarias hasta completar el producto de software final.

Entre las metodologías ágiles más destacadas hasta el momento podemos mencionar:

- XP – Extreme Programming.
- Scrum.
- Crystal Clear.
- DSDM – Dynamic Systems Developmemt Method.

- FDD – Feature Driven Development.
- ASD – Adaptive Software Development.
- XBreed.
- Extreme Modeling.

2.3.1.1 Scrum

Scrum [27] es un marco de gestión para el desarrollo incremental de productos, valiéndose de uno o más equipos multi-funcionales, autoorganizados, de aproximadamente siete personas cada uno. Proporciona una estructura de roles y prácticas. Los roles principales en Scrum son:

ScrumMaster o facilitador, se asegura que los procesos de desarrollo se realicen de forma correcta. No puede considerarse como el líder del equipo, ya que ellos se auto-organizan.

ProductOwner, representa la voz del cliente. Se asegura que el equipo trabaje de forma adecuada desde la perspectiva del negocio.

Stakeholders, son todas las personas interesadas en el proyecto ya sean internas o externas y a las cuales el proyecto les producirá algún beneficio.

Equipo de Desarrollo (Team), es el responsable de realizar y entregar el producto, tiene un total de 7 ± 2 miembros.

Scrum [13] es una metodología ágil en la que se trabaja en ciclos llamados Sprints, los cuales duran de dos a cuatro semanas. En cada Sprint se priorizan los requerimientos del cliente y se trabaja sobre aquellos con el valor más alto. Al final de cada Sprint el equipo de trabajo deberá entregar un software pequeño pero funcional.

El conjunto de características que forma parte de cada sprint viene del Product Backlog, que es un conjunto de requisitos de alto nivel priorizados que definen el trabajo a realizar. Los elementos del Product Backlog que forman parte del sprint se determinan durante la reunión de Sprint Planning. Durante esta reunión, el Product Owner identifica los elementos del Product Backlog que quiere ver completados y los hace del conocimiento del equipo. El equipo determina la cantidad de ese trabajo que puede comprometerse a realizar durante el siguiente sprint. Durante el sprint en curso nadie puede cambiar el Sprint Backlog, lo que significa que los requisitos están congelados durante ese sprint.

Scrum no describe qué hacer en cada proyecto, solo ofrece un marco de trabajo y un conjunto de prácticas que mantienen todo visible y guían los esfuerzos para obtener los mejores resultados posibles. Scrum es utilizado para trabajos complejos en el que no es posible predecir todo lo que ocurrirá, incluso esta metodología permite realizar cambios en cualquier parte del ciclo de vida del proyecto.

Esta metodología permite la adaptación, inspección continúa y propicia la innovación, incrementa el compromiso, el entusiasmo, la creatividad, por esta razón los resultados aumentan su valor y se disminuye el esfuerzo requerido en cada proceso. Esto puede producir un producto útil para el cliente y satisfacción por el trabajo.

3. DESARROLLO DEL REPOSITORIO

3.1 Objetivos y Alcances

Este proyecto de tesis tiene como objetivo crear una herramienta de repositorio que permita la captura, consulta y utilización de los conceptos manejados en la Vista Estática de KUALI-BEH [6], de tal forma que los practicantes de software tengan la posibilidad de almacenar sus conocimientos, gestionarlos y posteriormente poder reutilizarlos e incluso adaptarlos y actualizarlos. Estos conocimientos deberán ser modelados como Métodos y Prácticas de acuerdo al proyecto KUALI-BEH.

El repositorio será un sistema web, que permita el acceso a los practicantes desde cualquier equipo con acceso a internet, dicho sistema se alojará en un servidor y contará con una base de datos relacional, en la cual se almacenará toda la información registrada.

Los practicantes de la organización tendrán la posibilidad de dar de alta un catálogo de conocimientos, así como poder modificarlos y consultarlos en el momento que lo deseen, este conjunto de catálogos serán los siguientes:

- Knowledge and Skills (Conocimientos y Habilidades)
  Conjunto de habilidades, competencias y logros, adquirida por el profesional y necesarios para realizar una práctica.
- Tools (Herramientas)

Son los dispositivos utilizados para llevar a cabo una función particular.

- Tasks (Tareas)
  Son los requisitos, recomendaciones o acciones permisibles.
- Types (Tipos)
  Utilizado para clasificar las herramientas en tipos diferentes.
- Verification Criteria (Criterios de Verificación)
  Son los criterios que se deberán de considerar para verificar una práctica.
- Measures (Métricas)
  Medida o conjunto de medidas destinadas a conocer o estimar el tamaño o característica de una práctica.
- Work Product (Productos de Trabajo)
  Es un artefacto utilizado o generado por una práctica. Podría tener un estado asociado.
- Conditions (Condiciones)
  Situación, circunstancia o estado específico de algo o de alguien con respecto a la apariencia, la aptitud o a la orden de trabajo que tienen relación con el proyecto de software.
- Characteristics (Características)
  Características utilizadas para clasificar productos de trabajo y/o condiciones.

El sistema también permite la creación, consulta y modificación de diversas actividades, las cuales tendrán asociadas los catálogos creados previamente, permitiendo modificar en cualquier momento la relación entre estos.

Los practicantes podrán almacenar, modificar, versionar y consultar las prácticas necesarias para cualquier proyecto. Cada práctica contendrá un conjunto de guías asociadas, las cuales de la misma manera podrán ser almacenadas, modificadas, versionadas y consultadas según se requiera. Cada una de estas guías a su vez tendrá un conjunto de actividades. El conjunto de catálogos, actividades y guías darán vida a una nueva práctica.

Los métodos en KUALI-BEH, son un conjunto de prácticas que en conjunto tienden a cumplir un objetivo común. El sistema de repositorio permitirá la asociación de las distintas prácticas contenidas en el sistema para poder crear un método o conjunto de métodos.

3.2 Especificación de Requerimientos

Para mostrar el comportamiento del sistema de forma gráfica, se crearon diferentes diagramas de caso de uso, los cuales muestran de manera general y específica todo el funcionamiento del sistema. En la figura 5 se puede observar el diagrama general de casos de uso del repositorio y dentro del Apéndice A de esta tesis se encuentran plasmados todos los diagramas específicos y su descripción.

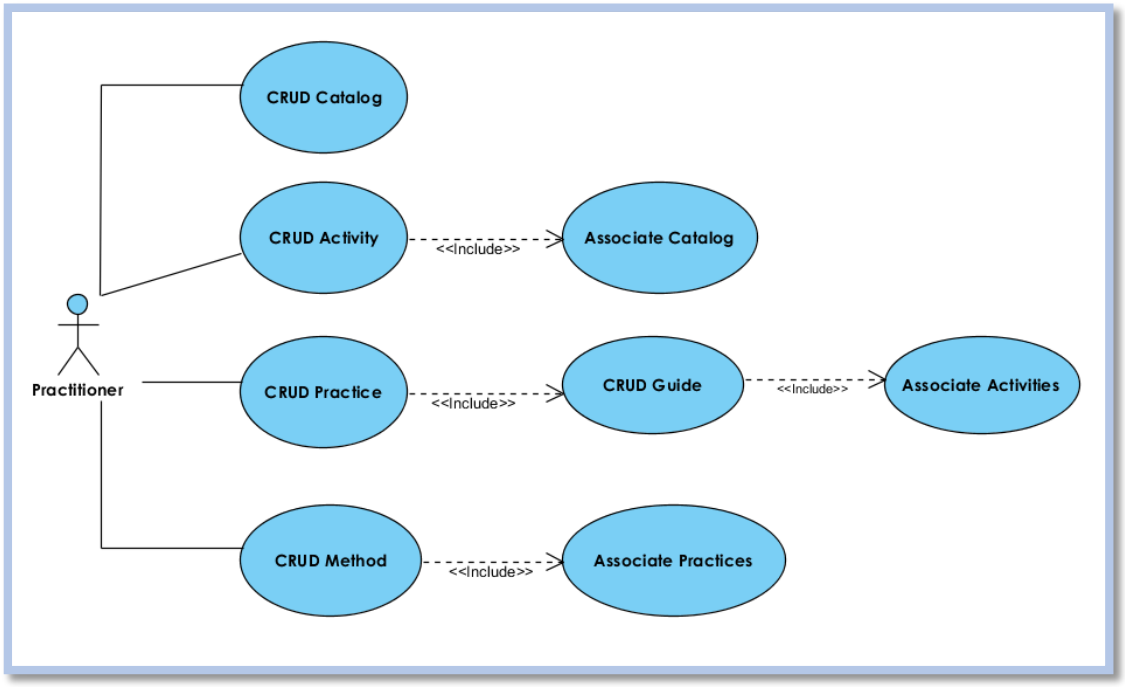


Figura 5. Diagrama General de Casos de Uso del Repositorio

Después de haber plasmado en los diagramas de casos de uso el comportamiento del sistema, se procedió con la creación de los prototipos de la interfaz, se sometieron a voto los colores principales y se creó el logo que portaría el sistema. La figura 6 presenta el menú principal, en el cual se observan los botones Method, Practice, Activities y Catalog, con los cuales se podrá navegar a través del sistema y hacer uso del mismo. En el Apéndice A se muestran todas las interfaces con las cuales cuenta el sistema terminado, así como una breve descripción de las mismas.

Figura 6. Menú Principal

Se proyecta que el sistema se aloje en un host, los usuarios podrán tener acceso a él desde cualquier equipo que cuente con un navegador web, y que tenga una correcta conexión a internet, incluso podrá instalarse dentro de una red LAN dentro de la organización, para poder acceder a él por medio de un navegador web independientemente del sistema operativo de los usuarios.

3.3 Arquitectura del Repositorio

Como arquitectura general del proyecto Entorno Computacional para KUALI-BEH, se optó por emplear Modelo Vista Controlador (MVC), la cual se detalla en la tesis Arquitectura de software para el entorno computacional de KUALI-BEH [10], misma arquitectura que se sigue en este proyecto de tesis.

Las características principales por las que se decidió utilizar la Arquitectura Modelo Vista Controlador se describen a continuación.

3.3.1 Modelo Vista Controlador

Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa la interfaz de usuario y la lógica de negocio en tres componentes distintos: el Modelo, la Vista y el Controlador. El objetivo principal para separar estos conceptos es impulsar de manera significativa la reutilización de código, esto en gran medida facilita el desarrollo de las aplicaciones y el mantenimiento de las mismas.

El Modelo Vista Controlador es muy utilizado hoy en día en proyectos de desarrollo de sistemas web por facilitar y proveer de una correcta distribución de sus componentes a estas aplicaciones.

A continuación se describen cada uno de los componen que conforman este patrón:

Modelo.

Esta es la representación específica de la información con la cual el sistema opera.

Es el encargado de administrar la lógica de la aplicación (lógica de negocio). Su finalidad es servir de abstracción de algún proceso del mundo real, tiene acceso a la base de datos y tiene funciones específicas que controlan la integridad del sistema.

Vista.

Muestra al usuario la información con la que se está trabajando. Pueden existir múltiples vistas del modelo. Sencillamente es la representación visual del modelo y la encargada de representar los componentes visuales en la pantalla.

Controlador.

Es el escuchador de los eventos que genera el usuario, es decir, es el que permite que interactúe el usuario con el sistema. Interpreta los eventos (las entradas) a través del teclado y/o ratón.

La figura 7 muestra el diagrama general del Modelo Vista Controlador.

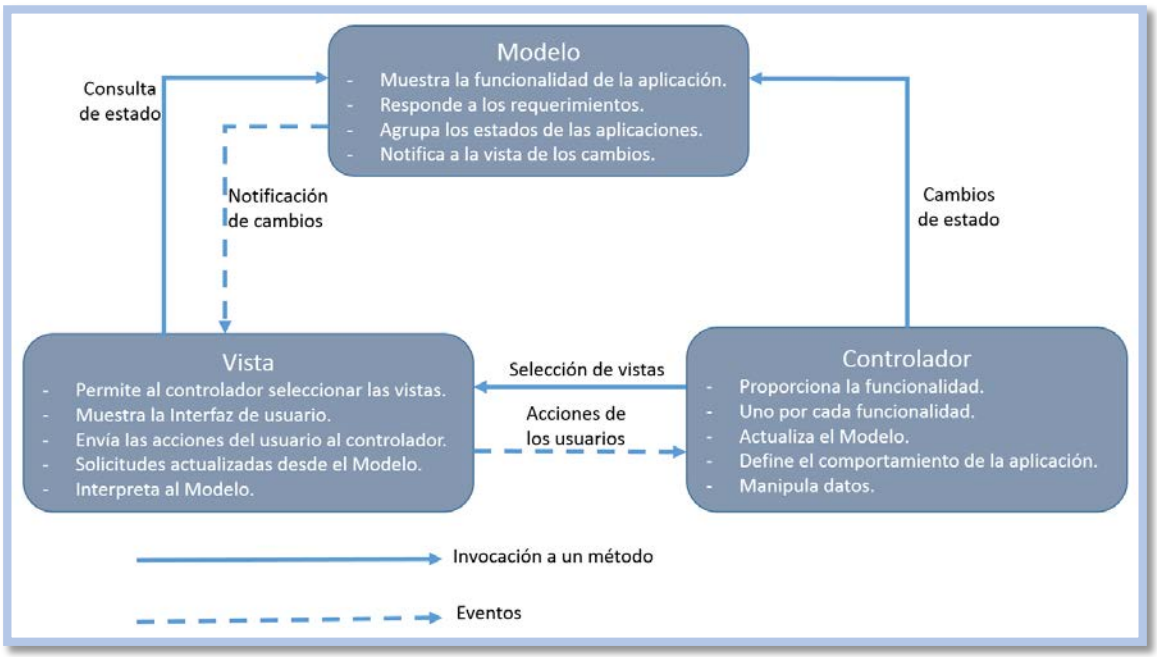


Figura 7. Diagrama, Patrón de Diseño. Modelo Vista Controlador

Se acordó el uso de este patrón de arquitectura para el desarrollo de este sistema porque logra cubrir y satisfacer todas las necesidades y requerimientos del proyecto.

Una vez acordado el uso de este patrón, se procedió a realizar los diagramas de clases, secuencia y paquetes correspondientes.

Las figuras 8, 9 y 10 muestran los diagramas de paquetes del Entorno Computacional para KUALI-BEH, en el cual se aprecia de forma general la

distribución de los componentes, estos diagramas fueron tomados de la tesis: Arquitectura de software para el entorno computacional de KUALI-BEH [10].

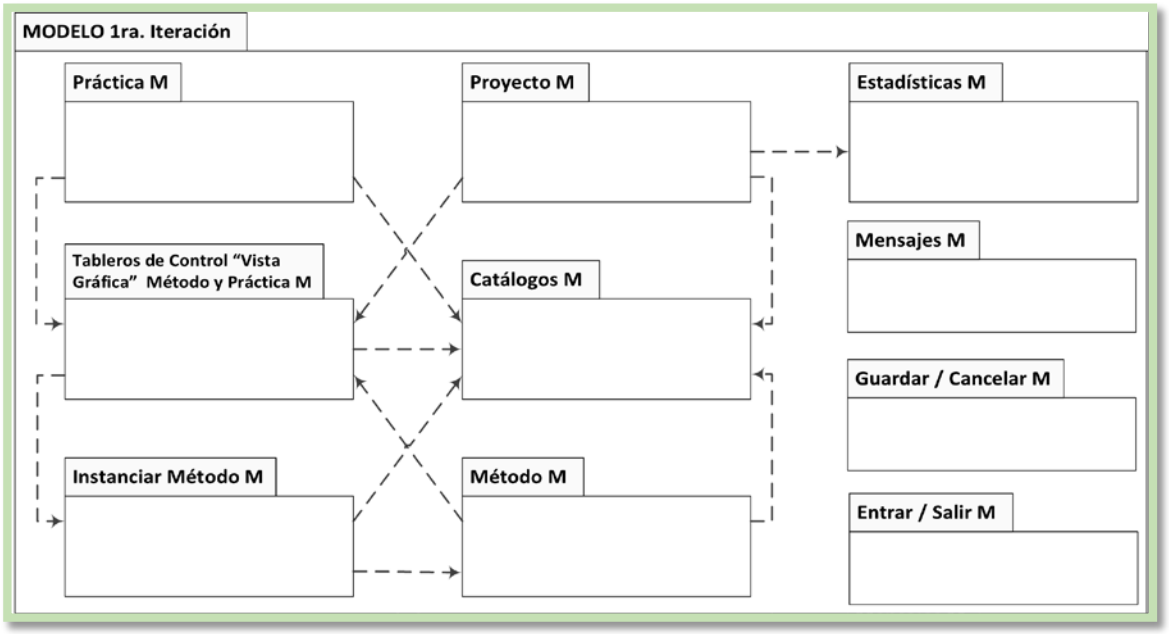


Figura 8. Modelo 1a Iteración [10]

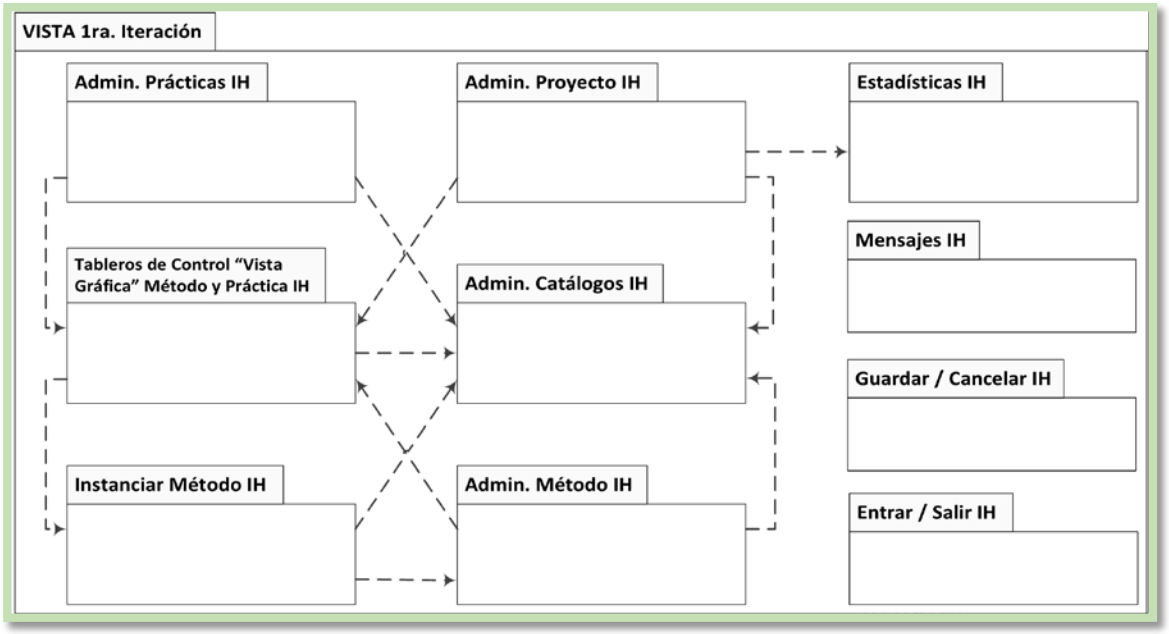


Figura 9. Vista 1a Iteración [10]

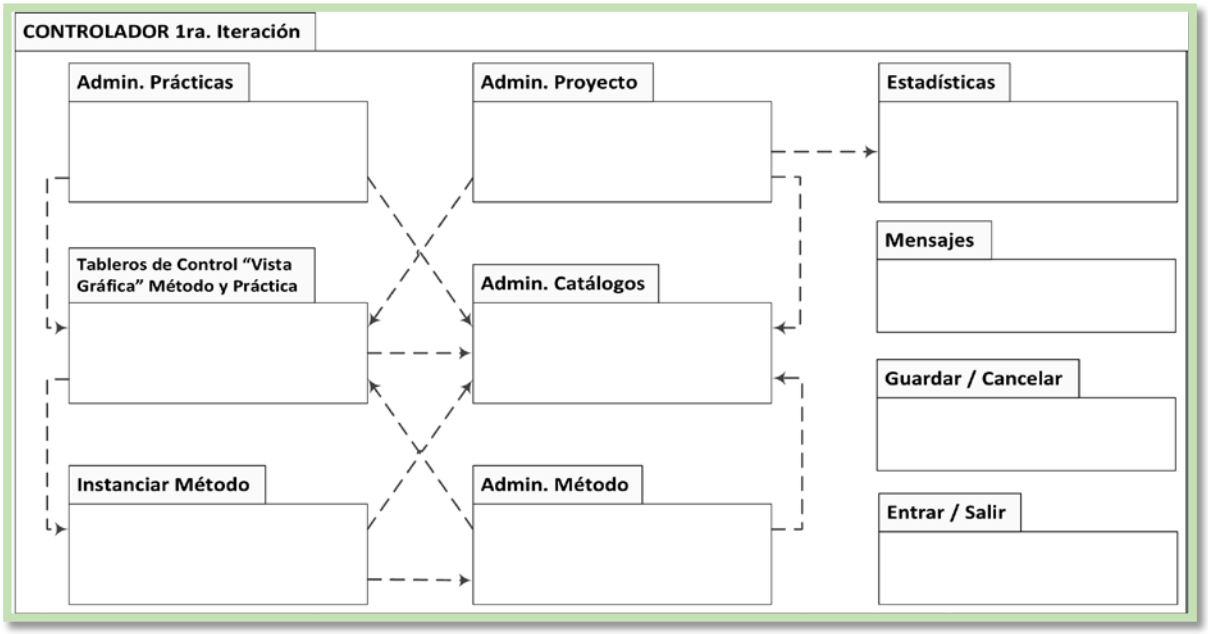


Figura 10. Controlador 1a Iteración [10]

El Apéndice B muestra estos diagramas de clases y diagramas de secuencia de forma detallada.

3.4 Herramientas de Desarrollo

En este apartado se presentan las herramientas que se utilizaron en el desarrollo de este proyecto.

3.4.1 Java

Para la creación de la herramienta de repositorio, se acordó utilizar Java como lenguaje de programación principal, por ser una tecnología, viable, rápida, segura y fiable, además de ser gratuita y logra cumplir con los requerimientos del

sistema, siendo un lenguaje multiplataforma, podrá ser instalado sin ningún problema bajo cualquier sistema operativo

Java tiene un gran valor para los desarrolladores, ya que permite:

- Escribir software en una plataforma y ejecutarla virtualmente en otra
- Crear programas que se puedan ejecutar en un explorador y acceder a servicios Web disponibles
- Desarrollar aplicaciones de servidor para foros en línea, almacenes, encuestas, procesamiento de formularios HTML y mucho más
- Combinar aplicaciones o servicios que utilizan el lenguaje Java para crear aplicaciones o servicios con un gran nivel de personalización
- Escribir aplicaciones potentes y eficaces para teléfonos móviles, procesadores remotos, productos de consumo y prácticamente cualquier otro dispositivo electrónico [25].

Como entorno de desarrollo se decidió por utilizar NetBeans 7.3, por ser un entorno de desarrollo libre y gratuito sin restricciones de uso, hecho principalmente para el lenguaje de programación Java.

NetBeans [19] es un entorno de desarrollo muy completo y profesional. Contiene muchas funcionalidades para distintos tipos de aplicaciones y facilita en gran medida la programación, la prueba y la depuración de las aplicaciones que se desarrollan.

3.4.2. PostgreSQL

Para la construcción de la base de datos se acordó el empleo de una base de datos relacional, por el motivo de que esta logra cubrir con los requerimientos expuestos al inicio del proyecto.

Como sistema gestor de base de datos para la realización del repositorio, se decidió por utilizar PostgreSQL, en su versión 9.2.

PostgreSQL [20] es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD[1] y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado a la fecha.

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos, esto con el objetivo de garantizar la estabilidad del sistema. Lo anterior permite que un fallo en uno de los procesos no afecte al resto y el sistema continúe funcionando.

La figura 11 ilustra de manera general los componentes más importantes en un sistema PostgreSQL.

[1] La licencia BSD es la licencia de software otorgada principalmente para los sistemas BSD (*Berkeley Software Distribution*).

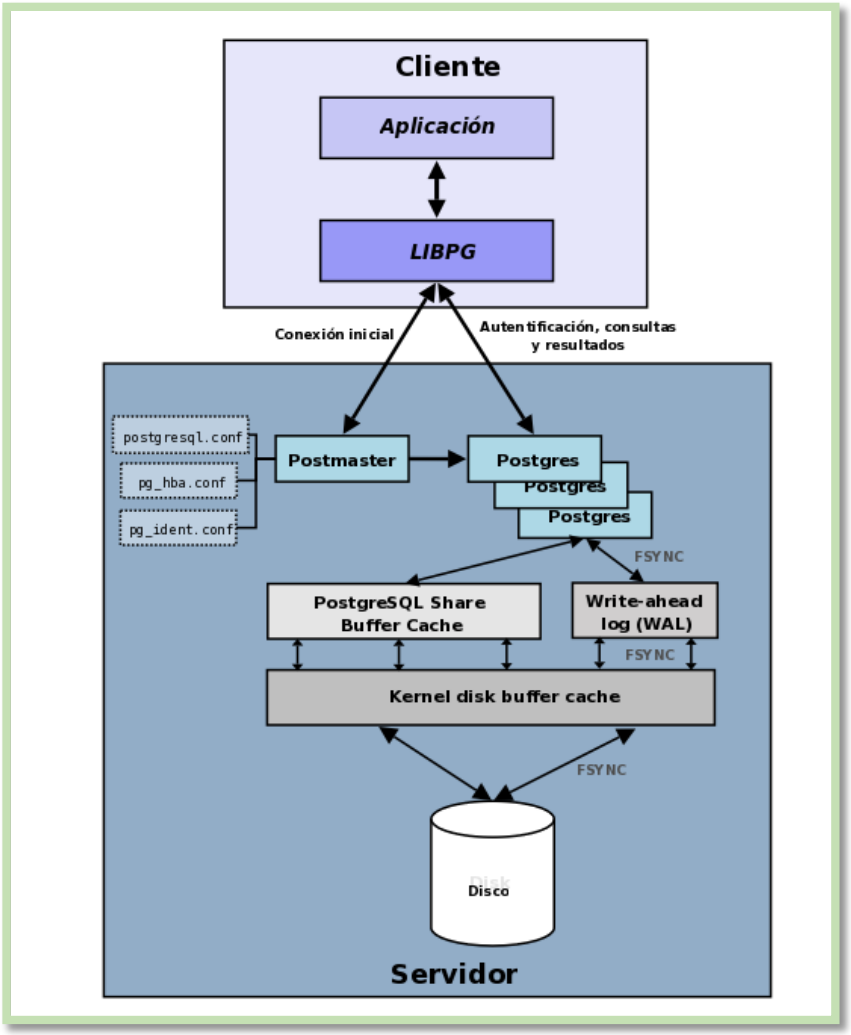


Figura 11. Componentes más importantes en un sistema PostgreSQL (Imagen obtenida de [20]).

- **Aplicación cliente:** Esta es la aplicación cliente que utiliza PostgreSQL como administrador de bases de datos. La conexión puede ocurrir via TCP/IP ó sockets locales.
- **Demonio postmaster:** Este es el proceso principal de PostgreSQL. Es el encargado de escuchar por un puerto/socket por conexiones entrantes de clientes. Tambien es el encargado de crear los procesos hijos que se encargaran de autentificar estas peticiones, gestionar las consultas y mandar los resultados a las aplicaciones clientes
- **Archivos de configuracion:** Los 3 archivos principales de configuración utilizados por PostgreSQL, postgresql.conf, pg_hba.conf y pg_ident.conf

- **Procesos hijos postgres:** Procesos hijos que se encargan de autentificar a los clientes, de gestionar las consultas y mandar los resultados a las aplicaciones clientes
- **PostgreSQL share buffer cache:** Memoria compartida usada por PostgreSQL para almacenar datos en caché.
- **Write-Ahead Log (WAL):** Componente del sistema encargado de asegurar la integridad de los datos (recuperación de tipo REDO)
- **Kernel disk buffer cache:** Caché de disco del sistema operativo
- **Disco:** Disco físico donde se almacenan los datos y toda la información necesaria para que PostgreSQL funcione [20].

Dentro del Apéndice B se encuentra de manera detallada el diagrama entidad relación de la vista estática de KUALI-BEH, el cual es utilizado en el desarrollo de este repositorio.

3.4.3 Frameworks

Para facilitar la codificación del proyecto Entorno Computacional para KUALI-BEH, se implementará el uso de frameworks, específicamente Hibernate y JSF.

Un Framework (marco de trabajo) en informática, se define como un conjunto estandarizado de conceptos, prácticas y criterios que se enfoca en resolver un problema particular y que posteriormente servirá como referencia para enfrentar y resolver nuevos problemas similares en el desarrollo de software. Comúnmente puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Los Frameworks están diseñados para acelerar el proceso de desarrollo de software, poder reutilizar código ya existente de forma eficiente y promover buenas prácticas de programación como el uso de patrones, lo que permite a los

programadores y diseñadores ocupar más tiempo identificando requerimientos de software y no malgastar su tiempo resolviendo problemas que ya se han solucionado en proyectos anteriores.

En Java se reconocen Framework como: Spring, Struts, JSF, Hibernate, entre otros que se han venido creando y que han llamado la atención de los desarrolladores alrededor mundo. Sin embargo cabe destacar que aunque el principal objetivo de los Frameworks es facilitar la vida de los desarrolladores, se requiere dedicar un tiempo considerable a su aprendizaje por parte de los programadores principiantes; pero después de que estos se adaptan a su utilización, la herramienta les permitirá el poder optimizar la lógica de programación de forma sencilla, rápida y eficaz.

3.4.3.1 Hibernate

Hibernate es una herramienta de mapeo objeto-relacional (ORM) para la plataforma Java (y disponible también para .Net con el nombre de NHibernate) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones.

Hibernate es software libre, distribuido bajo los términos de la licencia GNU LGPL. Es un framework que agiliza la relación entre la aplicación y la base de datos de forma rápida y relativamente sencilla.

La figura 12 brinda una perspectiva a alto nivel de la Arquitectura de Hibernate.

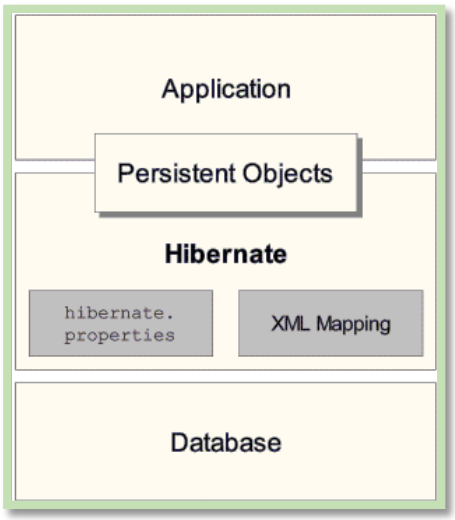


Figura 12. Arquitectura Hibernate.

3.4.3.2 Java Server Faces (JSF)

Por las necesidades con las que se contaba para interactuar con la herramienta, se investigaron diversas tecnologías que permitieran crear una interfaz de forma práctica pero eficiente y que al mismo tiempo respetara la arquitectura MVC previamente planteada para la realización de la herramienta.

Se optó por la utilización de un framework, y el que más se adecuaba a cumplir con las necesidades fue Java Server Faces.

Java Server Faces (JSF) es una tecnología y framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones java J2EE basadas en el patrón MVC. JSF usa Java Server Pages (JSP) como la tecnología que permite hacer el despliegue de las páginas.

JSF contiene algunas extensiones como son:

- **RichFaces** Agrega componentes visuales y soporte para AJAX.

- **ICEfaces** Contiene diversos componentes para interfaces de usuarios enriquecidos, tales como editores de texto, reproductores multimedia, entre otros.
- **jQuery4jsf** Contiene diversos componentes sobre la base de javascript jQuery.
- **OpenFaces** Librería open source que contiene diferentes componentes JSF, un Framework Ajax y un Framework de validación por parte del cliente.
- **PrimeFaces** Es una biblioteca muy liviana, todas sus componentes están basados en mantener a PrimeFaces lo más liviano posible. PrimeFaces es una librería muy simple que no necesita dependencias y configuraciones.

Analizando cada una de estas extensiones, se decidió la implementación de PrimeFaces [21] (versión 3.5), por sus características y componentes existentes.

3.4.3.2.1 PrimeFaces

PrimeFaces [21] es una librería de componentes visuales open source desarrollada y mantenida por Prime Technology, una compañía Turca de IT especializada en consultoría ágil, JSF, Java EE y Outsourcing

Las principales características de PrimeFaces son:

- Soporte nativo de Ajax.
- Kit para crear aplicaciones web móviles.
- Es compatible con otras librerías de componentes como RichFaces.
- Uso de javascript no intrusivo (no aparece en línea dentro de los elementos, sino dentro de un bloque <script>).
- Es un proyecto open source, activo y bastante estable entre versiones.
- Gran número de componentes.

- Facilidad de instalación, sin necesidad de configuración.
- Temas prediseñados.
- Extensa documentación.
- Rendimiento. Todos sus componentes están pensados para mantener la aplicación lo más ligera posible.

La figura 13 muestra la forma en la que interactúan las herramientas de desarrollo que se utilizan en la construcción de este repositorio.

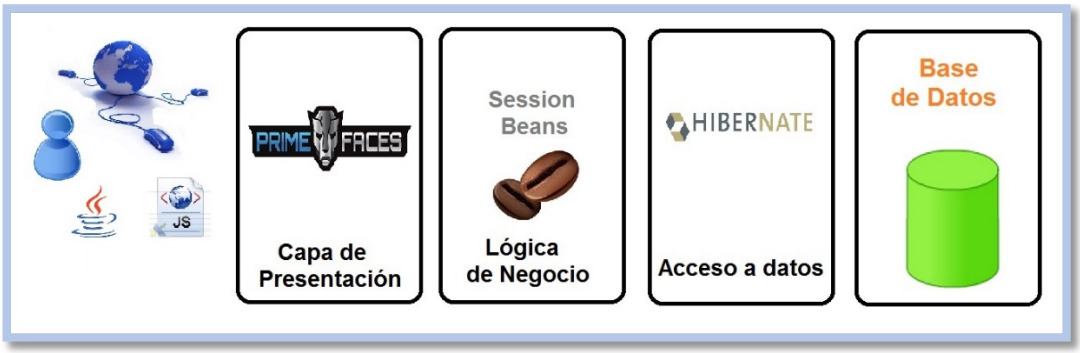


Figura 13. Interacción Herramientas de Desarrollo

3.5 Construcción de Repositorio

Como ya hemos mencionado de manera general, para la construcción de este repositorio, se utilizaron dos Frameworks: Hibernate y JSF en su extensión PrimeFaces. Utilizando Java como lenguaje de programación principal. Se construyó también una base de datos relacional, la cual se implementó y corre bajo el gestor PostgreSQL. La Arquitectura que soporta a este repositorio es el Modelo Vista Controlador.

La figura 14 muestra el Diagrama Entidad Relación correspondiente a la Vista Estática de KUALI-BEH, la cual es utilizada para la realización de este repositorio.

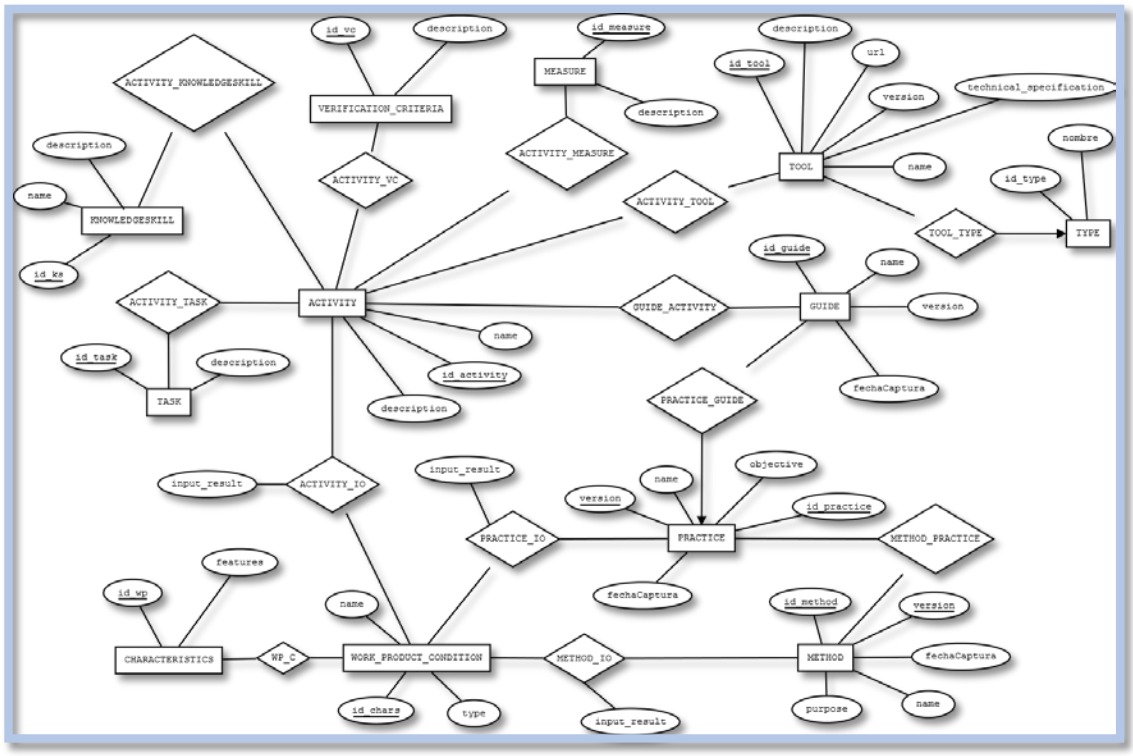


Figura 14. Diagrama Entidad Relación de la Vista Estática de KUALI-BEH

Los detalles de la construcción de este proyecto se encuentran plasmados en los Apéndices A, B y C de esta Tesis.

3.6 Pruebas

Se realizaron pruebas de funcionalidad y de usabilidad con el propósito de verificar y validar que se cumplieran los requerimientos tanto funcionales como no funcionales del sistema, así como señalar los defectos y aciertos que contenía el mismo en cuanto a facilidad de uso y manejo de la interfaz y corregirlos para dar como resultado, un sistema eficaz, eficiente, seguro y de utilidad para el usuario.

En el Apéndice C se incluyen los detalles de las pruebas realizadas al sistema.

4. CONCLUSIONES Y TRABAJO A FUTURO

El repositorio del entorno computacional para KUALI-BEH, generado en este trabajo, servirá de apoyo a los practicantes de las organizaciones de desarrollo de software.

Proporcionará un marco para la definición de distintas formas de trabajo por parte de los practicantes. Estas formas de trabajo se estructurarán como métodos compuestos por prácticas de acuerdo a la vista estática de KUALI-BEH.

Proveerá de un mecanismo para adoptar y manejar el concepto de prácticas y métodos bajo el enfoque de KUALI-BEH.

Permitirá la implementación de la Vista Operacional de KUALI-BEH, para el control y seguimiento de los proyectos de software.

Durante la realización de este trabajo se estudiaron, analizaron y aplicaron prácticas y métodos de ingeniería de software, se trabajó bajó la metodología ágil SCRUM obteniendo resultados favorables en la organización, comunicación y puesta en marcha del proyecto. De esta forma además de haber cumplido con los requerimientos propuestos desde un inicio, se comprobó que el empleo de las metodologías de desarrollo ágil puede favorecer en ciertos proyectos a un correcto desarrollo y con resultados satisfactorios.

Además de abarcar los conceptos manejados por KUALI-BEH, el proyecto Entorno Computacional para KUALI-BEH también logra abarcar los conceptos manejados por SPEM.

La tesis Arquitectura de software para el entorno computacional de KUALI-BEH se utilizó de apoyo en la Arquitectura de este proyecto, principalmente en la organización de los diagramas de paquetes plasmados en la misma, la cual permitirá que dos proyectos restantes (Selección y adaptación de métodos en proyectos de software aplicando KUALI-BEH y Tableros de control para el seguimiento de proyectos de software aplicando KUALI-BEH) se incorporen y adapten adecuadamente.

5. REFERENCIAS

[1] SEMAT. http://semat.org/wp-content/uploads/2012/03/SEMAT-vision.pdf, 1/8/2013

[2] SEMAT. http://semat.org/wp-content/uploads/2012/03/Final_ICSE_Semat_Presentation.pdf, 1/8/2013

[3] OMG. http://www.omg.org/ 8/04/2013

[4] A Foundation for the Agile Creation and Enactment of Software Engineering Methods, http://www.omg.org/cgi-bin/doc?ad/2011-6-26 16/05/11

[5] OMG. http://www.omg.org/cgi-bin/apps/membersearch.pl, 14/9/2013

[6] KUALI-BEH V1.1, https://docs.google.com/a/kuali-kaans.mx/viewer?a=v&pid=sites&srcid=a3VhbGkta2FhbnMubXh8a3VhbGkta2FhbnN8Z3g6MTYxMmQ2OWUxODJkZDI1YQ, 12/08/2012

[7] Oktaba, H., García, F., Piattini, M., Pino, F., Ruíz, F., Alquicira, C.: Software Process Improvement: The COMPETISOFT Project. IEEE Computer, Vol 40, No. 10 (2008)

[8] Norma mexicana NMX-I-059-NYCE-2005 Modelo de Procesos para la Industria del Software (MoProSoft) (2005)

[9] ISO/IEC 29110-5-1-2 Software engineering -- Lifecycle profiles for Very Small Entities (VSEs) – Management and Engineering Guide: Generic profile group: Basic Profile, http://standards.iso.org/ittf/PubliclyAvailableStandards/c051153_ISO_IEC_29110-5-1-2_2011.zip 07/08/12 (2011)

[10] Alberto Tapia Durán, tesis de maestría en ingeniería: Arquitectura de software para el entorno computacional de KUALI-BEH

[11] Eraím Ruíz Sánchez, tesis de maestría en ingeniería: Selección y adaptación de métodos en proyectos de software aplicando KUALI-BEH

[12] José Luis Urrutia Velázquez, tesis de maestría en ingeniería: Tableros de control para el seguimiento de proyectos de software aplicando KUALI-BEH

[13] Scrum Alliance. http://www.scrumalliance.org/ 19/09/2013

[14] Intland Software. codeBeamer. http://intland.com/products/codebeamer/overview/, 18/4/2013

[15] Osellus. IRIS Process Author. http://www.osellus.com/IRIS-PA, 14/4/2013

[16] Microsoft Corporation. Visual Studio Team System. http://msdn.microsoft.com/es-es/library/fda2bad5(v=vs.90).aspx, 19/4/2013

[17] Universidad Interamericana para el Desarrollo, UNID. http://unid.edu.mx/tecnologiaeinnovacion/967-repositorios-digitales.html, 23/9/2013

[18] Agile Manifesto. http://agilemanifesto.org/iso/es/, 23/9/2013

[19] NetBeans. https://netbeans.org/features/index.html, 19/2/2013

[20] PostgreSQL. http://www.postgresql.org.es/sobre_postgresql, 30/9/2013

[21] PrimeFaces. http://primefaces.org/, 18/8/2013

[22] SPEM 2.0. http://www.omg.org/spec/SPEM/2.0/, 17/11/2013

[23] Eclipse Process Framework Project (EPF). https://www.eclipse.org/epf/, 19/12/2013

[24] Scott Henninger. Turning Development Standards Into Repositories of Experiences. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.198.973&rep=rep1&type=pdf, 19/12/2013

[25] Java. http://www.java.com/es/about/, 19/12/2013

[26] EjemplosTIW. Universidad Carlos III de Madrid. http://www.lab.inf.uc3m.es/~a0080802/RAI/mvc.html. 17/11/2013

[27] La guía de Scrum. https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/2013/Scrum-Guide-ES.pdf, 02/05/2014

APÉNDICE A

ESPECIFICACIÓN DE REQUERIMIENTOS

En éste apartado se describen los requisitos funcionales y no funcionales del sistema, se documentan los casos de uso de forma general y específica y se presentan las interfaces que se mostrarán al usuario.

ÍNDICE

ESPECIFICACIÓN DE REQUERIMIENTOS

1. Introducción

Para brindar un mayor soporte al proyecto KUALI-BEH [6], se vio la necesidad de crear el proyecto Entorno Computacional para KUALI-BEH, el cual apoyará a las organizaciones que desarrollan software en las siguientes actividades:

1. Expresar sus métodos y prácticas empíricas de manera estructurada.
2. Resguardar esos métodos y prácticas en un repositorio de la organización.
3. Seleccionar y adaptar al contexto de un proyecto de desarrollo de software algún método y prácticas del repositorio de la organización.
4. Aplicar el método y sus prácticas durante la ejecución del proyecto, dándole seguimiento a través de tableros de control.
5. Enriquecer y mejorar el repositorio de la organización a partir de la experiencia en los proyectos y el conocimiento adquirido de fuentes externas.

A fin de construir este entorno computacional, se divide el trabajo en cuatro sub-proyectos de tesis:

1. Arquitectura de software para el entorno computacional de KUALI-BEH [10].

2. Repositorio de métodos y prácticas de proyectos de software para KUALI-BEH.

3. Selección y adaptación de métodos en proyectos de software aplicando KUALI-BEH [11].

4. Tableros de control para el seguimiento de proyectos de software aplicando KUALI-BEH [12].

En este apartado se presenta una descripción de las necesidades, en cuanto a los requerimientos del sub-proyecto “Repositorio de Métodos y Prácticas de Proyectos de Software para KUALI-BEH”, abarcando los requerimientos funcionales y no funcionales.

Los requerimientos van directamente centrados en las funcionalidades requeridas por el proyecto KUALI-BEH, los participantes en el proyecto y los usuarios finales (practitioner).

1.1 Propósito

El propósito de éste sistema es la creación de un repositorio, que ofrezca a los practicantes de software la posibilidad de almacenar sus conocimientos, gestionarlos y posteriormente poder reutilizarlos e incluso adaptarlos y actualizarlos. Estos conocimientos deberán ser modelados como métodos y prácticas de acuerdo al proyecto KUALI-BEH.

1.2 Alcance

El sistema de repositorio creado en este trabajo, abarcará la vista estática del proyecto KUALI-BEH, la cual permite almacenar, resguardar, organizar y consultar los conocimientos de los practicantes de desarrollo de software en Métodos y Prácticas.

1.3 Definiciones de Conceptos

La tabla 1 contiene las definiciones de los conceptos manejados en éste documento.

Tabla 1. Glosario de Conceptos.

NOMBRE	DEFINICIÓN
Proyecto de Software (Software Project)	Es un esfuerzo temporal que tiene como finalidad la realización de un producto de software que satisfaga las necesidades planteadas de los Stakeholders. Este esfuerzo es realizado por un equipo de trabajo integrado por practicantes activos de la ingeniería de software.
Producto de Trabajo (Work Product)	Es el recurso generado como resultado o utilizado como entrada de una práctica, actividad o método.
Equipo de Trabajo (Work Team)	Es un grupo de practicantes de ingeniería de software que trabaja en conjunto para la realización de los objetivos planteados a lo largo de un proyecto de software
Producto de Software (Software Product)	Un producto de software es el resultado de la ejecución del método. Puede contener un conjunto de programas informáticos, procedimientos y documentación asociada. Esta es una especialización de un Producto de Trabajo [6].
Actividad (Activity)	Es un conjunto de conocimientos y habilidades, tareas, métricas, criterios de verificación y herramientas que contribuyen a la realización del objetivo de una práctica. Un conjunto de actividades conforman una guía.
Practicante	Persona que forma parte del equipo de trabajo y que se encuentra participando activamente en un proyecto de software. El practicante debe contar con los conocimientos y habilidades requeridos para poder desempeñar las tareas y actividades contenidas en las prácticas en las que participa.

Condición **(Condition)**	Una condición es una situación o circunstancia de algo o alguien que tiene relación con el proyecto de software que se está realizando. Estas condiciones, pueden ser entradas o resultadas de una práctica, método o una actividad.
Guía **(Guide)**	Es un conjunto de actividades que unidas sirven de apoyo para lograr el objetivo de una práctica. Una práctica puede tener una o más guías que ayudan a transformar las entradas en resultados.
Entrada **(Input)**	Se define como las características esperadas por un producto de trabajo y / o las condiciones necesarias para iniciar la ejecución de una práctica, método o actividad.
Resultado **(Result)**	Se define como las características esperadas de un producto de trabajo y / o condiciones requeridas como salidas después de la ejecución de una práctica, guía o actividad.
Conocimientos y Habilidades **(Knowledge and Skills)**	Conjunto de habilidades, aptitudes y destrezas que son requeridas por el practicante de ingeniería de software para poder ejecutar una práctica.
Método **(Method)**	Es un conjunto **coherente**, **consistente** y **completo** de prácticas. Un conjunto de prácticas es **coherente** si el objetivo de cada práctica contribuye con el propósito del método. Es **consistente** si cada una de sus entradas y resultados están interrelacionados y son útiles. Es **completa** si el logro de todos los objetivos de las prácticas, cumplen totalmente el propósito del método.
Práctica **(Practice)**	Es un conjunto de guías, actividades y tareas que sirven para asesorar la producción de un resultado(s) a partir de una entrada(s). Una práctica contiene criterios de verificación asociados a cada una de sus actividades que ayudan a determinar que los resultados que se obtuvieron cumplen con el objetivo de la práctica. También cuenta con métricas que sirven para evaluar el rendimiento de cada una de las actividades que contiene y por lo tanto evalúan el rendimiento de la práctica en general. Cada práctica cuenta con un objetivo particular y el conjunto de

	guías y actividades contenidas en ella deberá servir de apoyo para su cumplimiento.
Stakeholder	Es un individuo u organización que tiene interés en la realización de un producto de software que satisfaga sus necesidades.
Herramienta (Tool)	Una herramienta es un documento, un software o cualquier dispositivo que ayude a los practicantes de ingeniería de software y al equipo en general a realizar una tarea.
Tarea (Task)	Una tarea es la mínima acción que se realiza dentro de una actividad.

1.4 Referencias

[6] KUALI-BEH V1.1, https://docs.google.com/a/kuali-kaans.mx/viewer?a=v&pid=sites&srcid=a3VhbGkta2FhbnMubXh8a3VhbGkta2FhbnN8Z3g6MTYxMmQ2OWUxODJkZDI1YQ, 12/08/2012

[10] Alberto Tapia Durán, tesis de maestría en ingeniería: Arquitectura de software para el entorno computacional de KUALI-BEH

[11] Eraím Ruíz Sánchez, tesis de maestría en ingeniería: Selección y adaptación de métodos en proyectos de software aplicando KUALI-BEH

[12] José Luis Urrutia Velázquez, tesis de maestría en ingeniería: Tableros de control para el seguimiento de proyectos de software aplicando KUALI-BEH

2. Descripción general

El producto pretende ser de ayuda en llevar un amplio control sobre los conocimientos almacenados por los practicantes de desarrollo de software.

El sistema permitirá a los practicantes dar de alta, modificar, consultar y borrar prácticas, métodos, actividades, así como el contenido del catálogo de conocimientos manejados por KUALI-BEH y permitirá la relación de todos estos

conceptos de acuerdo a la vista estática. Ver figura 1.

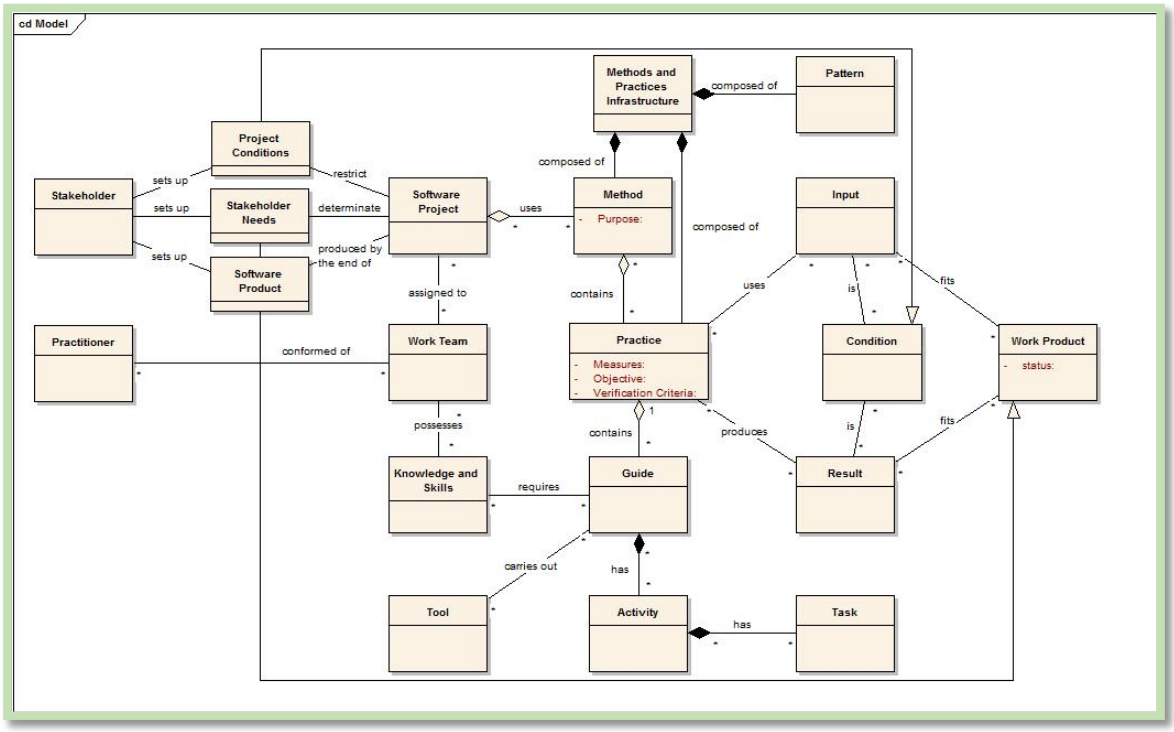


Figura 1. Proyecto de software. Conceptos comunes, sus relaciones y atributos [6]

Se proyecta que el sistema se aloje en un host o red LAN y los usuarios puedan tener acceso a él desde cualquier equipo que cuente con un navegador web y conexión a internet o intranet.

3. Requisitos No Funcionales

Son aquellos que no tienen que ver con la funcionalidad del sistema, pero que son de importancia para la eficacia y eficiencia en su uso.

3.1 Usabilidad

Los usuarios que utilizarán la aplicación son personas que tienen experiencia en

el desarrollo de software, debiendo tener conocimientos previos sobre los conceptos manejados por KUALI-BEH. Estos usuarios podrán gestionar la información contenida en el sistema, entendiendo como tal el hecho de poder almacenar información sobre proyectos realizados previamente, así como utilizar la información que haya sido dada de alta por ellos o por practicantes previos.

La interfaz de la herramienta deberá ser intuitiva, de tal forma que pudiera utilizarse sin previa capacitación, o con una capacitación mínima. Los principales conocimientos que se deberá tener son los conceptos de prácticas, métodos, actividades y sus respectivas asociaciones con el catálogo manejado en la vista estática de KUALI-BEH.

3.2 Confiabilidad

El tiempo disponible de la aplicación será de 24hrs al día, los practicantes podrán ingresar a ella en cualquier momento puesto que podrán ingresar a través de Internet. En caso de ser instalada en una red LAN, el sistema funcionará de acuerdo a las normas de la organización.

El promedio de fallas en cuestión de disponibilidad dependerá del proveedor que sea contratado para proporcionar el servicio de hospedaje y en el cual se encuentre guardado el sistema de repositorio, por tanto el tiempo que lleve en reparar las fallas para poder recuperar el servicio del sistema dependerá del proveedor de hospedaje.

3.3 Eficiencia

Los recursos que el sistema necesitará por cada equipo que haga uso de él, será el mínimo, ya que únicamente se necesitarán las características necesarias para poder abrir una página de internet y el ancho de banda adecuado para el envío y recepción de información.

3.4 Restricciones de diseño y construcción

El Sistema estará implementado bajo el lenguaje de programación Java, usando de éste la versión 1.7, también implementará una base de datos bajo el gestor PostgreSQL en su versión 9.2. Para la implementación de validaciones y en general de la interfaz del sistema, se empleará el framework JSF específicamente su extensión conocida como PrimeFaces en su versión 3.5.

Para la conexión y manipulación de la base de datos se hará también uso del framework Hibernate en su versión 3.2.5 como herramienta de mapeo, empleando su lenguaje de consulta HQL para realizar las distintas transacciones a la base de datos.

Se acordó el empleo de la arquitectura Modelo Vista Controlador (MVC).

3.5 Paquetes

En éste apartado se presenta el diagrama de paquetes del sistema.

3.5.1 Paquetes del Software

La figura 2 muestra los diferentes paquetes contenidos en el sistema, ésta distribución fue acordada por todo el equipo de trabajo con la finalidad de facilitar la integración de todos los proyectos.

Partiendo de la arquitectura MVC se tiene un paquete Model, en el cual se encuentran mapeadas las diferentes tablas contenidas en la base de datos en forma de clases, éste mapeo es auxiliado por Hibernate el cual utiliza notaciones que permiten la fluidez del mismo. El paquete Dao contiene las clases de tipo interface que describen la lógica del comportamiento, en el paquete Impl se encuentran las implementaciones de cada una de estas clases. El paquete Bean contiene las clases Managed Bean que utilizará PrimeFaces en la interfaz. Los paquetes xhtml, images y resources, almacenan todos los archivos que compondrán la interfaz de usuario.

Las clases interfaces que contiene el paquete Dao son las que permitirán la comunicación del sistema con los demás proyectos. Todos los proyectos contendrán un paquete similar, en el cual únicamente harán el llamado a éste paquete para poder utilizar las diferentes clases y métodos que proporciona el sistema.

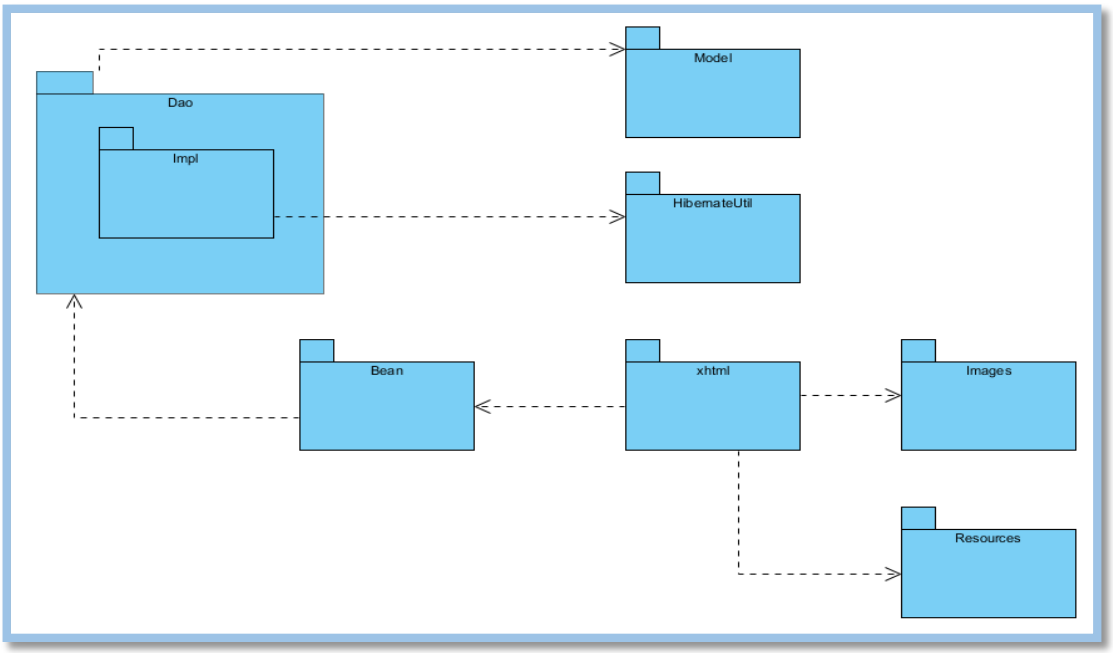


Figura 2. Diagrama de Paquetes

3.6 Licenciamiento

El sistema estará desarrollado con herramientas que posean licencias que no generan gastos.

4. Requisitos Funcionales

Son las funciones o comportamientos que debe cumplir el sistema.

4.1 Identificación de los casos de uso

En la figura 3 se presenta el caso de uso general del repositorio.

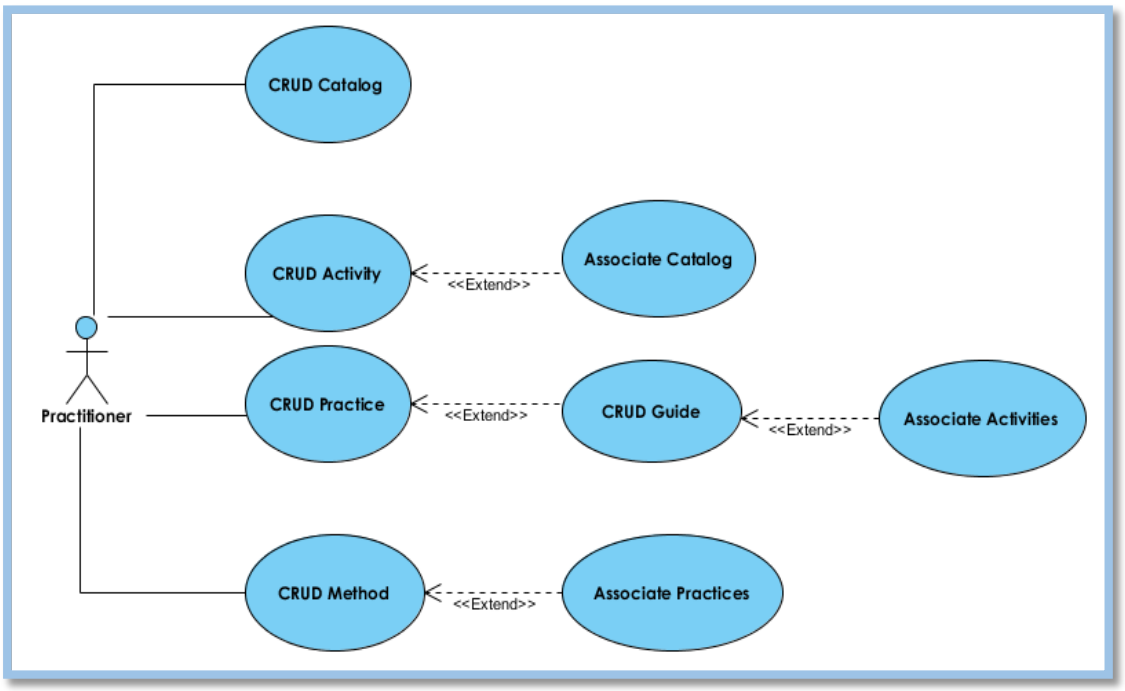


Figura 3. Diagrama General de Casos de Uso del Repositorio.

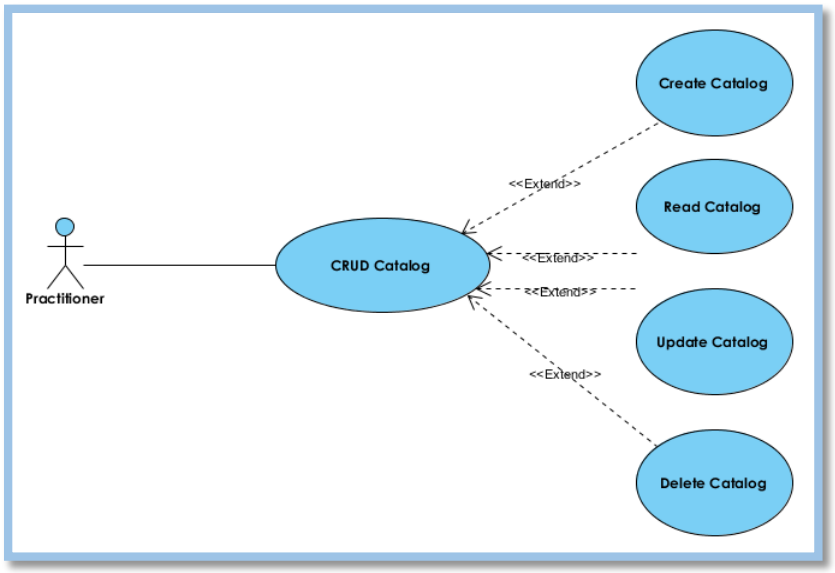
4.2 Detallado de los casos de uso

Este apartado detalla los casos de uso resultantes de los requisitos funcionales.

4.2.1 CRUD Catalog

En esta sección se presenta el diagrama de casos de uso general CRUD (Create, Read, Update y Delete) de catálogo. Ver tabla 2. Un catálogo se refiere a cualquiera de los siguientes conceptos: Knowledge and Skills, Tool, Type, Task, Verification Criteria, Measure, Characteristic, Work Product/Condition

Tabla 2. CRUD Catalog

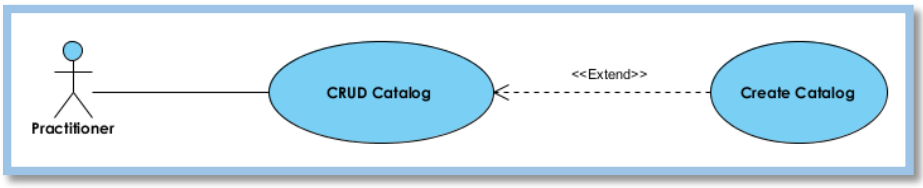
DIAGRAMA DEL CASO DE USO	
	
Caso de Uso:	CRUD Catalog
Tipo:	General.
Actor Principal:	Practitioner (Practicante)
Descripción:	Se describe la forma en que los practicantes podrán crear, consultar, actualizar y borrar la información de los catálogos en el sistema.
Precondiciones:	Haber seleccionado el catálogo deseado: ▪ **Knowledge Skills (Conocimientos y Habilidades)** ▪ **Tools (Herramientas)** ▪ **Types (Tipos)** ▪ **Tasks (Tareas)** ▪ **Verification Criteria (Criterios de Verificación)**

	▪ **Measures (Métricas)** ▪ **Characteristics (Características)** ▪ **WorkProduct/Condition (Producto de Trabajo / Condición)**
Poscondiciones:	El practicante habrá creado, actualizado, consultado o borrado, la información deseada en el catálogo seleccionado.

4.2.1.1 Create Catalog

La tabla 3 (Create Catalog) detalla el caso de uso en el cual el practicante podrá dar de alta un nuevo catálogo (Knowledge and Skills, Tool, Type, Task, Verification Criteria, Measure, Characteristic, Work Product/Condition).

Tabla 3. Create Catalog.

DIAGRAMA DEL CASO DE USO	
	
Caso de Uso:	Create Catalog
Tipo:	Extends
Actor Principal:	Practitioner (Practicante)
Descripción:	En este caso se describe la forma en que los practicantes podrán dar de alta información en algún catálogo.
Precondiciones:	Haber seleccionado el catálogo deseado:

	▪ Knowledge Skills (Conocimientos y Habilidades) ▪ Tools (Herramientas) ▪ Types (Tipos) ▪ Tasks (Tareas) ▪ Verification Criteria (Criterios de Verificación) ▪ Measures (Métricas) ▪ Characteristics (Características) ▪ WorkProduct/Condition (Producto de Trabajo / Condición)
Poscondiciones:	El sistema resguardará nueva información correspondiente a alguno de los catálogos

	ACTOR	**SISTEMA**	
PASO	ACCIÓN	ACCIÓN	EXCEPCIÓN
1.	El practicante indica al sistema que desea agregar información al catálogo.		
2.		El sistema muestra el formulario correspondiente al catálogo seleccionado.	
3.	El practicante llena los campos del formulario y envía los datos		
4.		El sistema valida los datos	E1
5.		El sistema guarda los datos	

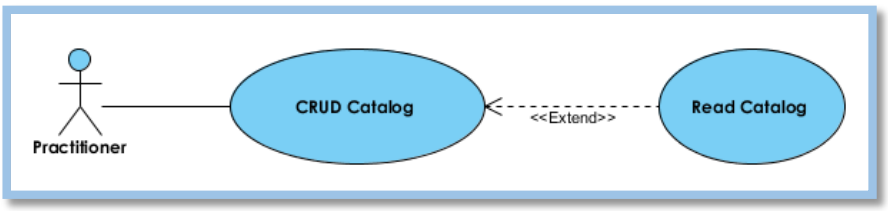
		que el practicante almacenó y manda un mensaje de aviso.	

ID	DESCRIPCIÓN	ACCIÓN
E1	Faltan Datos o Datos ya Existentes	El sistema envía un mensaje de error y termina CU

4.2.1.2 Read Catalog

El caso de uso Read Catalog mostrado en la tabla 4, permite comprender la manera en la que el practicante podrá visualizar la información contenida en los catálogos del sistema (Knowledge and Skills, Tool, Type, Task, Verification Criteria, Measure, Characteristic, Work Product/Condition).

Tabla 4. Read Catalog.

DIAGRAMA DEL CASO DE USO	
	
Caso de Uso:	Read Catalog
Tipo:	Extends
Actor Principal:	Practitioner (Practicante)

Descripción:	Se describe la forma en que los Practicantes podrán consultar la información de algún catálogo.
Precondiciones:	Haber seleccionado el catálogo deseado: ▪ Knowledge Skills (Conocimientos y Habilidades) ▪ Tools (Herramientas) ▪ Types (Tipos) ▪ Tasks (Tareas) ▪ Verification Criteria (Criterios de Verificación) ▪ Measures (Métricas) ▪ Characteristics (Características) ▪ WorkProduct/Condition (Producto de Trabajo / Condición) Deben existir datos dentro de los catálogos del sistema
Poscondiciones:	

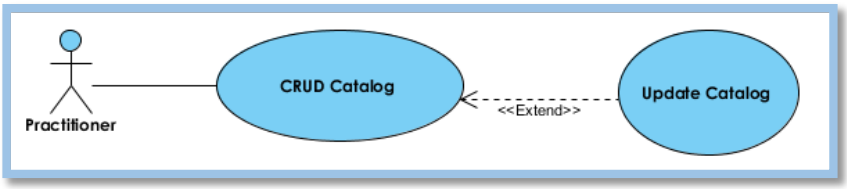
	ACTOR	SISTEMA	
PASO	ACCIÓN	ACCIÓN	EXCEPCIÓN
1.		El sistema muestra una lista de los datos existentes dentro del catálogo seleccionado	
2.	El practicante selecciona uno de los datos mostrados en la lista.		

3.		El sistema muestra la información detallada del elemento seleccionado.	

4.2.1.3 Update Catalog

El caso de uso Update Catalog de la tabla 5, describe la manera en la que el sistema permite al practicante poder modificar la información contenida en algunos de los catálogos (Knowledge and Skills, Tool, Type, Task, Verification Criteria, Measure, Characteristic, Work Product/Condition).

Tabla 5. Update Catalog.

DIAGRAMA DEL CASO DE USO	
	
Caso de Uso:	Update Catalog
Tipo:	Extends
Actor Principal:	Practitioner (Practicante)
Descripción:	Se describe la forma en que los practicantes podrán actualizar la información de algún catálogo.
Precondiciones:	Haber seleccionado el catálogo deseado: ▪ Knowledge Skills (Conocimientos y Habilidades) ▪ Tools (Herramientas)

	▪ Types (Tipos) ▪ Tasks (Tareas) ▪ Verification Criteria (Criterios de Verificación) ▪ Measures (Métricas) ▪ Characteristics (Características) ▪ WorkProduct/Condition (Producto de Trabajo / Condición) CU Read Catalog
Poscondiciones:	El sistema almacenará la información modificada dentro de la base de datos.

	ACTOR	**SISTEMA**	
PASO	ACCIÓN	ACCIÓN	EXCEPCIÓN
1.	El practicante indica al sistema que desea modificar información del catálogo		
2.		El sistema muestra el formulario del catálogo seleccionado con los valores que se habían almacenado previamente	
3.	El practicante modifica los valores de los campos del formulario y envía la información		

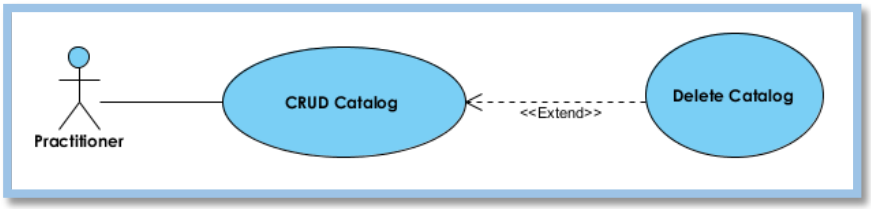
4.		El sistema valida los datos	E1
5.		El sistema almacena los datos que el practicante modificó y manda un mensaje de aviso.	

ID	DESCRIPCIÓN	ACCIÓN
E1	Faltan Datos o Datos ya Existentes	El sistema envía un mensaje de error y termina CU

4.2.1.4 Delete Catalog

La tabla 6 Delete Catalog es el caso de uso que describe la forma en la que el practicante podrá eliminar la información de los catálogos (Knowledge and Skills, Tool, Type, Task, Verification Criteria, Measure, Characteristic, Work Product/Condition).

Tabla 6. Delete Catalog.

DIAGRAMA DEL CASO DE USO	
	
Caso de Uso:	Delete Catalog
Tipo:	Extends

Actor Principal:	Practitioner (Practicante)
Descripción:	Describe la forma en que los practicantes podrán eliminar la información correspondiente a algún catálogo.
Precondiciones:	Haber seleccionado el catálogo deseado: ▪ Knowledge Skills (Conocimientos y Habilidades) ▪ Tools (Herramientas) ▪ Types (Tipos) ▪ Tasks (Tareas) ▪ Verification Criteria (Criterios de Verificación) ▪ Measures (Métricas) ▪ Characteristics (Características) ▪ WorkProduct/Condition (Producto de Trabajo / Condición) CU Read Catalog
Poscondiciones:	El sistema eliminará la información deseada de la base de datos.

	ACTOR	SISTEMA	
PASO	ACCIÓN	ACCIÓN	EXCEPCIÓN
1.	El practicante indica al sistema que desea eliminar los datos mostrados del catálogo		
2.		El sistema muestra un mensaje preguntando al	

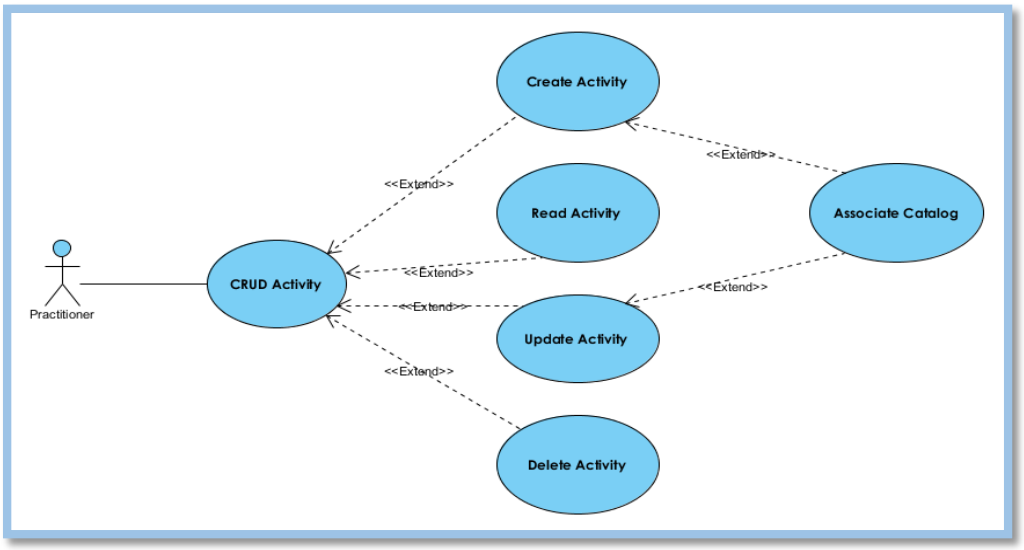
		usuario si realmente desea eliminar el valor seleccionado.	
3.	El practicante reitera su deseo de eliminar la información		E1
4.		El sistema elimina la información	

ID	DESCRIPCIÓN	ACCIÓN
E1	El practicante dice al sistema que no desea eliminar la información	Termina CU

4.2.2 CRUD Activity

El caso de uso CRUD Activity mostrado en la Tabla 7, describe de manera general los casos de uso Create Activity, Read Activity, Update Activity y Delete Activity, mediante los que el practicante podrá crear, consultar, actualizar y eliminar la información de las actividades.

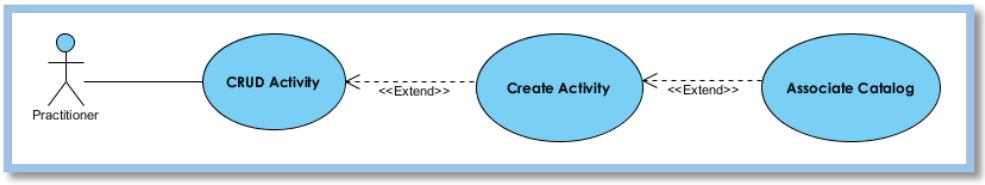
Tabla 7. CRUD Activity.

DIAGRAMA DEL CASO DE USO	
	
Caso de Uso:	CRUD Activity
Tipo:	General.
Actor Principal:	Practitioner (Practicante)
Descripción:	Se describe la forma en que los practicantes podrán crear, consultar, actualizar y borrar información de las actividades en el sistema.
Precondiciones:	Haber seleccionado la opción Activities en el menú principal
Poscondiciones:	El practicante habrá creado, actualizado, consultado o borrado, la información deseada de alguna actividad

4.2.2.1 Create Activity

El caso de uso Create Activity de la tabla 8 describe la forma en que el Practicante podrá almacenar una nueva actividad dentro del sistema.

Tabla 8. Create Activity.

DIAGRAMA DEL CASO DE USO			
			
Caso de Uso:	Create Activity		
Tipo:	Extends		
Actor Principal:	Practitioner (Practicante)		
Descripción:	Describe la forma en que el Practicante podrá almacenar una nueva actividad dentro del sistema.		
Precondiciones:	Haber seleccionado Activities dentro del menú principal del sistema		
Poscondiciones:	El Practicante habrá almacenado información correspondiente a una nueva actividad dentro del sistema		
	ACTOR	**SISTEMA**	
PASO	ACCIÓN	ACCIÓN	EXCEPCIÓN
1.	El practicante indica al sistema que desea agregar información de una nueva actividad		-
2.		El sistema muestra el formulario correspondiente a actividades	
3.	El practicante llena los		

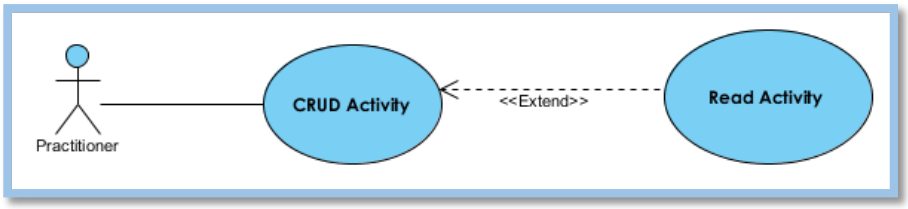
	campos del formulario y envía los datos		
4.		El sistema valida los datos	E1
5.		El sistema guarda los datos que el practicante almacenó y manda un mensaje de aviso.	

ID	DESCRIPCIÓN	ACCIÓN
E1	Faltan Datos o Datos ya Existentes	El sistema envía un mensaje de error y termina CU

4.2.2.2 Read Activity

Describe la forma en que el practicante podrá consultar información referente a una actividad dentro del sistema. Ver tabla 9.

Tabla 9. Read Activity.

DIAGRAMA DEL CASO DE USO	
	
Caso de Uso:	Read Activity
Tipo:	Extends

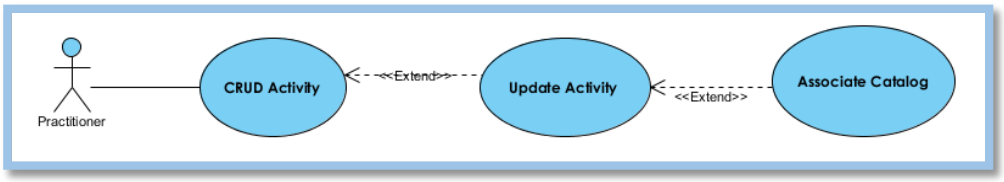
Actor Principal:	Practitioner (Practicante)
Descripción:	Describe la forma en que el practicante podrá consultar información referente a una actividad dentro del sistema.
Precondiciones:	Haber seleccionado Activities dentro del menú principal del sistema Deben existir actividades guardadas en el sistema
Poscondiciones:	El sistema mostrará la información deseada al practicante.

	ACTOR	SISTEMA	
PASO	ACCIÓN	ACCIÓN	EXCEPCIÓN
1.		El sistema muestra una lista de las actividades existentes	
2.	El practicante selecciona una actividad de las mostradas en la lista		
3.		El sistema muestra la información detallada de la actividad seleccionada	

4.2.2.3 Update Activity

Se describe la forma en que los practicantes podrán actualizar la información de una actividad contenida en el sistema. Ver tabla 10.

Tabla 10.Update Activity.

DIAGRAMA DEL CASO DE USO			
			
Caso de Uso:	Update Activity		
Tipo:	Extends		
Actor Principal:	Practitioner (Practicante)		
Descripción:	Se describe la forma en que los practicantes podrán actualizar la información de una actividad contenida en el sistema.		
Precondiciones:	Haber seleccionado Activities dentro del menú principal del sistema CU Read Activity		
Poscondiciones:	El practicante habrá actualizado información referente a una actividad dentro del sistema		
	ACTOR	**SISTEMA**	
PASO	ACCIÓN	ACCIÓN	EXCEPCIÓN
1.	El practicante indica al sistema que desea modificar información de una actividad		
2.		El sistema muestra el formulario con los datos de la actividad	
3.	El practicante modifica los		

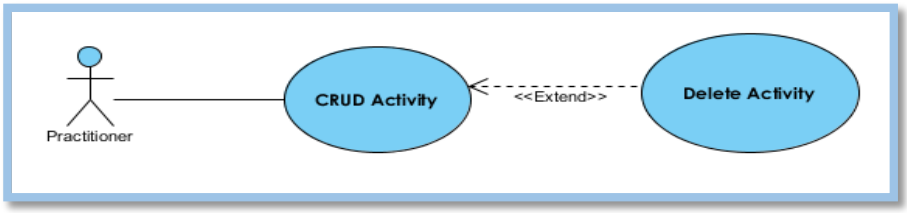
	campos del formulario y envía los datos		
4.		El sistema valida los datos	E1
5.		El sistema guarda los datos que el practicante modificó y manda un mensaje de aviso.	

ID	DESCRIPCIÓN	ACCIÓN
E1	Faltan Datos o Datos ya Existentes	El sistema envía un mensaje de error y termina CU

4.2.2.4 Delete Activity

Describe la forma en que los practicantes podrán eliminar la información correspondiente a una actividad. Ver tabla 11.

Tabla 11.Delete Activity.

DIAGRAMA DEL CASO DE USO	
	
Caso de Uso:	Delete Activity
Tipo:	Extends

Actor Principal:	Practitioner (Practicante)
Descripción:	Describe la forma en que los practicantes podrán eliminar la información correspondiente a una actividad.
Precondiciones:	Haber seleccionado Activities dentro del menú principal del sistema CU Read Activity
Poscondiciones:	El sistema habrá eliminado la información deseada.

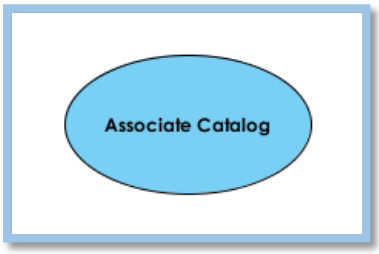
	ACTOR	**SISTEMA**	
PASO	ACCIÓN	ACCIÓN	EXCEPCIÓN
1.	El practicante indica al sistema que desea eliminar los datos mostrados de la actividad		
2.		El sistema muestra un mensaje preguntando al usuario si realmente desea eliminar la actividad seleccionada	
3.	El practicante reitera su deseo de eliminar la información		E1
4.		El sistema elimina la información	

ID	DESCRIPCIÓN	ACCIÓN
E1	El practicante dice al sistema que no desea eliminar la información	Termina CU

4.2.2.5 Associate Catalog

Se describe la forma en que los practicantes podrán asociar los distintos catálogos (Knowledge and Skills, Tool, Type, Task, Verification Criteria, Measure, Characteristic, Work Product/Condition) a una actividad específica. Ver tabla 12.

Tabla 12. Associate Catalog.

DIAGRAMA DEL CASO DE USO	
	
Caso de Uso:	Associate Catalog
Tipo:	Extends
Actor Principal:	Practitioner (Practicante)
Descripción:	Se describe la forma en que los practicantes podrán asociar los distintos catálogos a una actividad específica.
Precondiciones:	CU Update Activity / CU Create Activity
Poscondiciones:	El practicante habrá asociado distintos catálogos a una actividad dentro del sistema

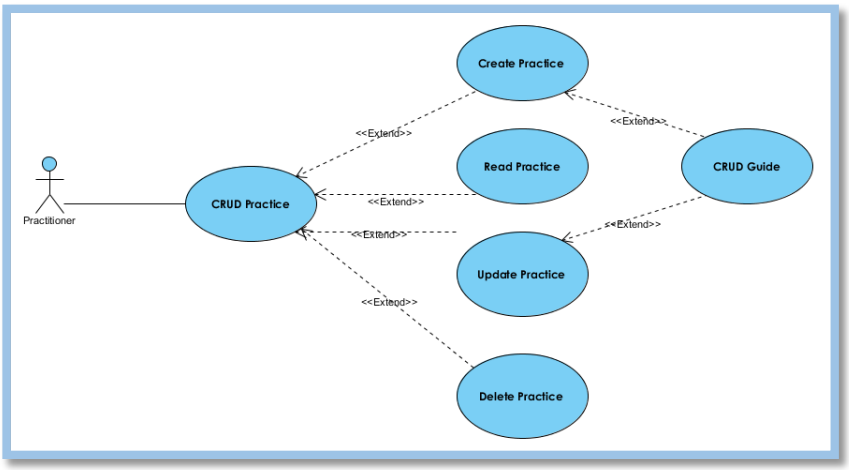
	ACTOR	SISTEMA	
PASO	ACCIÓN	ACCIÓN	EXCEPCIÓN
1.		El sistema muestra los distintos catálogos existentes dentro del sistema	E1
2.	El practicante selecciona los catálogos que desee asociar a la actividad en la que se encuentra trabajando		
3.	El practicante le dice al sistema que desea aceptar las asociaciones realizadas		
4.		El sistema guarda las asociaciones correspondientes	

ID	DESCRIPCIÓN	ACCIÓN
E1	No existen los catálogos requeridos en el sistema	El sistema permite al usuario efectuar CU Create Catalog

4.2.3 CRUD Practice

Se describe la forma en que los practicantes podrán crear, consultar, actualizar y borrar información de las prácticas en el sistema. Ver tabla 13.

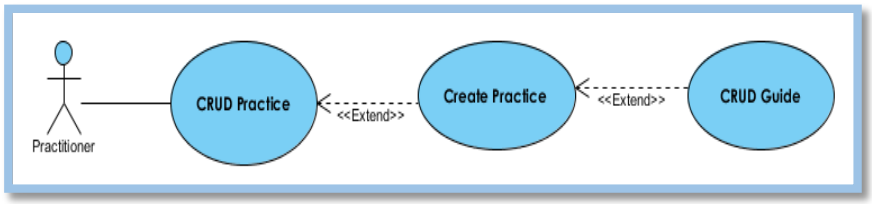
Tabla 13. CRUD Practice.

DIAGRAMA DEL CASO DE USO	
	
Caso de Uso:	CRUD Practice
Tipo:	General.
Actor Principal:	Practitioner (Practicante)
Descripción:	Se describe la forma en que los practicantes podrán crear, consultar, actualizar y borrar información de las prácticas en el sistema.
Precondiciones:	Haber seleccionado la opción Practice dentro del menú principal del sistema
Poscondiciones:	El practicante habrá creado, actualizado, consultado o borrado, la información deseada de alguna practica

4.2.3.1 Create Practice

Describe la forma en que el Practicante podrá almacenar una nueva práctica dentro del sistema. Ver tabla 14.

Tabla 14. Create Practice.

DIAGRAMA DEL CASO DE USO			
			
Caso de Uso:	Create Practice		
Tipo:	Extends		
Actor Principal:	Practitioner (Practicante)		
Descripción:	Describe la forma en que el Practicante podrá almacenar una nueva práctica dentro del sistema.		
Precondiciones:	Haber seleccionado la opción Practice dentro del menú principal del sistema		
Poscondiciones:	El practicante habrá almacenado información correspondiente a una nueva practica dentro del sistema		
	ACTOR	**SISTEMA**	
PASO	ACCIÓN	ACCIÓN	EXCEPCIÓN
1.	El practicante indica al sistema que desea agregar información de una nueva practica		
2.		El sistema muestra el formulario correspondiente para agregar una nueva practica	

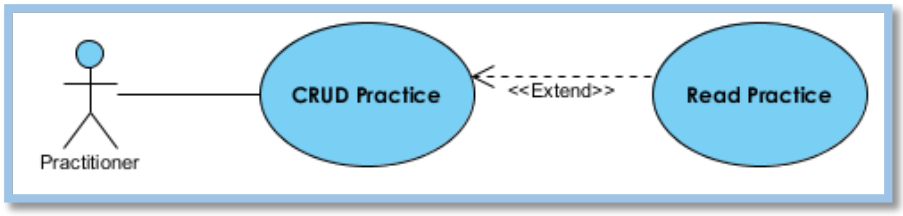
3.	El practicante llena los campos del formulario y envía los datos		
4.		El sistema valida los datos	E1
5.		El sistema guarda los datos que el practicante almacenó y manda un mensaje de aviso.	

ID	DESCRIPCIÓN	ACCIÓN
E1	Faltan Datos o Datos ya Existentes	El sistema envía un mensaje de error y termina CU

4.2.3.2 Read Practice

Describe la manera en que el practicante podrá consultar información referente a una práctica dentro del sistema. Ver tabla 15.

Tabla 15. Read Practice.

DIAGRAMA DEL CASO DE USO


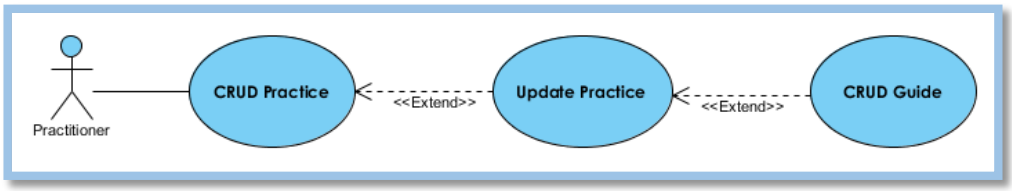
Caso de Uso:	Read Practice
Tipo:	Extends
Actor Principal:	Practitioner (Practicante)
Descripción:	Describe la forma en que el practicante podrá consultar información referente a una práctica dentro del sistema.
Precondiciones:	Haber seleccionado la opción Practice dentro del menú principal del sistema Deben existir prácticas guardadas en el sistema
Poscondiciones:	El sistema mostrará la información deseada al practicante.

	ACTOR	SISTEMA	
PASO	ACCIÓN	ACCIÓN	EXCEPCIÓN
1.		El sistema muestra una lista de las prácticas existentes	-
2.	El practicante selecciona una práctica de las mostradas en la lista		
3.		El sistema muestra la información detallada de la práctica seleccionada	

4.2.3.3 Update Practice

Se describe la forma en que los practicantes podrán actualizar la información de una práctica contenida en el sistema. Ver tabla 16.

Tabla 16. Update Practice.

DIAGRAMA DEL CASO DE USO			
			
Caso de Uso:	Update Practice		
Tipo:	Extends		
Actor Principal:	Practitioner (Practicante)		
Descripción:	Se describe la forma en que los practicantes podrán actualizar la información de una práctica contenida en el sistema.		
Precondiciones:	Haber seleccionado la opción Practice dentro del menú principal del sistema CU Read Practice		
Poscondiciones:	El practicante habrá actualizado información referente a una práctica dentro del sistema		
	ACTOR	**SISTEMA**	
PASO	ACCIÓN	ACCIÓN	EXCEPCIÓN
1.	El practicante indica al sistema que desea modificar información de una práctica		-
2.		El sistema muestra el formulario con los datos de la práctica	

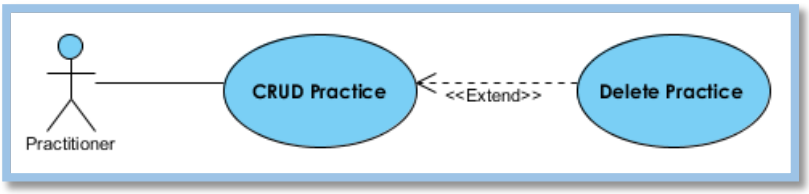
3.	El practicante modifica los campos del formulario y envía los datos		
4.		El sistema valida los datos	E1
5.		El sistema guarda los datos que el practicante modificó y manda un mensaje de aviso.	

ID	DESCRIPCIÓN	ACCIÓN
E1	Faltan Datos o Datos ya Existentes	El sistema envía un mensaje de error y termina CU

4.2.3.4 Delete Practice

El caso de uso Delete Practice describe la forma en que los practicantes podrán eliminar la información correspondiente a una práctica. Ver tabla 17.

Tabla 17. Delete Practice.

DIAGRAMA DEL CASO DE USO	
	
Caso de Uso:	Delete Practice

Tipo:	Extends
Actor Principal:	Practitioner (Practicante)
Descripción:	Describe la forma en que los practicantes podrán eliminar la información correspondiente a una práctica.
Precondiciones:	Haber seleccionado la opción Practice dentro del menú principal del sistema CU Read Practice
Poscondiciones:	El sistema habrá eliminado la información seleccionada.

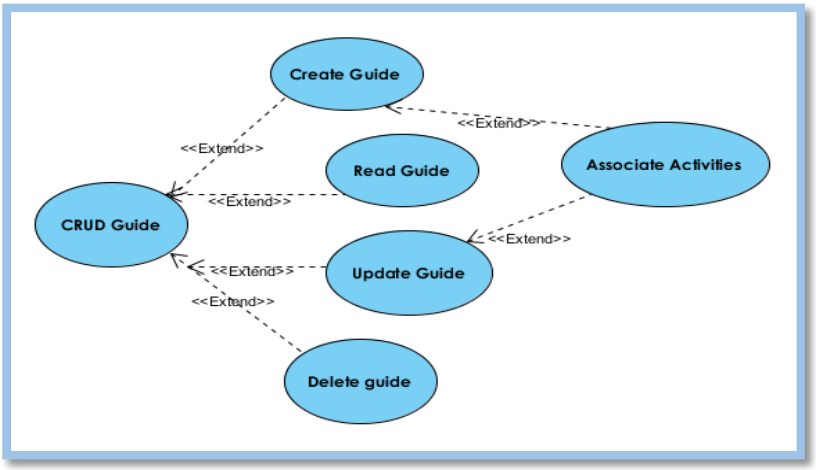
	ACTOR	SISTEMA	
PASO	ACCIÓN	ACCIÓN	EXCEPCIÓN
1.	El practicante indica al sistema que desea eliminar los datos mostrados de la práctica		
2.		El sistema muestra un mensaje preguntando al usuario si realmente desea eliminar la práctica seleccionada	
3.	El practicante reitera su deseo de eliminar la información		E1
4.		El sistema elimina la información	

ID	DESCRIPCIÓN	ACCIÓN
E1	El practicante dice al sistema que no desea eliminar la información	Termina CU

4.2.4 CRUD Guide

Se describe la forma en que los practicantes podrán crear, consultar, actualizar y borrar información de las guías en el sistema. Ver tabla 18.

Tabla 18. CRUD Guide.

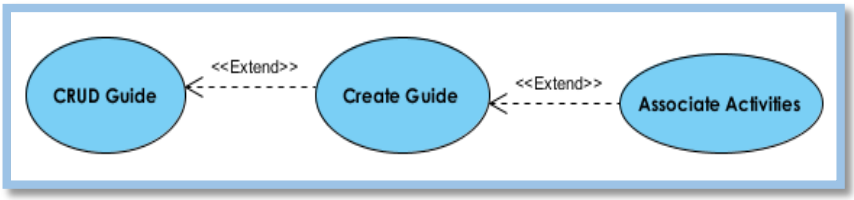
DIAGRAMA DEL CASO DE USO	
	
Caso de Uso:	CRUD Guide
Tipo:	General.
Actor Principal:	Practitioner (Practicante)
Descripción:	Se describe la forma en que los practicantes podrán crear, consultar, actualizar y borrar información de las guías en el sistema.

Precondiciones:	Deben existir prácticas guardadas en el sistema Haber seleccionado la opción Practice dentro del menú principal del sistema CU Read Practice
Poscondiciones:	El practicante habrá creado, actualizado, consultado o borrado, la información deseada de alguna guía

4.2.4.1 Create Guide

La tabla 19 describe la forma en que el Practicante podrá almacenar y asociar a una práctica una nueva guía dentro del sistema (Create Guide).

Tabla 19. Create Guide.

DIAGRAMA DEL CASO DE USO	
	
Caso de Uso:	Create Guide
Tipo:	Extends
Actor Principal:	Practitioner (Practicante)
Descripción:	Describe la forma en que el Practicante podrá almacenar y asociar a una práctica una nueva guía dentro del sistema.
Precondiciones:	Haber seleccionado la opción Practice dentro del menú principal del sistema CU Read Practice

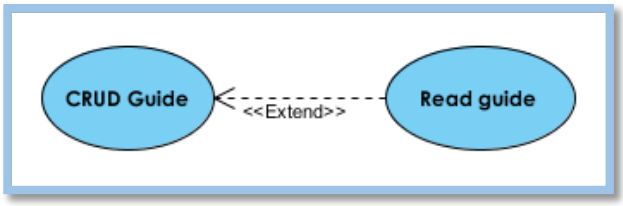
Poscondiciones:	El practicante habrá almacenado información correspondiente a una nueva guía dentro del sistema		
	ACTOR	**SISTEMA**	
PASO	ACCIÓN	ACCIÓN	EXCEPCIÓN
1.	El practicante indica al sistema que desea agregar información de una nueva guía		
2.		El sistema muestra el formulario correspondiente para agregar una nueva guía	
3.	El practicante llena los campos del formulario y envía los datos		
4.		El sistema valida los datos	E1
5.		El sistema guarda los datos que el practicante almacenó y manda un mensaje de aviso.	

ID	DESCRIPCIÓN	ACCIÓN
E1	Faltan Datos o Datos ya Existentes	El sistema envía un mensaje de error y termina CU

4.2.4.2 Read Guide

El caso de uso Read Guide de la tabla 20 describe la forma en que el practicante podrá consultar información referente a las guías asociadas a una práctica dentro del sistema.

Tabla 20. Read Guide.

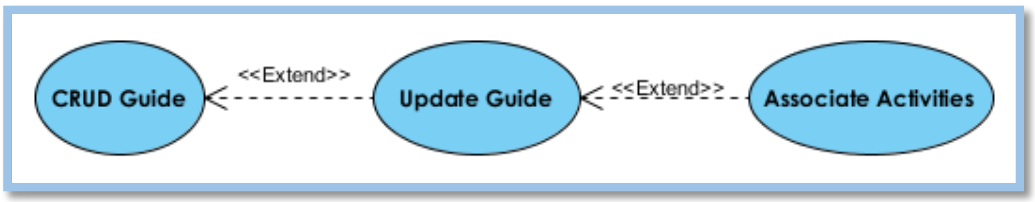
DIAGRAMA DEL CASO DE USO			
			
Caso de Uso:	Read Guide		
Tipo:	Extends		
Actor Principal:	Practitioner (Practicante)		
Descripción:	Describe la forma en que el practicante podrá consultar información referente a las guías asociadas a una práctica dentro del sistema.		
Precondiciones:	Haber seleccionado la opción Practice dentro del menú principal del sistema CU Read Practice		
Poscondiciones:	El sistema mostrará la información deseada al practicante.		
	ACTOR	**SISTEMA**	
PASO	ACCIÓN	ACCIÓN	EXCEPCIÓN
1.		El sistema muestra las	-

		guías existentes dentro de la práctica seleccionada	
2.	El practicante selecciona una guía de todas las mostradas		
3.		El sistema muestra la información detallada de la guía seleccionada	

4.2.4.3 Update Guide

La tabla 21 describe la forma en que los practicantes podrán actualizar la información de una guía contenida en el sistema (Update Guide).

Tabla 21. Update Guide.

DIAGRAMA DEL CASO DE USO	
	
Caso de Uso:	Update Guide
Tipo:	Extends
Actor Principal:	Practitioner (Practicante)
Descripción:	Se describe la forma en que los practicantes podrán actualizar la información de una guía contenida en el sistema.
Precondiciones:	Haber seleccionado la opción Practice dentro del menú principal del sistema

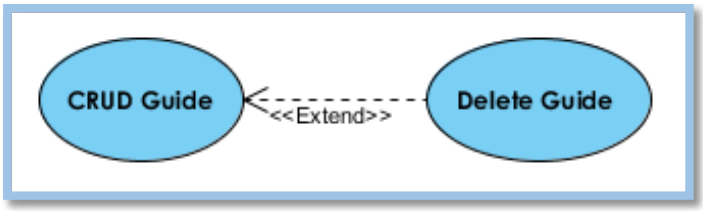
	CU Read Guide		
Poscondiciones:	El practicante habrá actualizado información referente a una guía dentro del sistema		
	ACTOR	**SISTEMA**	
PASO	ACCIÓN	ACCIÓN	EXCEPCIÓN
1.	El practicante indica al sistema que desea modificar información de una guía		-
2.		El sistema muestra el formulario con los datos de la guía	
3.	El practicante modifica los campos del formulario y envía los datos		
4.		El sistema valida los datos	E1
5.		El sistema guarda los datos que el practicante modificó y manda un mensaje de aviso.	

ID	DESCRIPCIÓN	ACCIÓN
E1	Faltan Datos o Datos ya Existentes	El sistema envía un mensaje de error y termina CU

4.2.4.4 Delete Guide

Describe la forma en que los practicantes podrán eliminar la información correspondiente a una guía. Ver tabla 22.

Tabla 22. Delete Guide.

DIAGRAMA DEL CASO DE USO			
			
Caso de Uso:	Delete Guide		
Tipo:	Extends		
Actor Principal:	Practitioner (Practicante)		
Descripción:	Describe la forma en que los practicantes podrán eliminar la información correspondiente a una guía.		
Precondiciones:	Haber seleccionado la opción Practice dentro del menú principal del sistema CU Read Guide		
Poscondiciones:	El sistema habrá eliminado la información seleccionada.		
	ACTOR	**SISTEMA**	
PASO	ACCIÓN	ACCIÓN	EXCEPCIÓN
1.	El practicante indica al sistema que desea eliminar los datos mostrados de la		

	guía		
2.		El sistema muestra un mensaje preguntando al usuario si realmente desea eliminar la guía seleccionada	
3.	El practicante reitera su deseo de eliminar la información		E1
4.		El sistema elimina la información	

ID	DESCRIPCIÓN	ACCIÓN
E1	El practicante dice al sistema que no desea eliminar la información	Termina CU

4.2.4.5 Associate Activities

Se describe la forma en que los practicantes podrán asociar las distintas actividades a una guía específica. Ver tabla 23.

Tabla 23. Associate Activities.

DIAGRAMA DEL CASO DE USO			
			
Caso de Uso:	Associate Activities		
Tipo:	Extends		
Actor Principal:	Practitioner (Practicante)		
Descripción:	Se describe la forma en que los practicantes podrán asociar las distintas actividades a una guía específica.		
Precondiciones:	CU Update Guide / CU Create Guide		
Poscondiciones:	El practicante habrá asociado distintas actividades a una guía dentro del sistema		
	ACTOR	**SISTEMA**	
PASO	ACCIÓN	ACCIÓN	EXCEPCIÓN
1.		El sistema muestra las distintas actividades existentes dentro del sistema	E1
2.	El practicante selecciona las actividades que desee asociar a la guía en la que se encuentra trabajando		

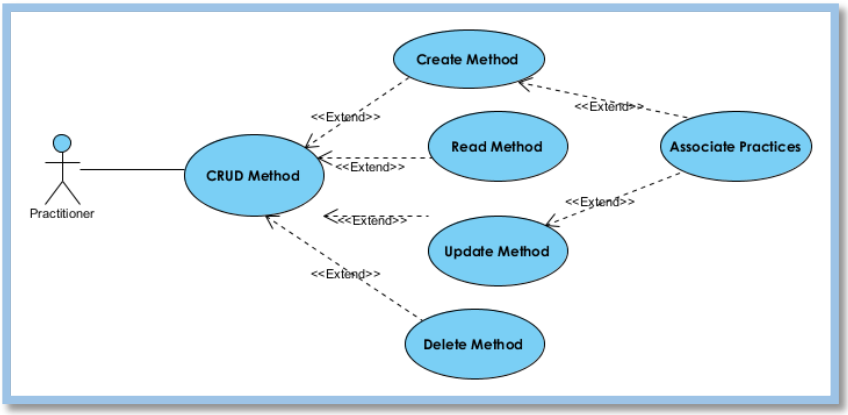
3.	El practicante le dice al sistema que desea aceptar las asociaciones realizadas		
4.		El sistema guarda las asociaciones correspondientes	

ID	DESCRIPCIÓN	ACCIÓN
E1	No existen las actividades requeridas en el sistema	El sistema permite al usuario efectuar CU Create Activity

4.2.5 CRUD Method

El caso de uso CRUD Method mostrado en la tabla 24 describe la forma en que los practicantes podrán crear, consultar, actualizar y borrar información de los métodos en el sistema.

Tabla 24. CRUD Method.

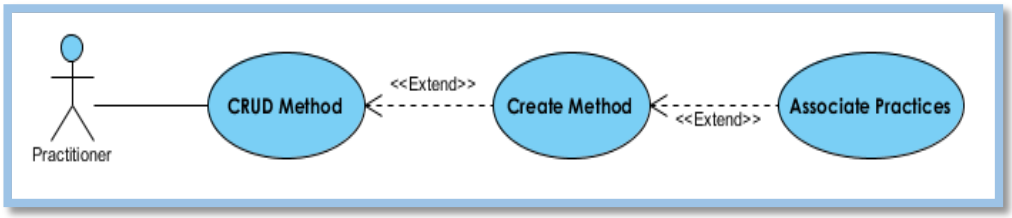
DIAGRAMA DEL CASO DE USO


Caso de Uso:	CRUD Method
Tipo:	General.
Actor Principal:	Practitioner (Practicante)
Descripción:	Se describe la forma en que los practicantes podrán crear, consultar, actualizar y borrar información de los métodos en el sistema.
Precondiciones:	Haber seleccionado la opción Method dentro del menú principal del sistema
Poscondiciones:	El practicante habrá creado, actualizado, consultado o borrado, la información deseada de algún método

4.2.5.1 Create Method

El caso de uso Create Method mostrado en la tabla 25 describe la forma en que el Practicante podrá almacenar un nuevo método dentro del sistema.

Tabla 25. Create Method.

DIAGRAMA DEL CASO DE USO


Caso de Uso:	Create Method
Tipo:	Extends

Actor Principal:	Practitioner (Practicante)
Descripción:	Describe la forma en que el Practicante podrá almacenar un nuevo método dentro del sistema.
Precondiciones:	Haber seleccionado la opción Method dentro del menú principal del sistema
Poscondiciones:	El practicante habrá almacenado información correspondiente a un nuevo método dentro del sistema

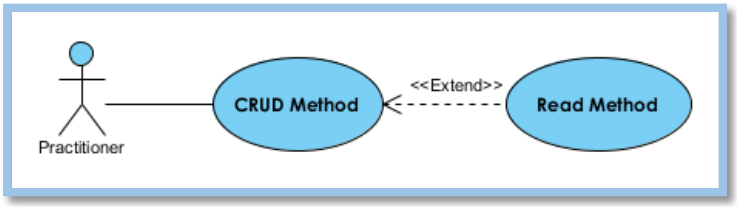
	ACTOR	**SISTEMA**	
PASO	ACCIÓN	ACCIÓN	EXCEPCIÓN
1.	El practicante indica al sistema que desea agregar información de un nuevo método		
2.		El sistema muestra el formulario correspondiente para agregar un nuevo método	
3.	El practicante llena los campos del formulario y envía los datos		
4.		El sistema valida los datos	E1
5.		El sistema guarda los datos que el practicante almacenó y manda un mensaje de aviso.	

ID	DESCRIPCIÓN	ACCIÓN
E1	Faltan Datos o Datos ya Existentes	El sistema envía un mensaje de error y termina CU

4.2.5.2 Read Method

El caso de uso Read Method describe la forma en que el practicante podrá consultar información referente a un método dentro del sistema. Ver tabla 26.

Tabla 26. Read Method.

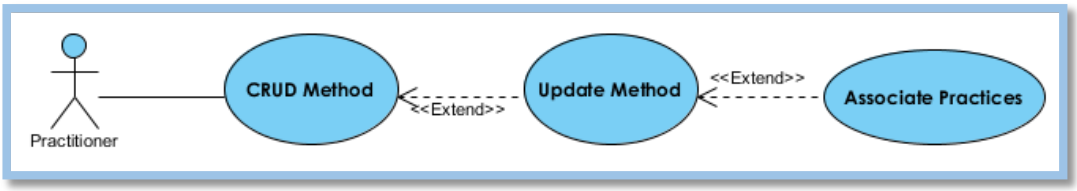
DIAGRAMA DEL CASO DE USO	
	
Caso de Uso:	Read Method
Tipo:	Extends
Actor Principal:	Practitioner (Practicante)
Descripción:	Describe la forma en que el practicante podrá consultar información referente a un método dentro del sistema.
Precondiciones:	Haber seleccionado la opción Method dentro del menú principal del sistema Deben existir métodos guardados en el sistema
Poscondiciones:	El sistema mostrará la información deseada al practicante.

	ACTOR	SISTEMA	
PASO	ACCIÓN	ACCIÓN	EXCEPCIÓN
1.		El sistema muestra una lista de los métodos existentes	-
2.	El practicante selecciona un método de los mostrados en la lista		
3.		El sistema muestra la información detallada del método seleccionado	

4.2.5.3 Update Method

La tabla 27 muestra el caso de uso Update Method que describe la forma en que los practicantes podrán actualizar la información de un método contenido en el sistema.

Tabla 27. Update Method.

DIAGRAMA DEL CASO DE USO	
	
Caso de Uso:	Update Method
Tipo:	Extends

Actor Principal:	Practitioner (Practicante)
Descripción:	Se describe la forma en que los practicantes podrán actualizar la información de un método contenido en el sistema.
Precondiciones:	Haber seleccionado la opción Method dentro del menú principal del sistema CU Read Method
Poscondiciones:	El practicante habrá actualizado información referente a un método dentro del sistema

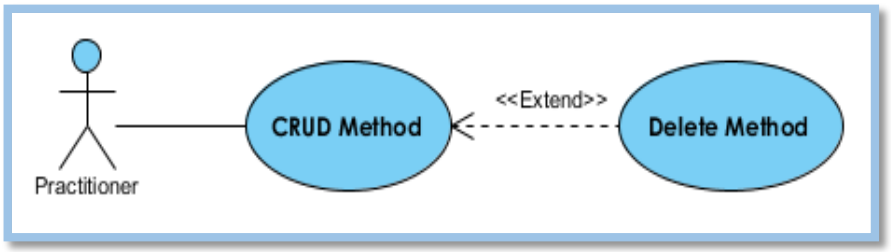
	ACTOR	SISTEMA	
PASO	ACCIÓN	ACCIÓN	EXCEPCIÓN
1.	El practicante indica al sistema que desea modificar información de un método		-
2.		El sistema muestra el formulario con los datos del método	
3.	El practicante modifica los campos del formulario y envía los datos		
4.		El sistema valida los datos	E1
5.		El sistema guarda los datos que el practicante modificó y manda un mensaje de aviso.	

ID	DESCRIPCIÓN	ACCIÓN
E1	Faltan Datos o Datos ya Existentes	El sistema envía un mensaje de error y termina CU

4.2.5.4 Delete Method

EL caso de uso Delete Method describe la forma en que los practicantes podrán eliminar la información correspondiente a un método. Ver tabla 28.

Tabla 28. Delete Method.

DIAGRAMA DEL CASO DE USO	
	
Caso de Uso:	Delete Method
Tipo:	Extends
Actor Principal:	Practitioner (Practicante)
Descripción:	Describe la forma en que los practicantes podrán eliminar la información correspondiente a un método.
Precondiciones:	Haber seleccionado la opción Method dentro del menú principal del sistema CU Read Method
Poscondiciones:	Se habrá eliminado la información seleccionada del sistema.

	ACTOR	SISTEMA	
PASO	ACCIÓN	ACCIÓN	EXCEPCIÓN
1.	El practicante indica al sistema que desea eliminar los datos mostrados del método		
2.		El sistema muestra un mensaje preguntando al usuario si realmente desea eliminar el método seleccionado	
3.	El practicante reitera su deseo de eliminar la información		E1
4.		El sistema elimina la información	

ID	DESCRIPCIÓN	ACCIÓN
E1	El practicante dice al sistema que no desea eliminar la información	Termina CU

4.2.5.5 Associate Practices

El caso de uso Associate Practices mostrado en la tabla 29 describe la forma en que los practicantes podrán asociar las distintas prácticas a un método específico.

Tabla 29. Associate Practices.

DIAGRAMA DEL CASO DE USO			
			
Caso de Uso:	Associate Practices		
Tipo:	Extends		
Actor Principal:	Practitioner (Practicante)		
Descripción:	Se describe la forma en que los practicantes podrán asociar las distintas prácticas a un método específico.		
Precondiciones:	CU Update Method / CU Create Method		
Poscondiciones:	El practicante habrá asociado distintas prácticas a un método dentro del sistema		
	ACTOR	**SISTEMA**	
PASO	ACCIÓN	ACCIÓN	EXCEPCIÓN
1.		El sistema muestra las distintas prácticas existentes dentro del sistema	
2.	El practicante selecciona las prácticas que desee asociar al método en el que se encuentra trabajando		

3.	El practicante le dice al sistema que desea aceptar las asociaciones realizadas		
4.		El sistema guarda las asociaciones correspondientes	

4.3 Interfaz del usuario

Las siguientes son muestras del conjunto de pantallas que conforman la interfaz de usuario del sistema y se presentan de acuerdo a las secciones contenidas en el menú principal.

4.3.1 Menú Principal

A continuación se muestra el menú principal acordado para todo el sistema: “Entorno Computacional para KUALI-BEH. El menú principal contiene cuatro secciones generales: Method, Practice, Activities y Catalog. Dentro del menú catalog se encuentran seis submenús: Knowledge and Skills, Tools, Tasks, Verification Criteria, Measures y Work Product/condition. Ver figura 4.

Figura 4. Menú Principal.

4.3.2 Menú Catalog

El menú Catalog contiene diferentes submenús. Las figuras 5, 6 y 7 muestran de manera general el contenido del menú Catalog.

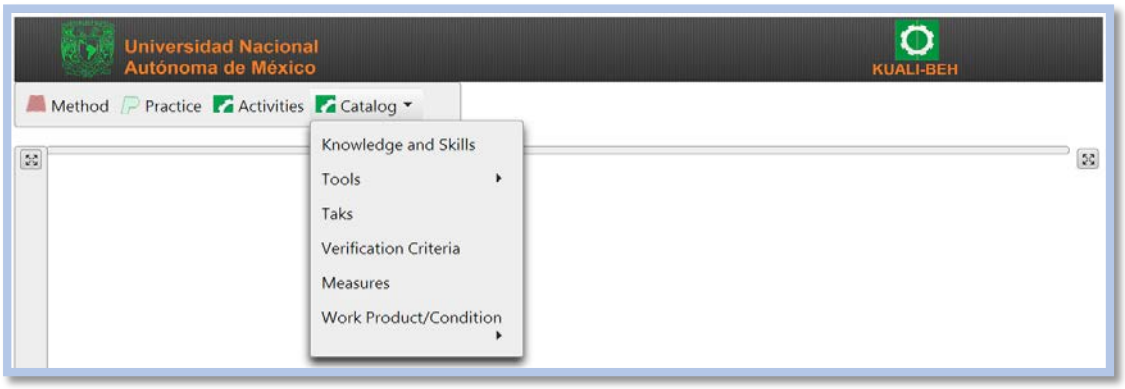


Figura 5. Menú Catalog.

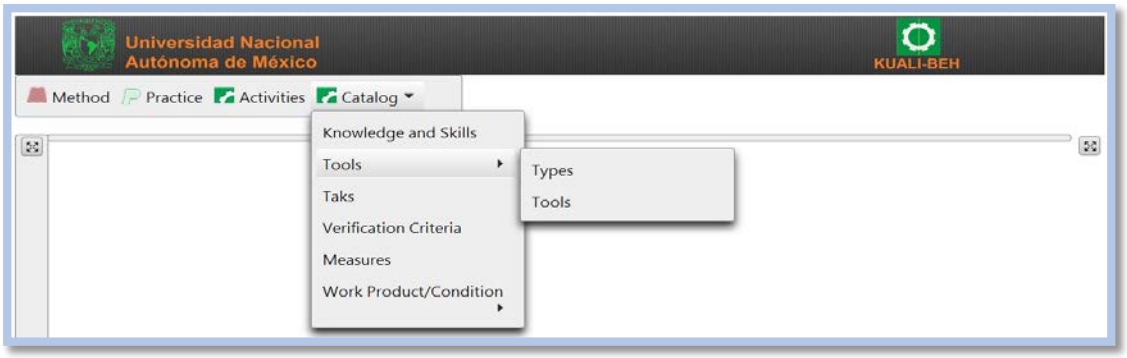


Figura 6. Menú Catalog - Submenú Tools.

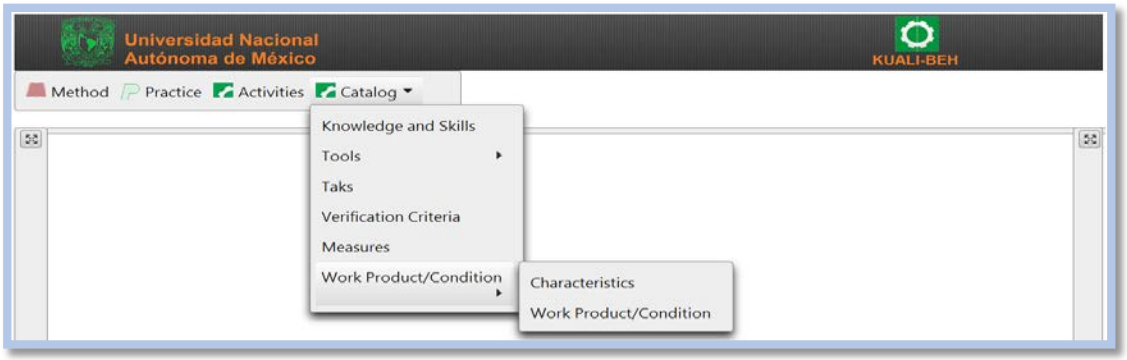


Figura 7. Menú Catalog - Submenú Work Product/Condition.

4.3.2.1 Interfaces del menú Catalog

Las figuras 8, 9, 10, 11, 12, 13, 14 y 15 muestran las pantallas principales correspondientes a las opciones marcadas en el menú catálogo, las cuales son: Knowledge and Skills, Types, Tools, Tasks, Verification Criteria, Measures, Characteristics y Work Product/Condition respectivamente. En éstas pantallas se pueden observar del lado izquierdo una lista con los datos contenidos en el sistema de acuerdo a cada submenú, en dicha lista se presenta un identificador, una breve descripción y un botón que permite seleccionar el dato deseado y ver en la pantalla principal toda la información contenida en él. Del lado superior derecho se encuentran en un inicio dos botones: Add y Cancel.

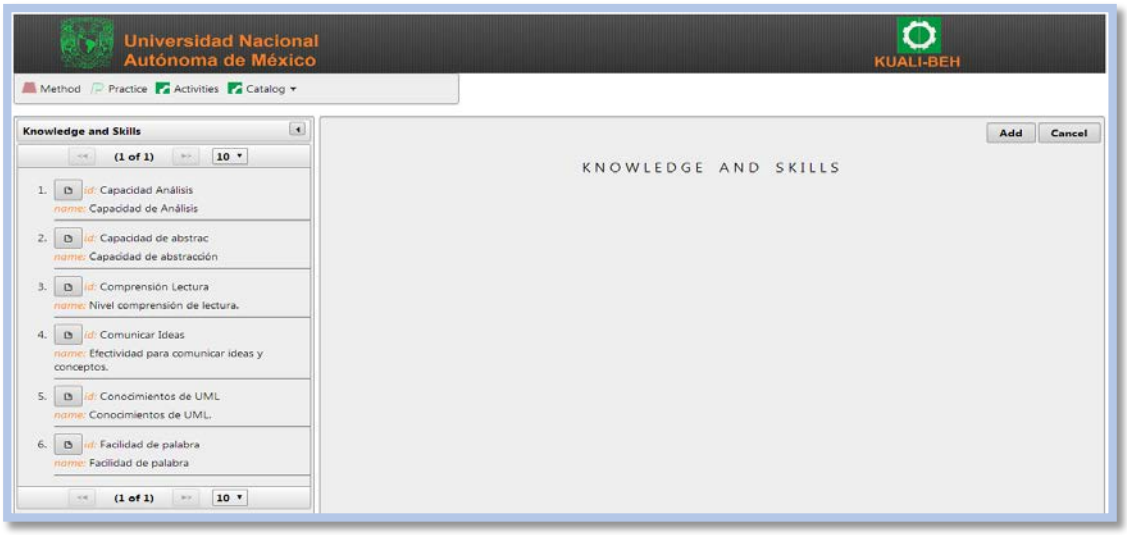


Figura 8. Knowledge and Skills.

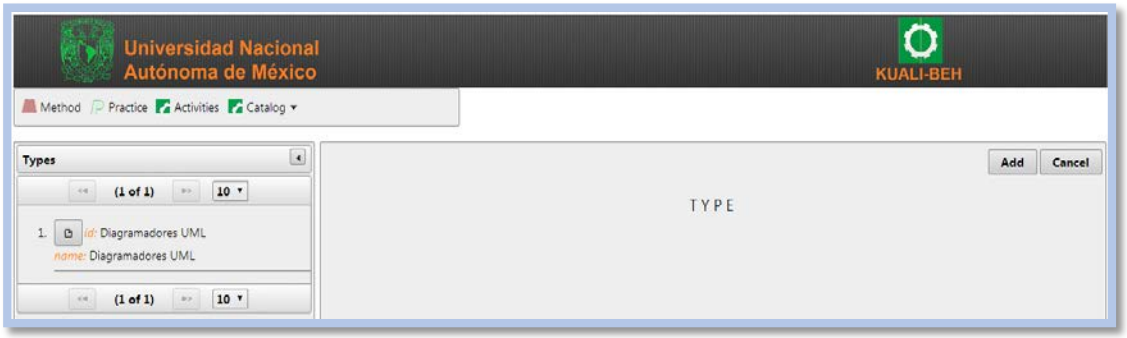


Figura 9. Type.

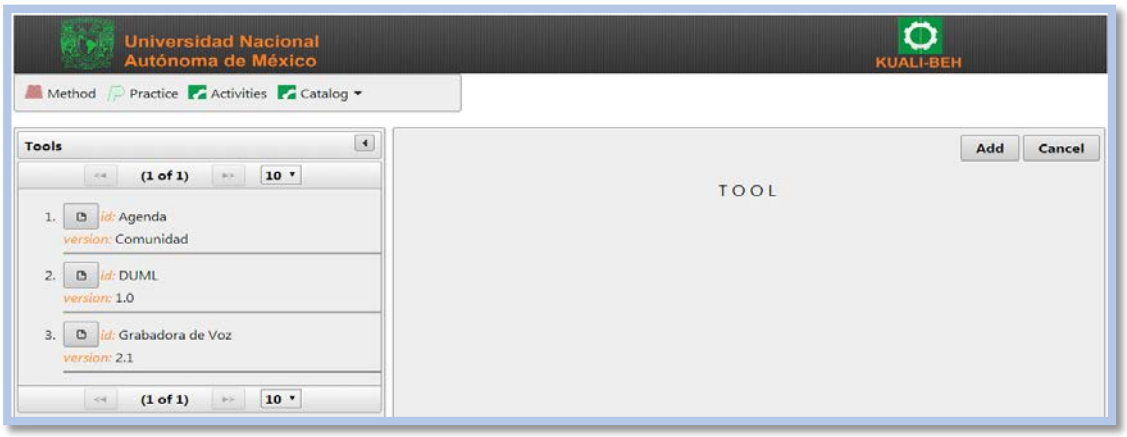


Figura 10. Tool.

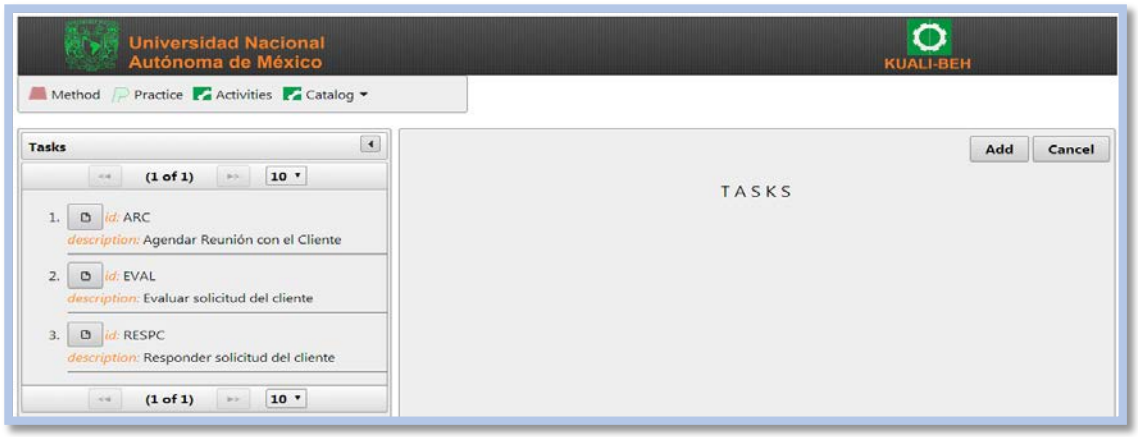


Figura 11. Task.

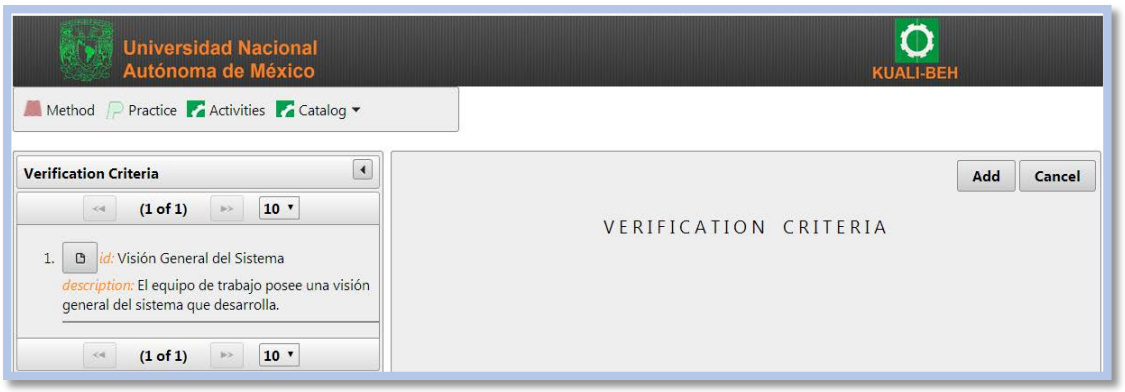


Figura 12. Verification Criteria.

Figura 13. Measure.

Figura 14. Characteristic.

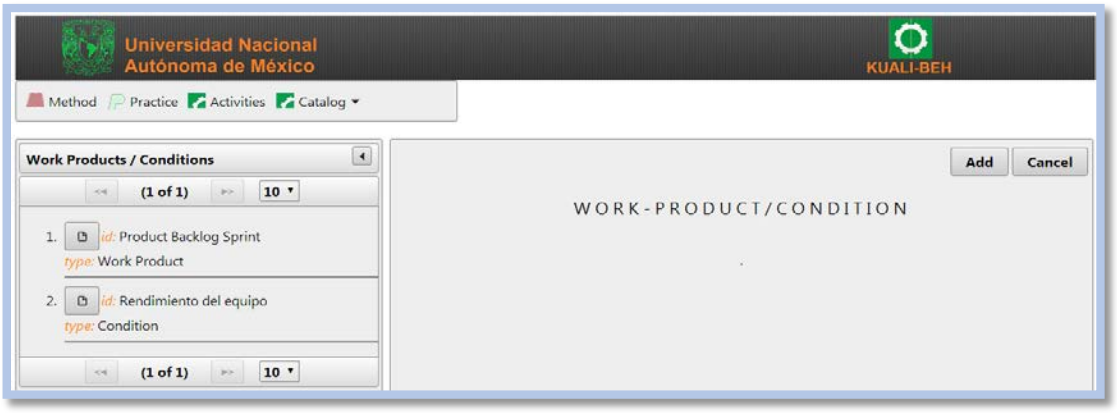


Figura 15. Work Product/Condition.

4.3.2.2 Interfaces Create del menú Catalog

Al presionar el botón Add en cualquiera de las pantallas anteriores (figuras 8, 9, 10, 11, 12, 13, 14, 15) se desplegará una ventana emergente, la cual mostrará un formulario de acuerdo a cada caso, permitiendo al practicante escribir la información requerida y agregar ésta nueva información al sistema presionando el botón Save, tal como se muestra en las figuras 16, 17, 18, 19, 20, 21, 22 y 23.

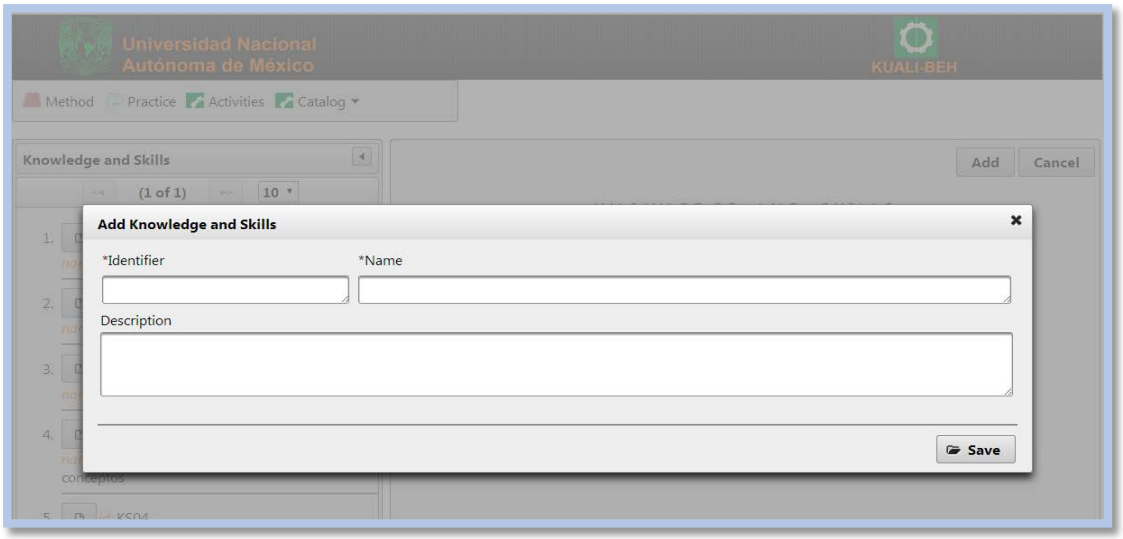


Figura 16. Add Knowledge and Skills.

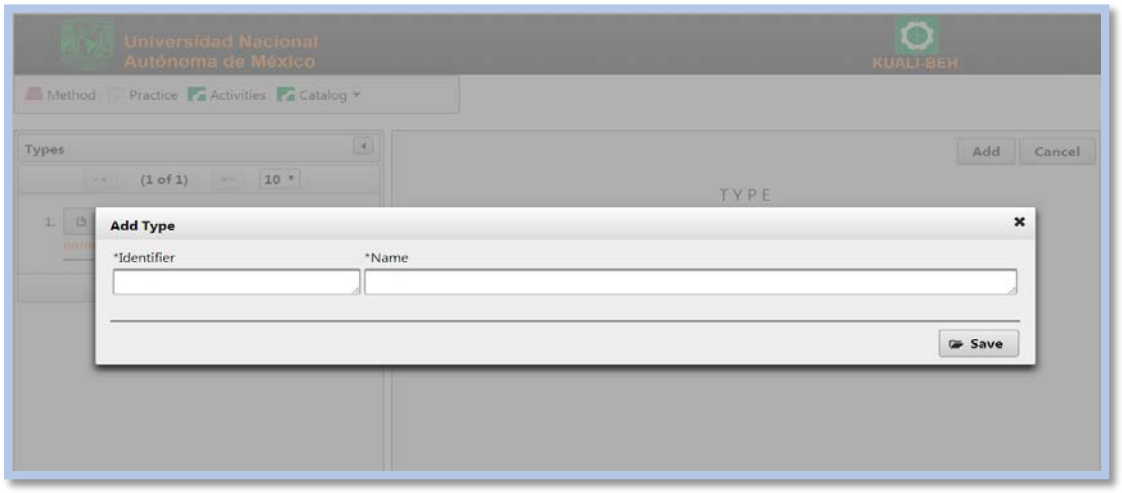


Figura 17. Add Type.

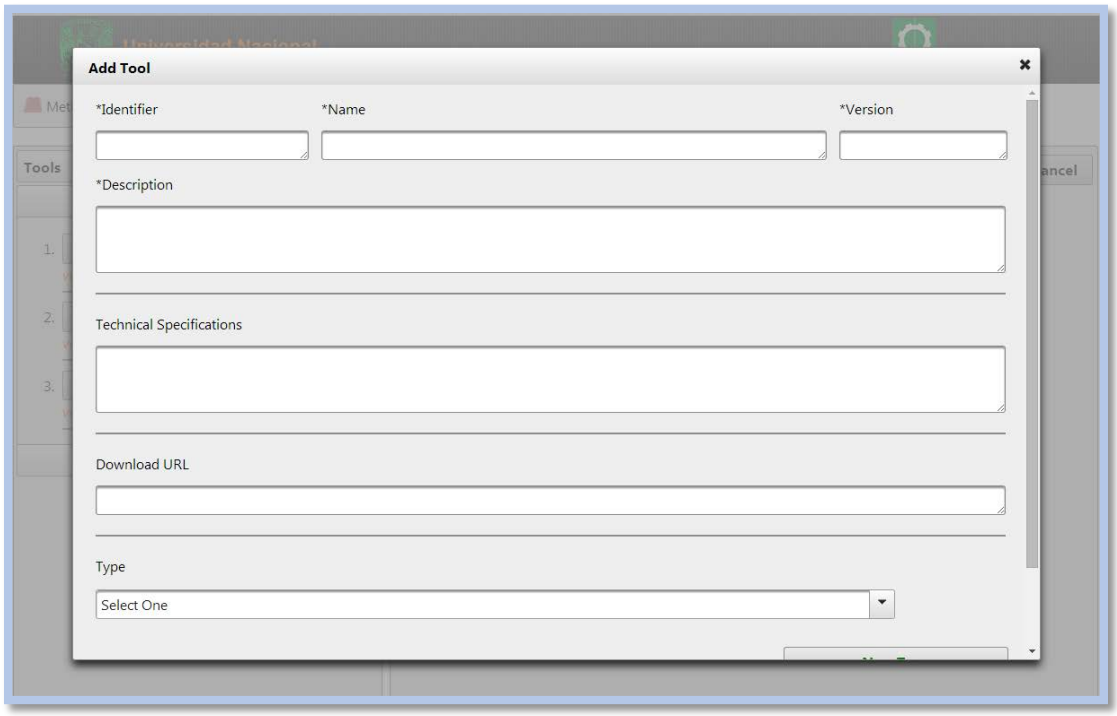


Figura 18. Add Tool.

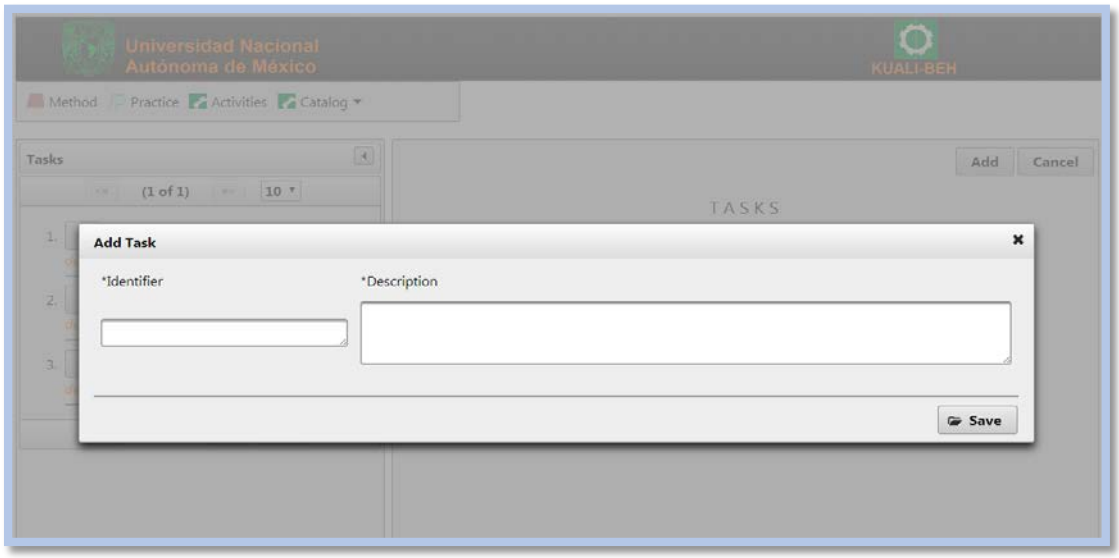


Figura 19. Add Task.

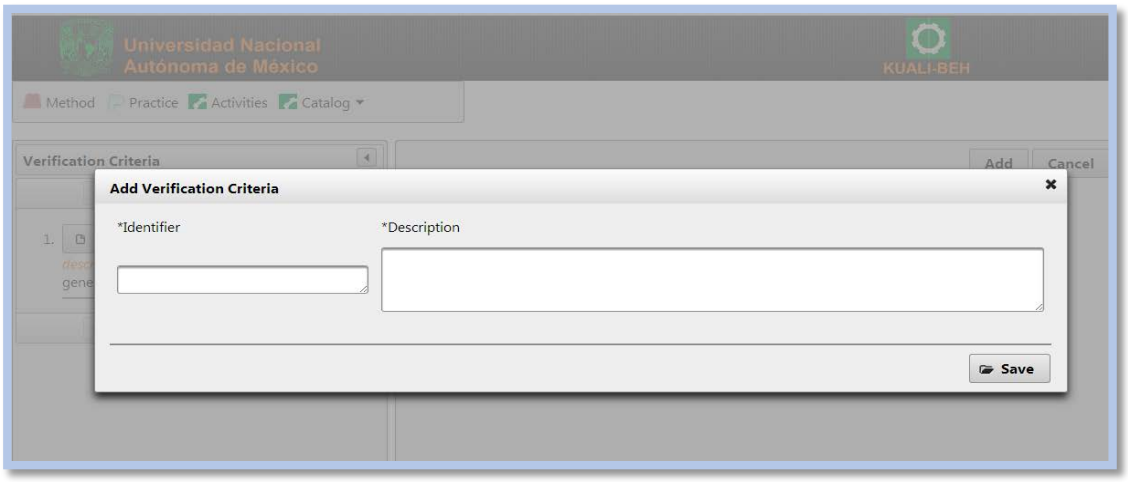


Figura 20. Add Verification Criteria.

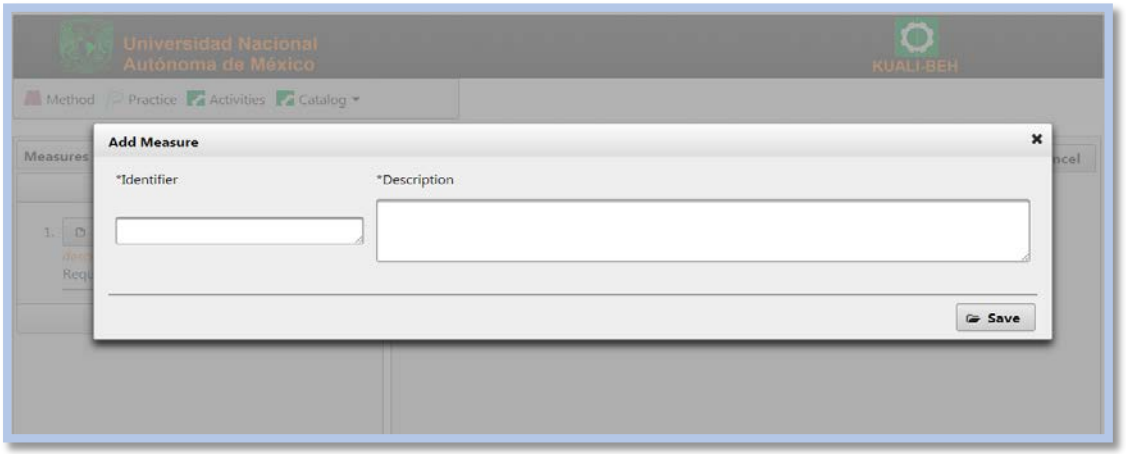


Figura 21. Add Measure.

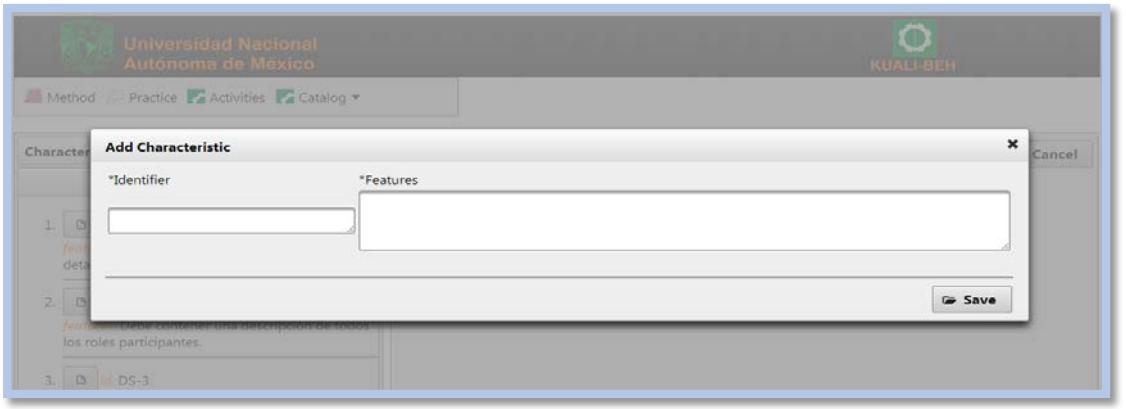


Figura 22. Add Characteristic.

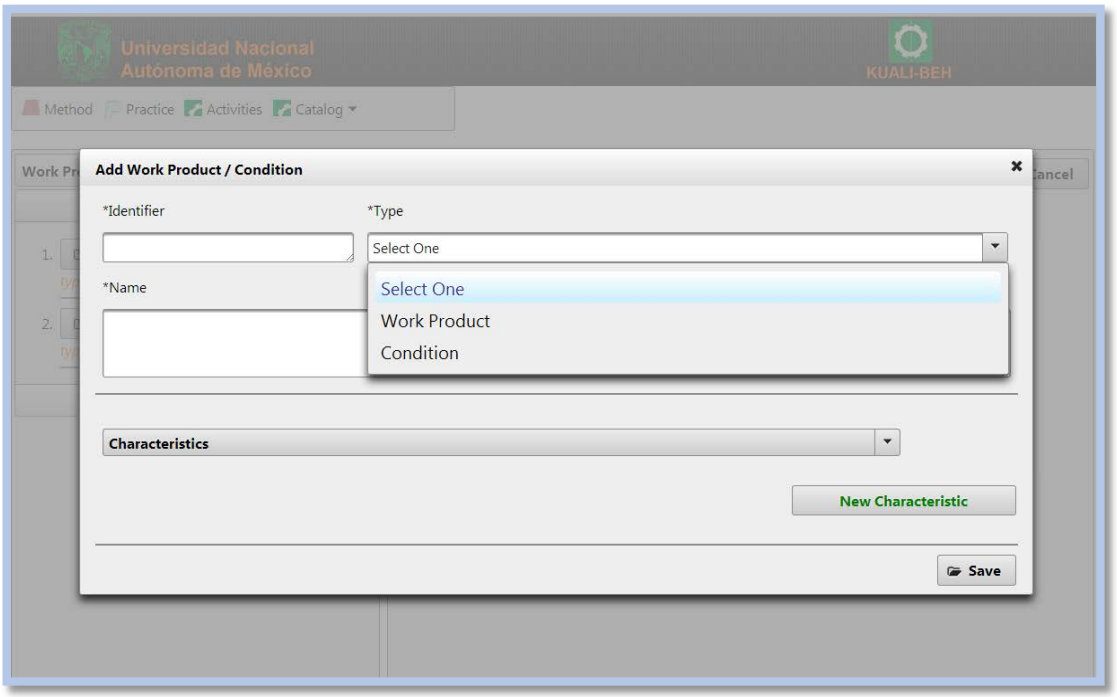


Figura 23. Add Work Product/Condition.

4.3.2.3 Interfaces Read del menú Catalog

Una vez que se tiene agregado algún catálogo dentro del sistema, es posible visualizar su información. Del lado izquierdo se muestra una lista con los datos que se han anexado previamente y un botón que al presionarlo desplegará en la pantalla principal la información correspondiente tal como se muestra en las figuras 24, 25, 26, 27, 28, 29, 30 y 31.

Cuando el sistema muestra la información contenida en algún catálogo, también activa dos botones más en el lado superior derecho (Modify y Delete), los cuales permitirán al practicante modificar o eliminar ésta información si así se requiere.

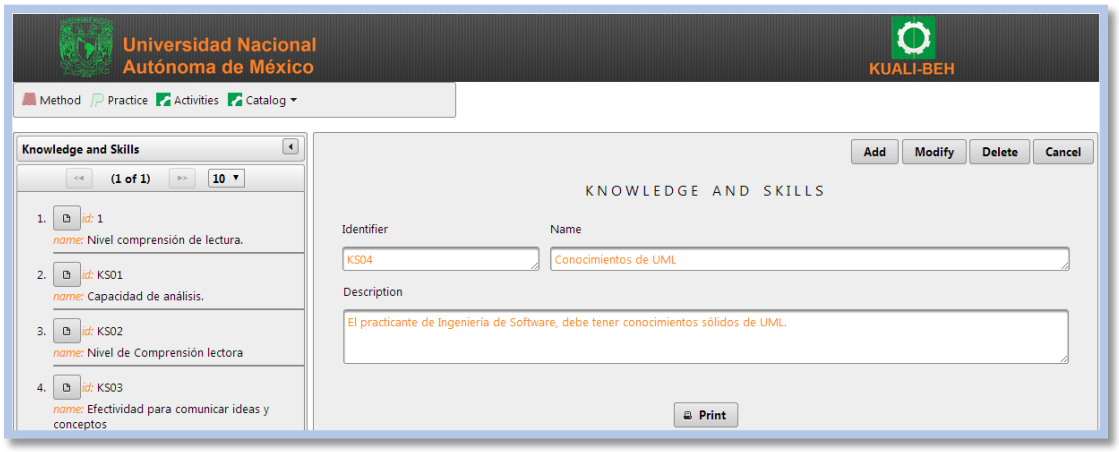


Figura 24. Read Knowledge and Skills.

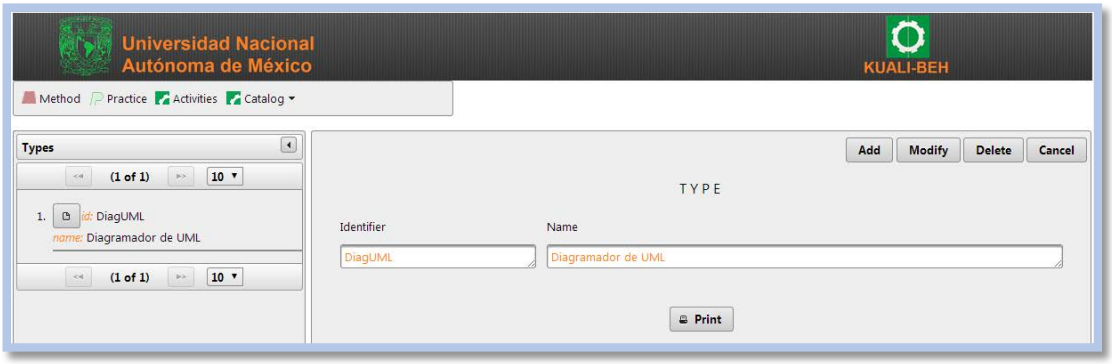


Figura 25. Read Type.

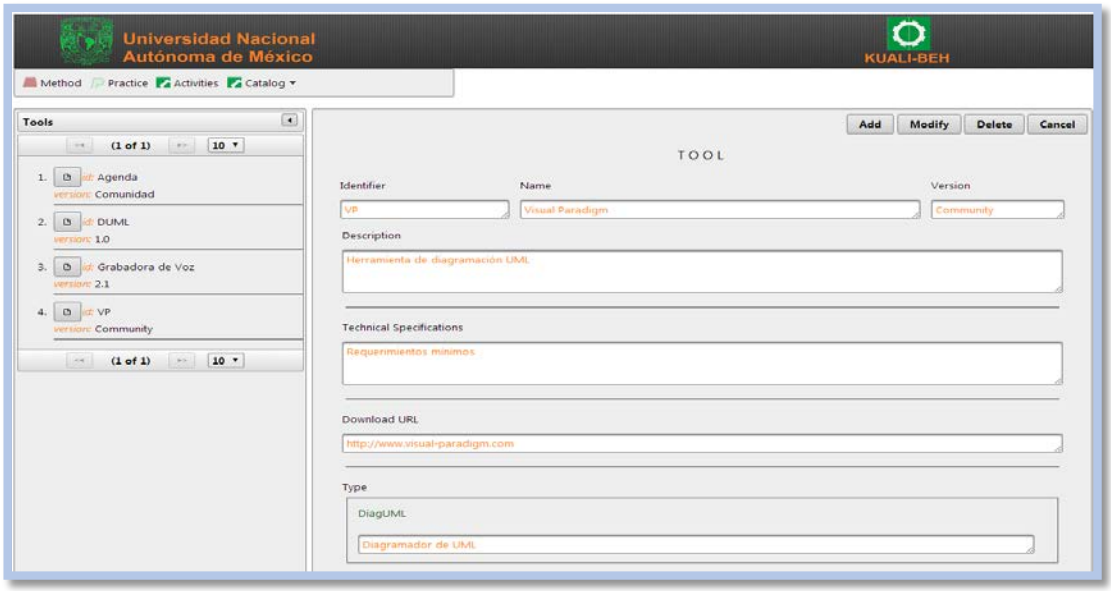


Figura 26. Read Tool.

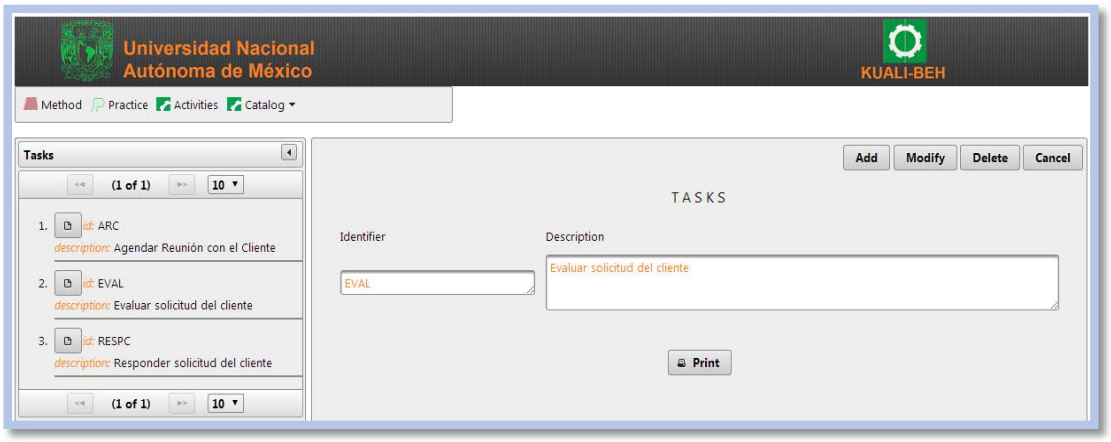


Figura 27. Read Task.

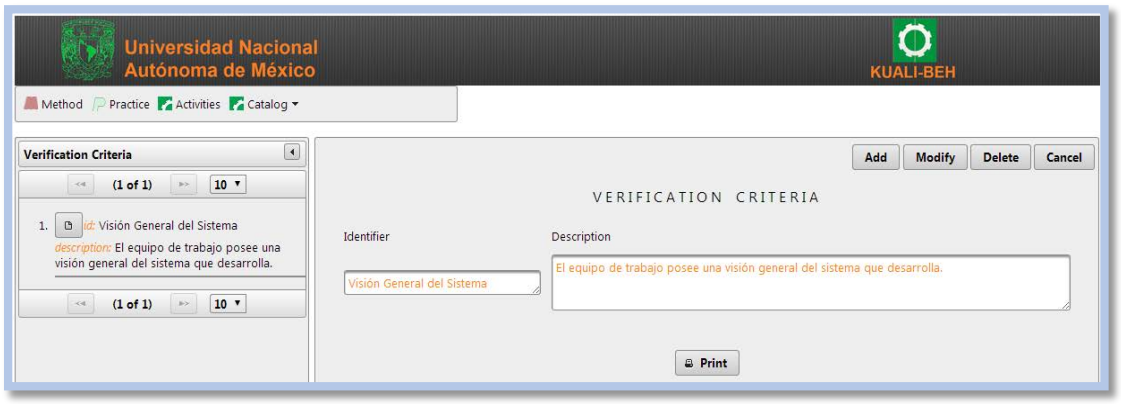


Figura 28. Read Verification Criteria.

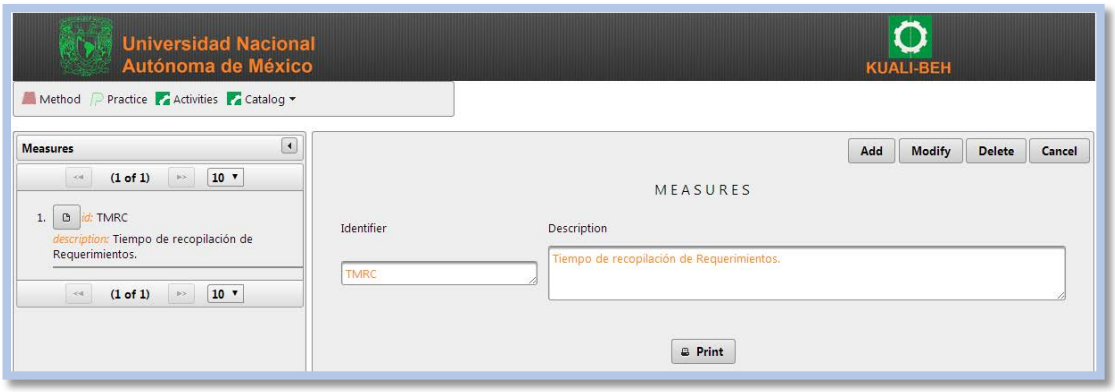


Figura 29. Read Measure.

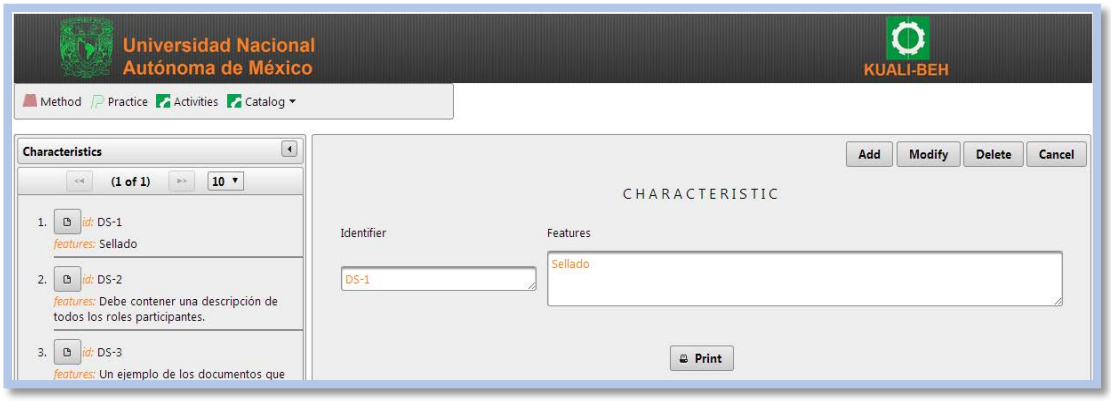


Figura 30. Read Characteristic.

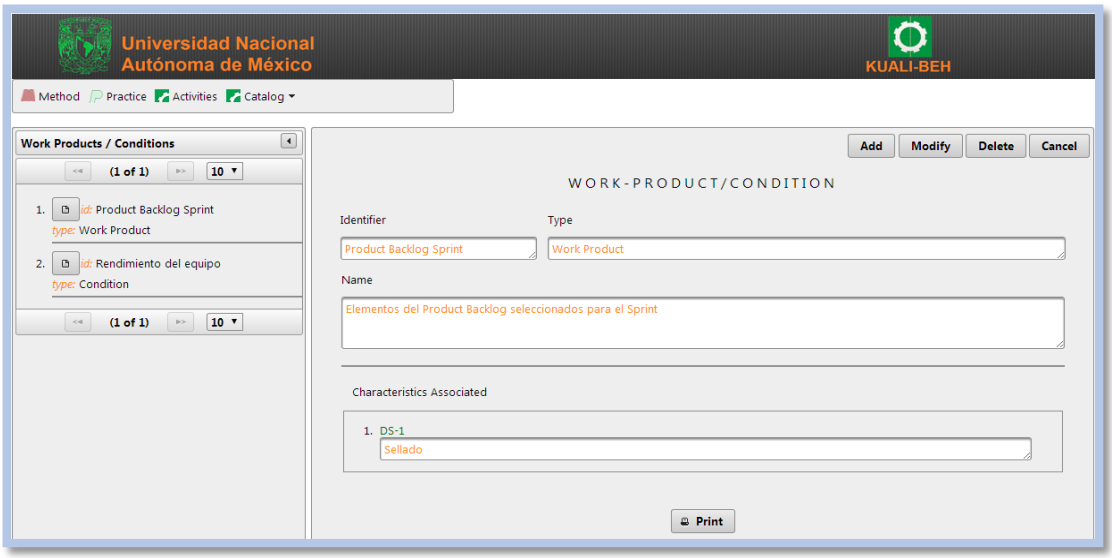


Figura 31. Read Work Product/Condition.

4.3.2.4 Interfaces Modify del menú Catalog

Al presionar el botón modify que se encuentra situado en la esquina superior derecha, el sistema mostrará un nuevo formulario con los datos contenidos de acuerdo al catálogo que corresponda, permitiendo al practicante modificar ésta información y actualizarla. Ver figuras 32, 33, 34, 35, 36, 37, 38, 39.

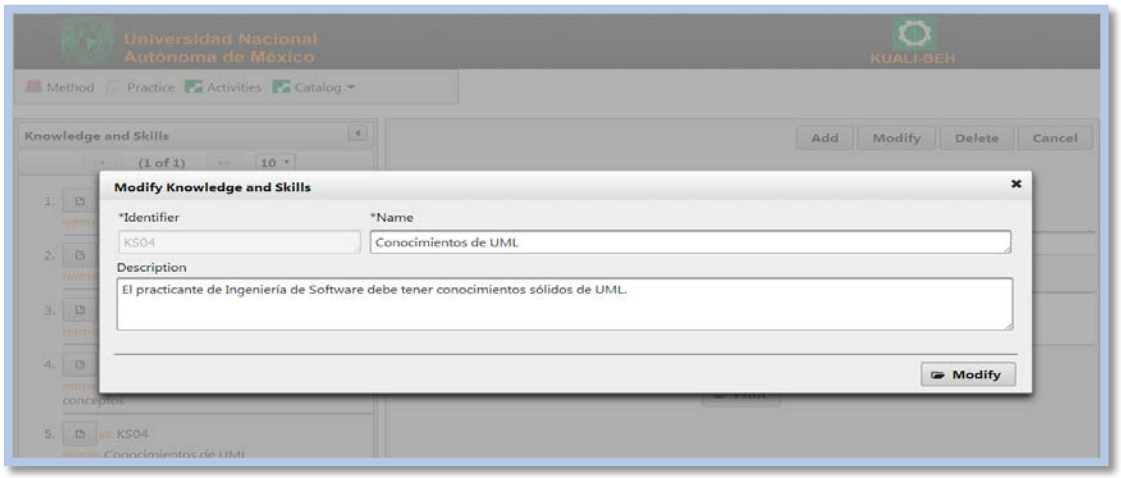


Figura 32. Modify Knowledge and Skills.

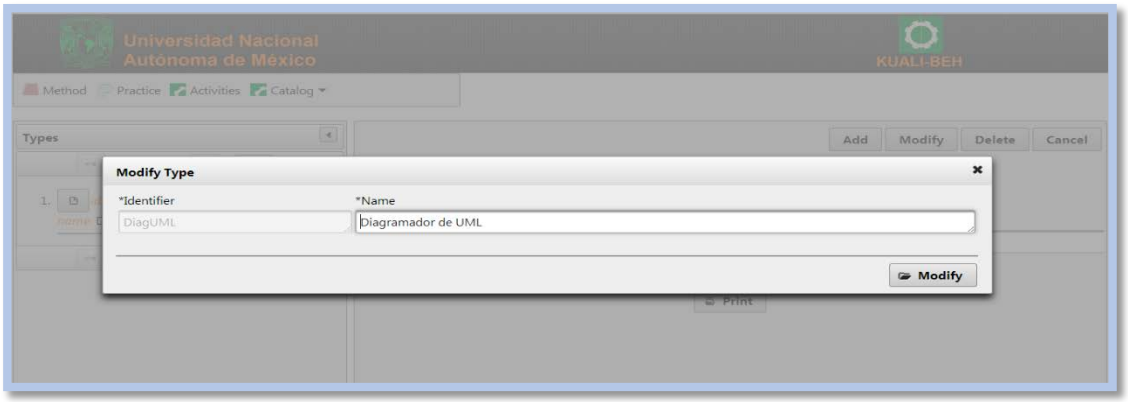


Figura 33. Modify Type.

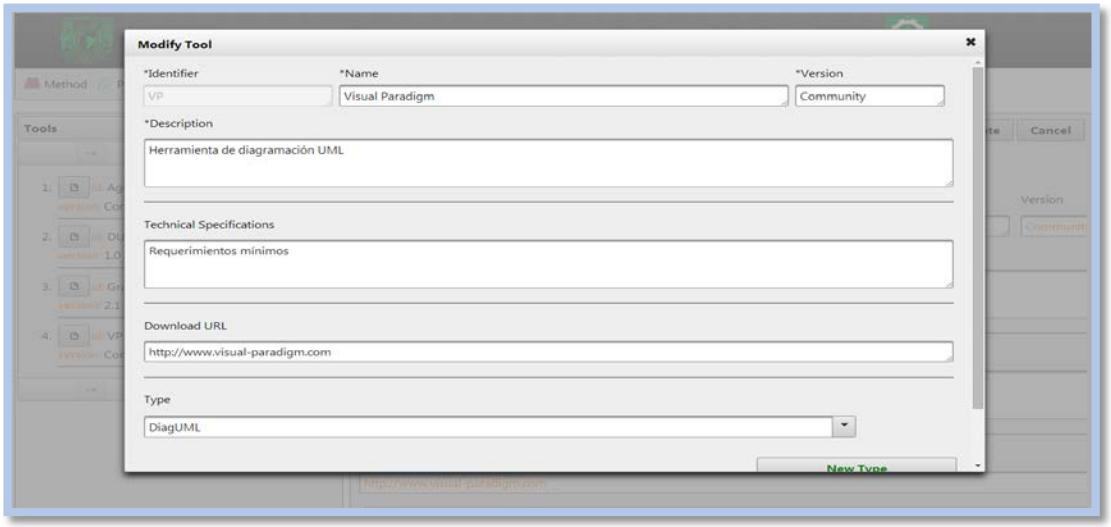


Figura 34. Modify Tool.

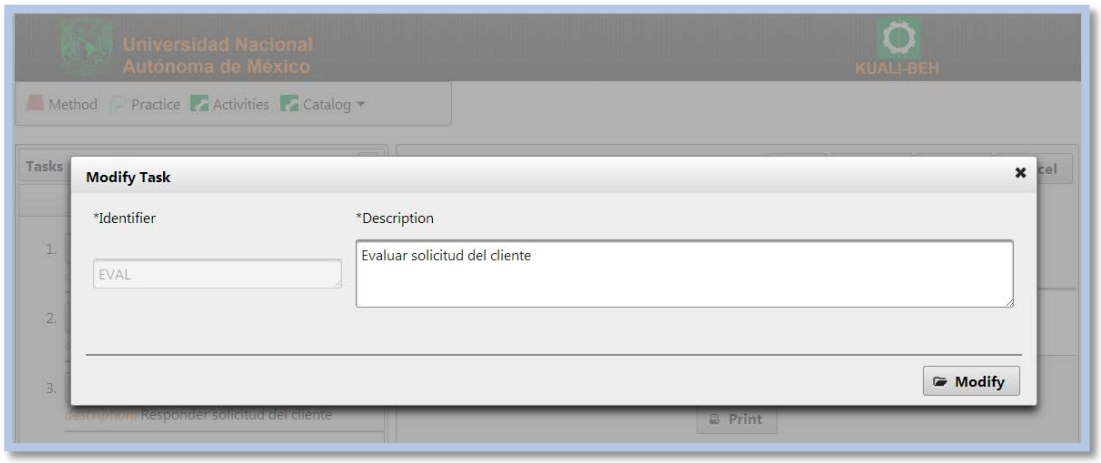


Figura 35. Modify Task.

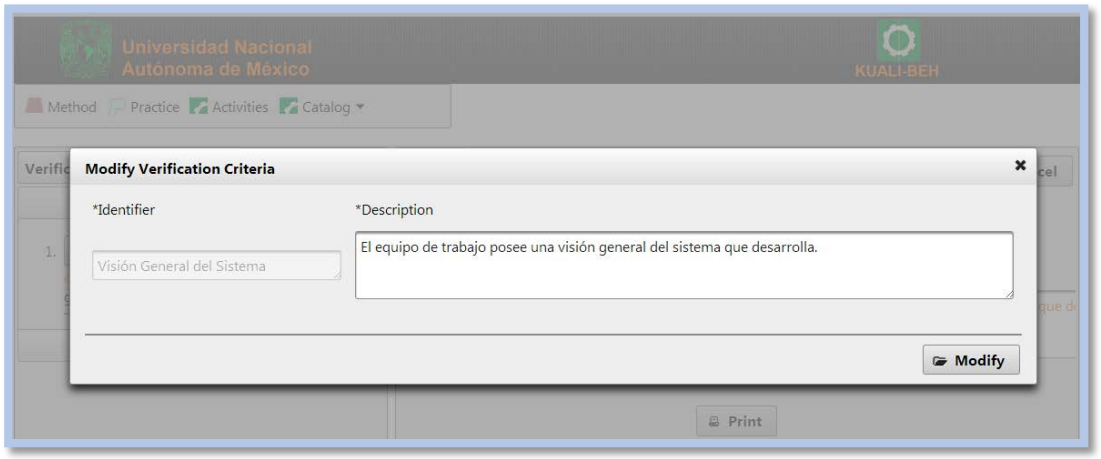


Figura 36. Modify Verification Criteria.

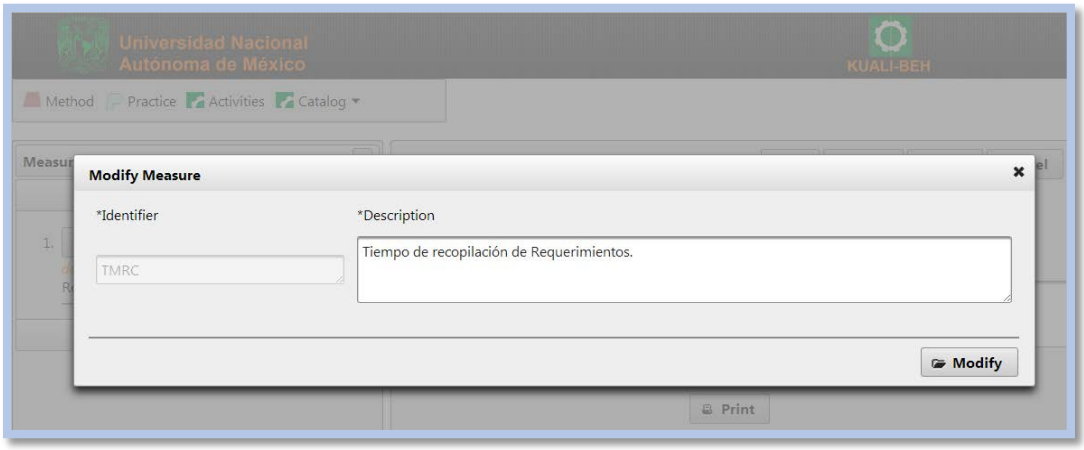


Figura 37. Modify Measure.

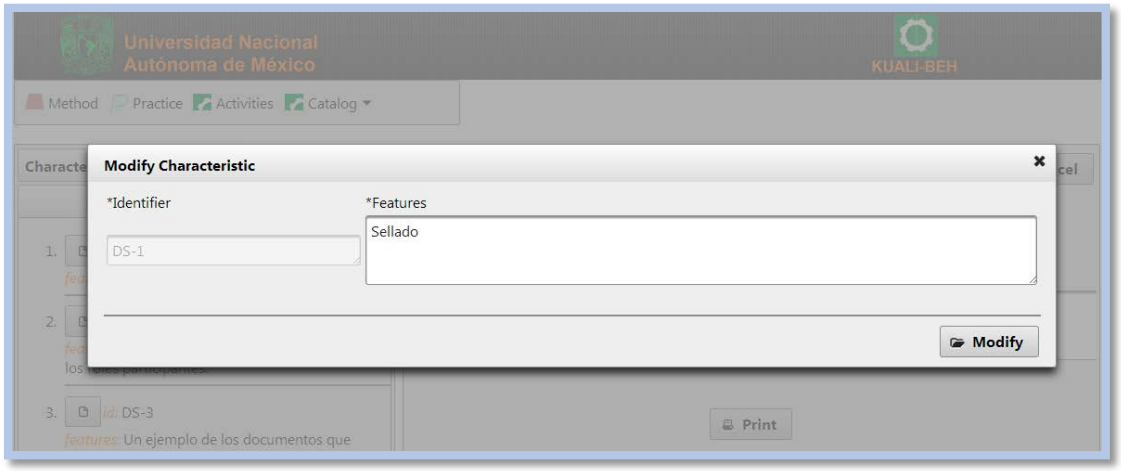


Figura 38. Modify Characteristic.

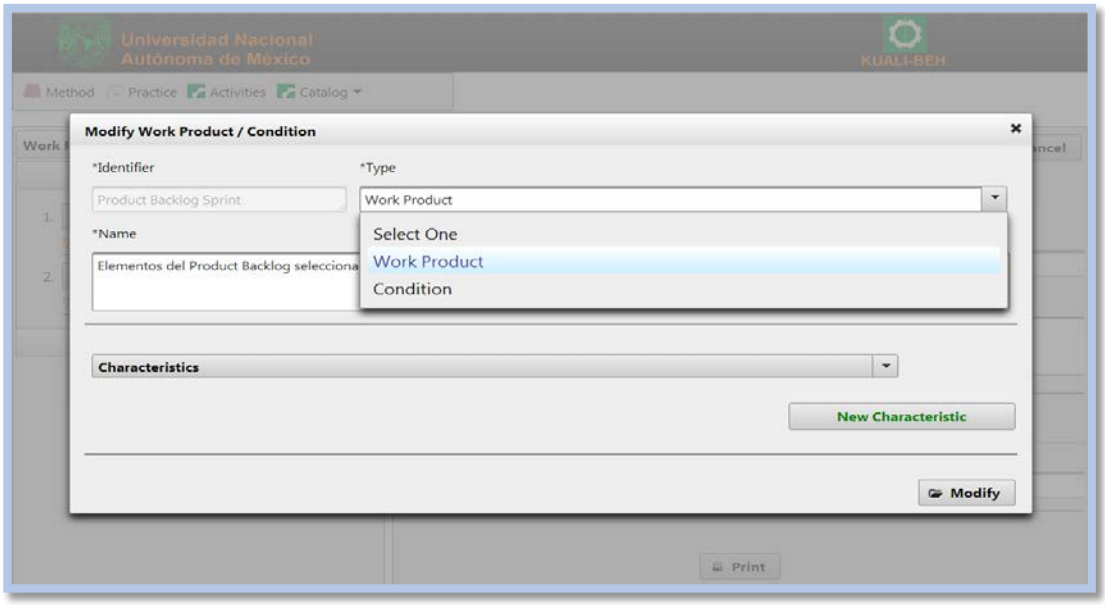


Figura 39. Modify Work Product/Condition.

4.3.3 Activities

Dentro del menú principal se encuentra la opción Activities, el cual, al seleccionarlo muestra la pantalla de la figura 40. En la parte superior muestra el nombre Activities correspondiente y los botones Add y Cancel del lado superior derecho.

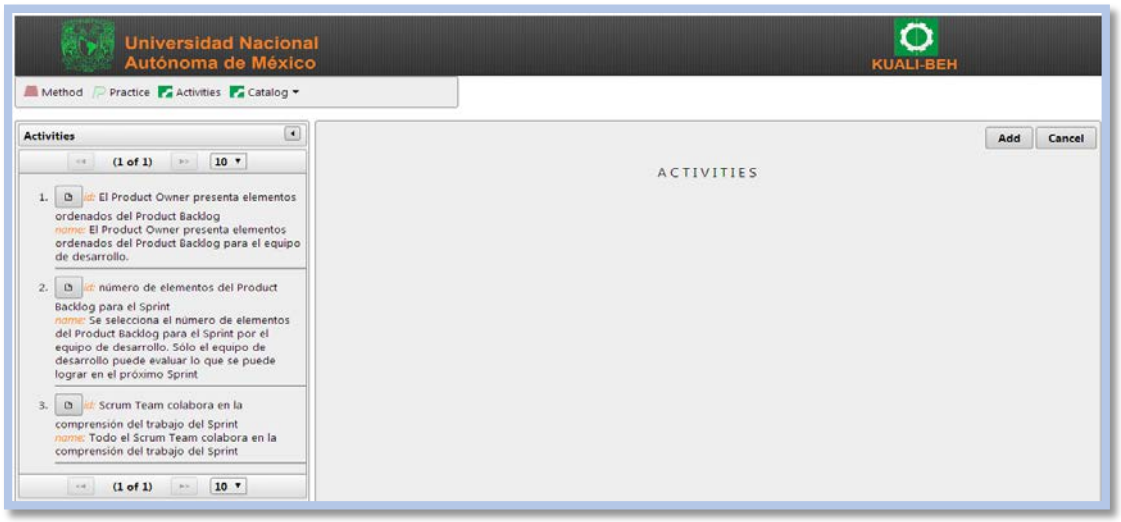


Figura 40. Activities.

4.3.3.1 Create Activity

Si el practicante desea agregar una nueva actividad al sistema, debe presionar el botón Add de la pantalla principal, al hacer esto, el sistema desplegará una ventana como el que se presenta en la figura 41, donde se observa un formulario pidiendo al practicante agregar la información necesaria.

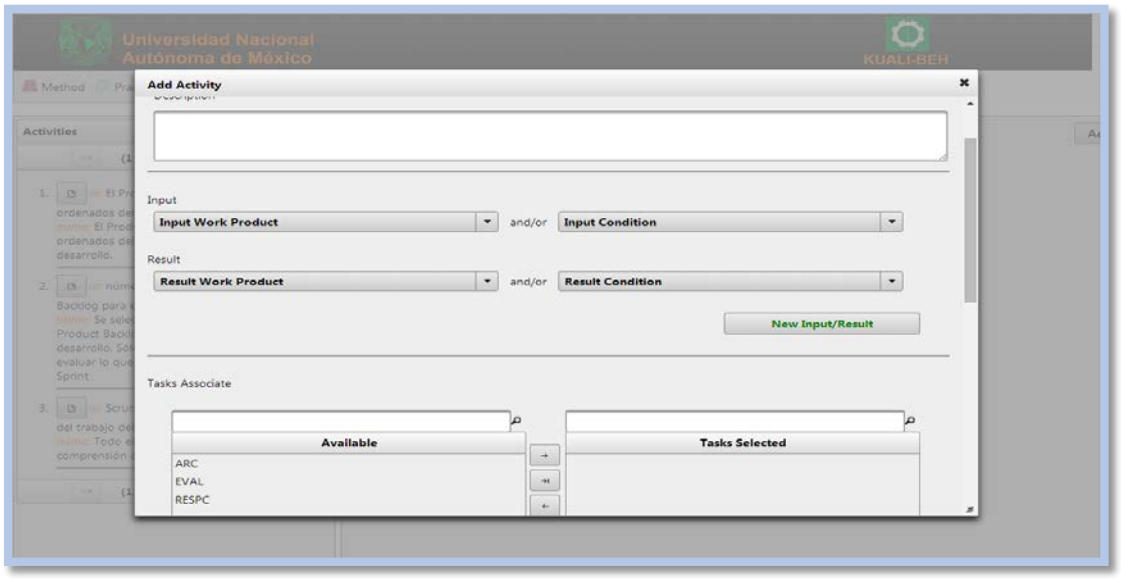


Figura 41. Add Activity.

4.3.3.2 Read Activity

El sistema muestra del lado izquierdo, una lista con las actividades que se encuentran dadas de alta en el sistema. El usuario podrá seleccionar la actividad requerida para poder consultar su información, tal como se muestra en la figura 42.

Figura 42. Read Activity.

4.3.3.3 Modify Activity

Después de haber seleccionado una actividad, el practicante podrá modificar la información de la misma presionando el botón modify, situado en la esquina superior derecha. El sistema mostrará una nueva ventana donde permitirá al practicante actualizar la información necesaria. Ver figura 43.

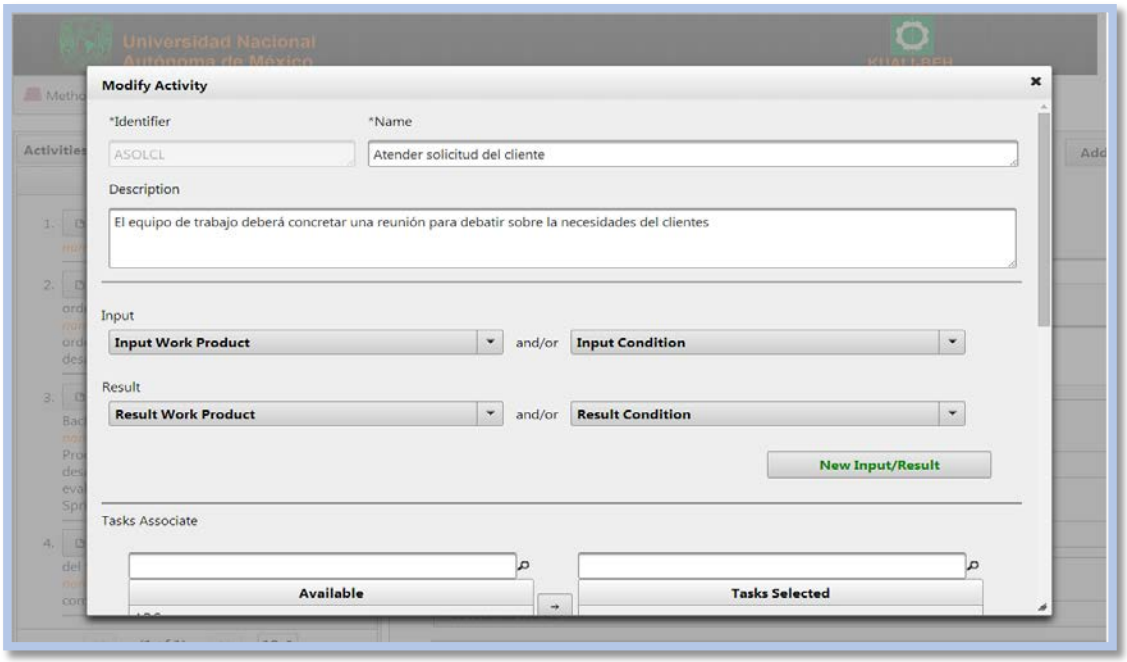


Figura 43. Modify Activity.

4.3.4 Practices

En la pantalla Practices mostrado en la figura 44 se presenta del lado izquierdo las prácticas que se tienen agregadas en la base de datos. En la esquina superior derecha estarán también los botones Add para poder crear nuevas prácticas y Cancel. Además se anexa otra lista del lado derecho, en el cual se podrán visualizar todas las actividades que se encuentran almacenadas en el sistema, con el objetivo de poder visualizar su contenido fácilmente.

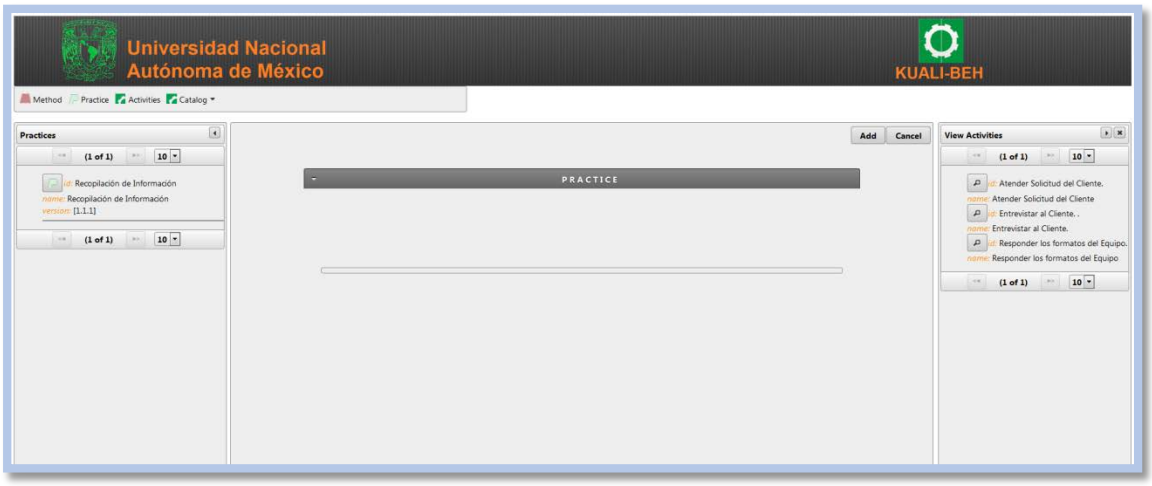


Figura 44. Practice.

4.3.4.1 Create Practice

Al presionar el botón Add, el practicante podrá dar de alta una nueva práctica. El sistema mostrará un formulario con los datos requeridos y una vez que el practicante escriba la información correspondiente y presione el botón Save, esta información quedará registrada dentro del sistema. Ver figura 45.

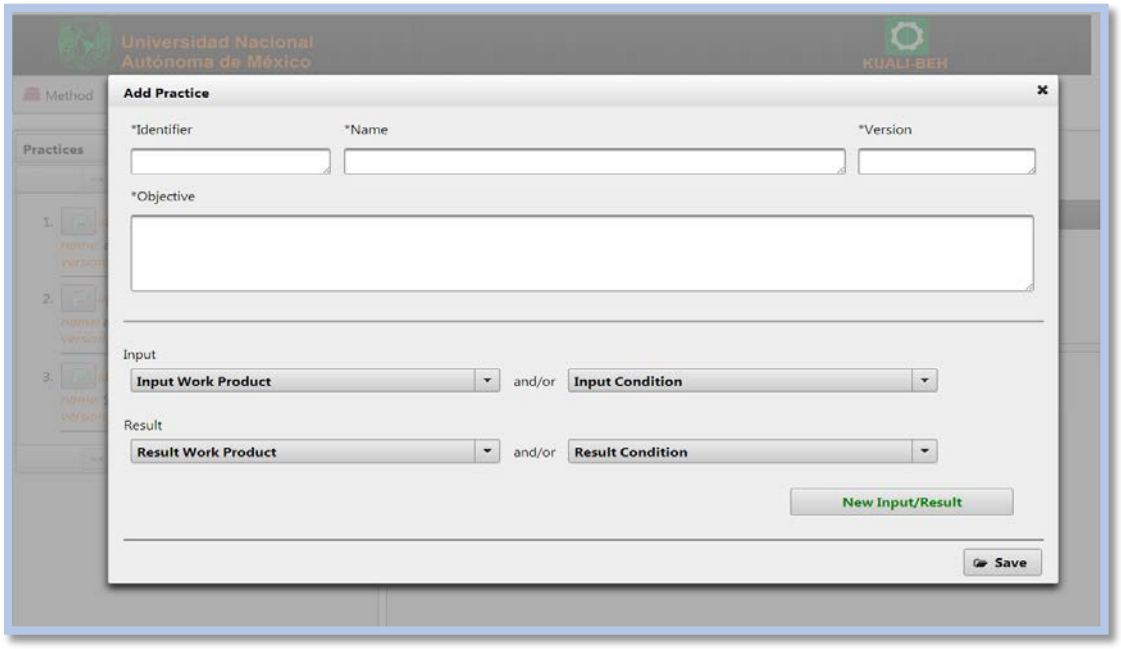


Figura 45. Add Practice.

4.3.4.2 Read Practice

Las prácticas contenidas dentro del sistema, se muestran en la parte izquierda de la pantalla, al seleccionar una de ellas el sistema desplegará toda su información. Al mismo tiempo activará los botones Modify y Delete en la parte superior derecha y mostrará el botón Add Guide, el cual le permitirá al practicante agregar una nueva guía y relacionarla a la práctica seleccionada. Ver figura 46.

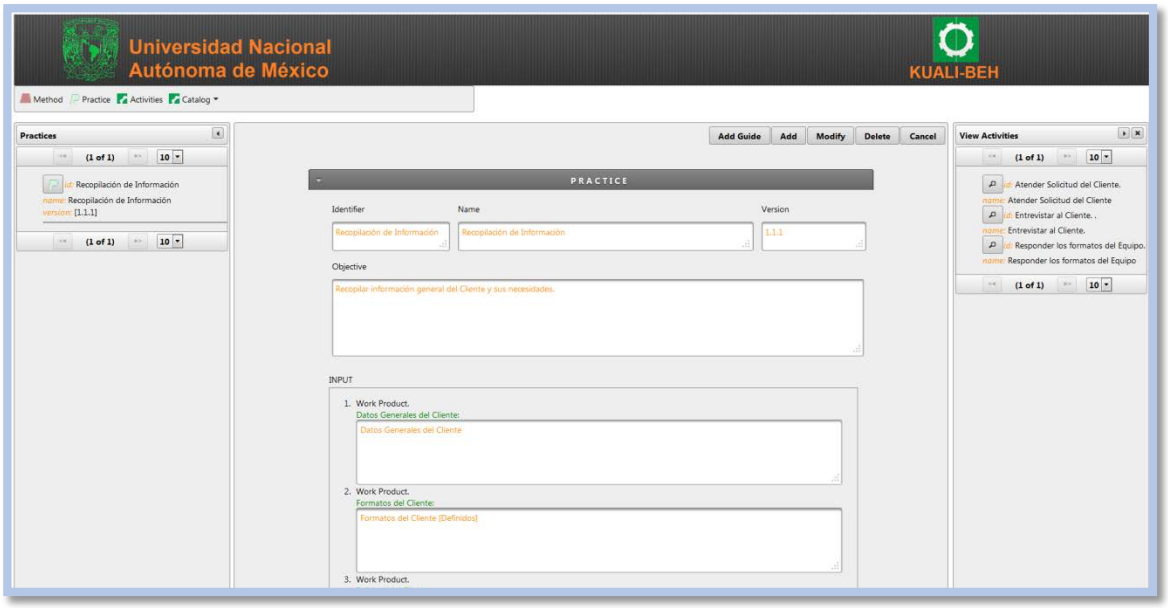


Figura 46. Read Practice.

4.3.4.2.1 Create Guide

Una vez que se ha seleccionado una práctica, se activarán los botones Modify y Delete, además de un nuevo botón llamado Add Guide, el cual nos permitirá agregar una nueva guía y asociarla a la práctica. Ver figura 47.

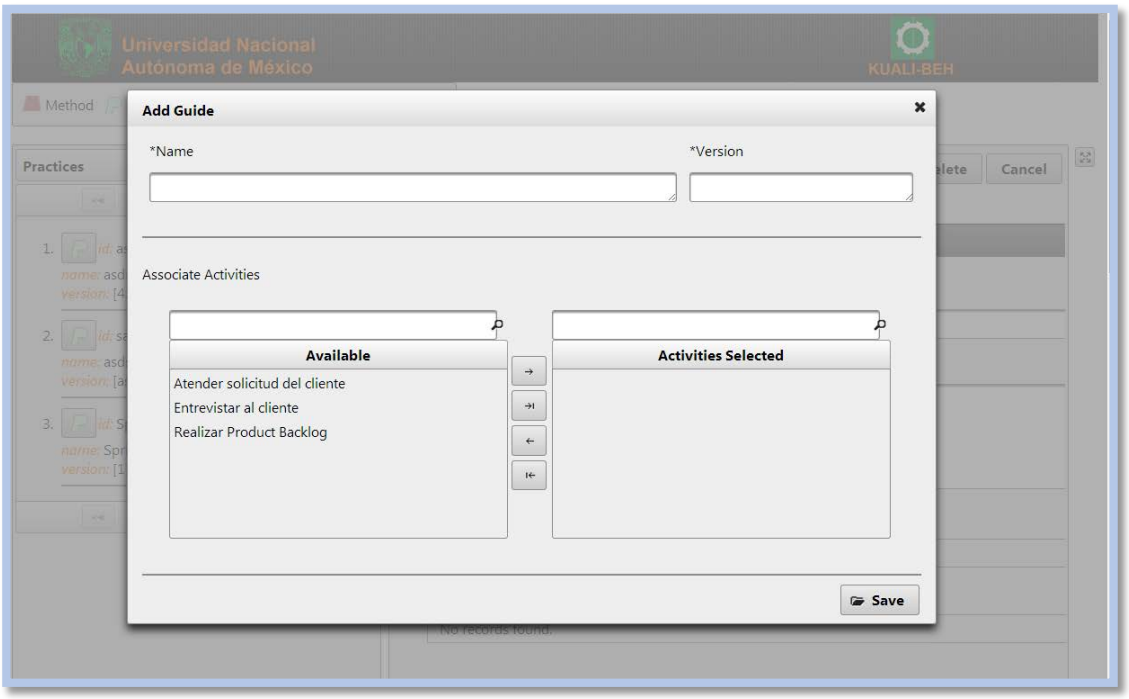


Figura 47. Add Guide.

4.3.4.2.2 Read Guide

Las guías asociadas a la práctica se presentan en una sección de la misma, como lo muestra la figura 48. El practicante podrá seleccionar cual desea ver en ese momento. Dentro de cada guía se podrán encontrar las actividades que tiene asociada y al seleccionarlas el sistema desplegará una ventana con toda su información.

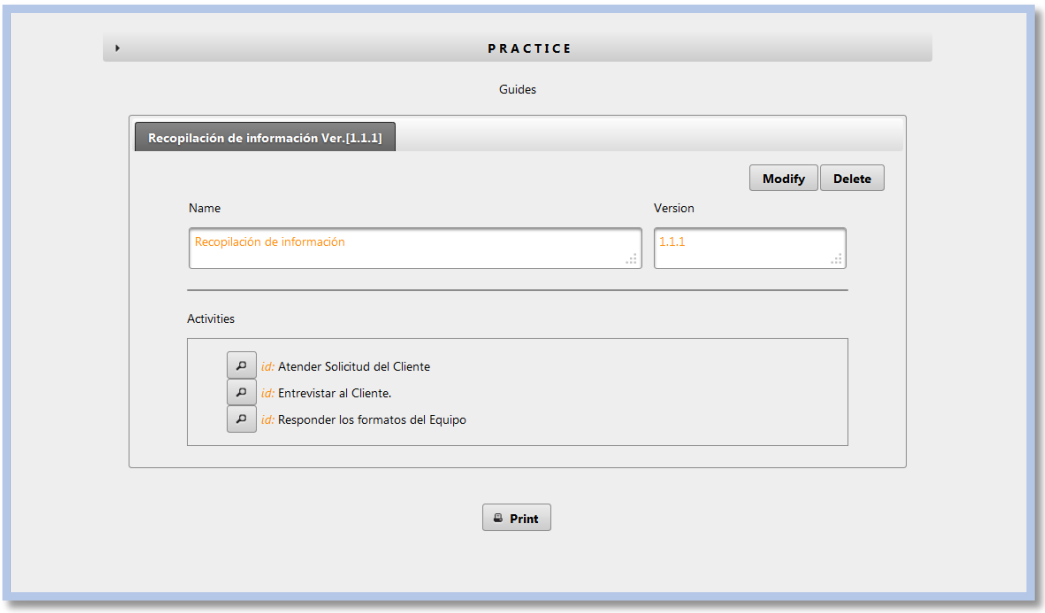


Figura 48. Read Guide.

4.3.4.2.3 Modify Guide

Dentro de cada guía se encuentra un botón Modify, el cual permite al practicante poder actualizar toda la información contenida en ella. Ver figura 49.

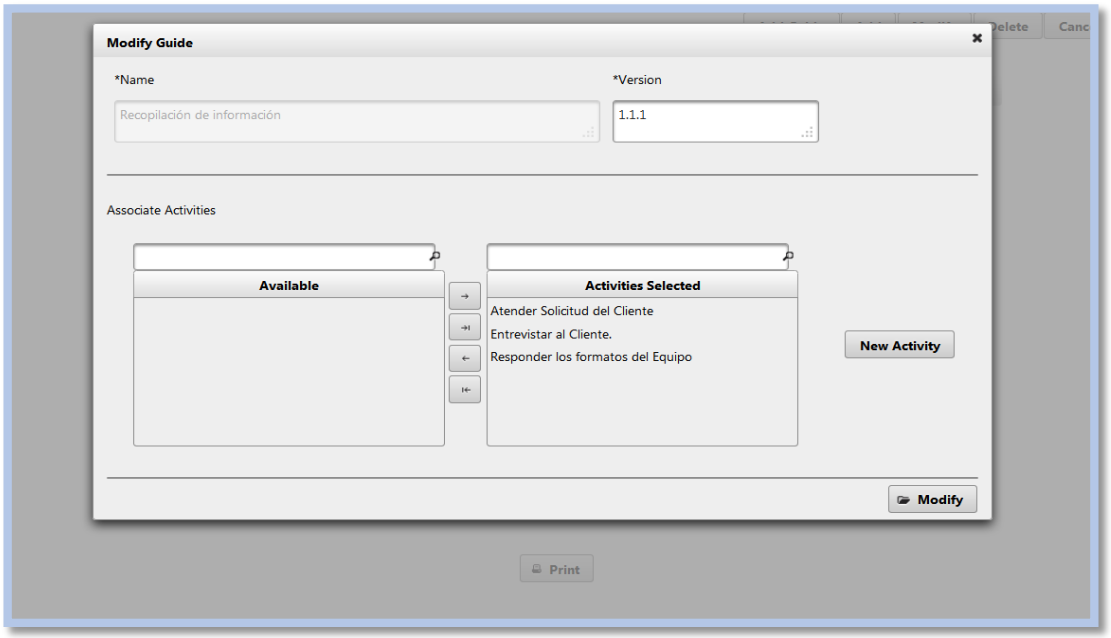


Figura 49. Modify Guide.

4.3.4.3 Modify Practice

La figura 50 muestra la ventana emergente Modify Practice, en la que el sistema presenta al practicante toda la información de la práctica, permitiéndole poder modificar la que necesite.

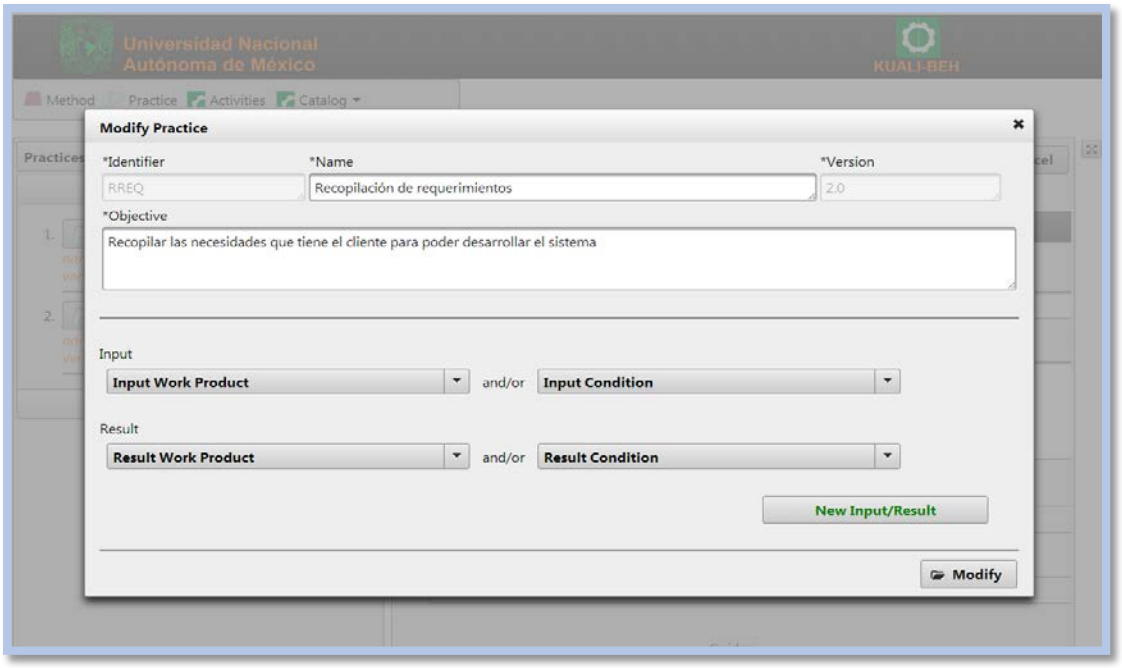


Figura 50. Modify Practice.

4.3.4.3.1 Change Version

Existen dos opciones para modificar una práctica, actualizar y actualizar con cambios de versión.

Al dar clic en el botón Modify de la ventana emergente, se mostrará una nueva ventana, en la cual se le preguntará al usuario si desea cambiar la versión de la práctica, en caso de no querer hacerlo, se dará clic en el botón No y el sistema guardará los valores sin alterar la versión de la práctica. Ver figura 51.

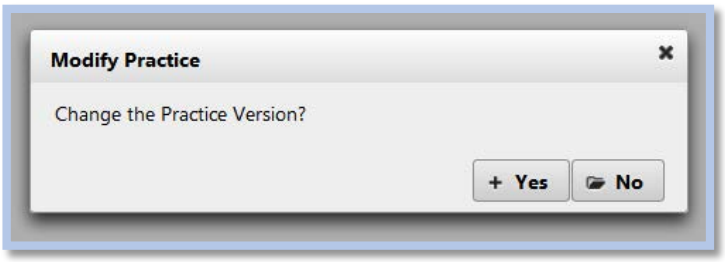


Figura 51. Change the Practice Version.

Si el practicante desea realizar cambios a la versión de la práctica, dará clic en el botón Yes y el sistema mostrará una nueva ventana para escribir la nueva versión, como se muestra en la figura 52.

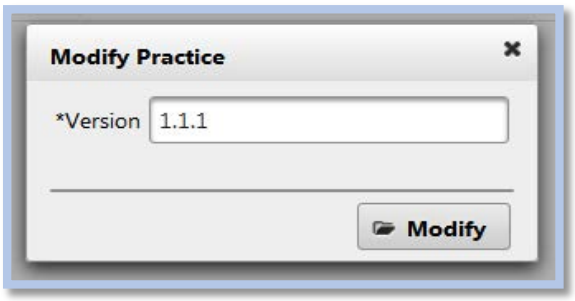


Figura 52. Version Practice.

4.3.4.4 Method

Al seleccionar la opción Method del menú principal, el sistema muestra la ventana correspondiente. Ver figura 53. Del lado izquierdo se puede observar una lista con los métodos que se encuentran agregados y del lado derecho otra lista con las prácticas existentes.

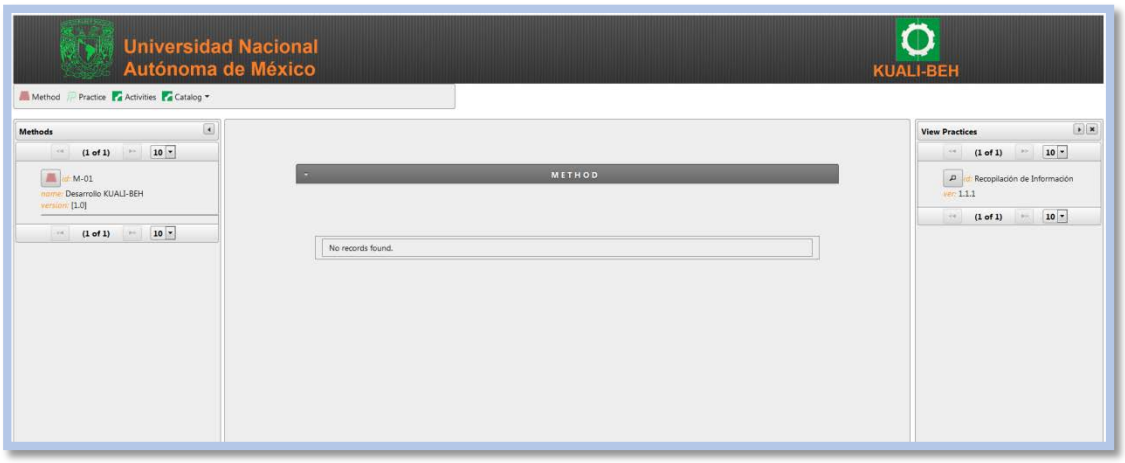


Figura 53. Method.

4.3.4.4.1 Create Method

El practicante podrá agregar un nuevo método a través de un formulario mostrado por el sistema, como el que se muestra en la figura 54, en él se escribirá la información requerida y al mismo tiempo se podrán elegir las prácticas necesarias para crear el método.

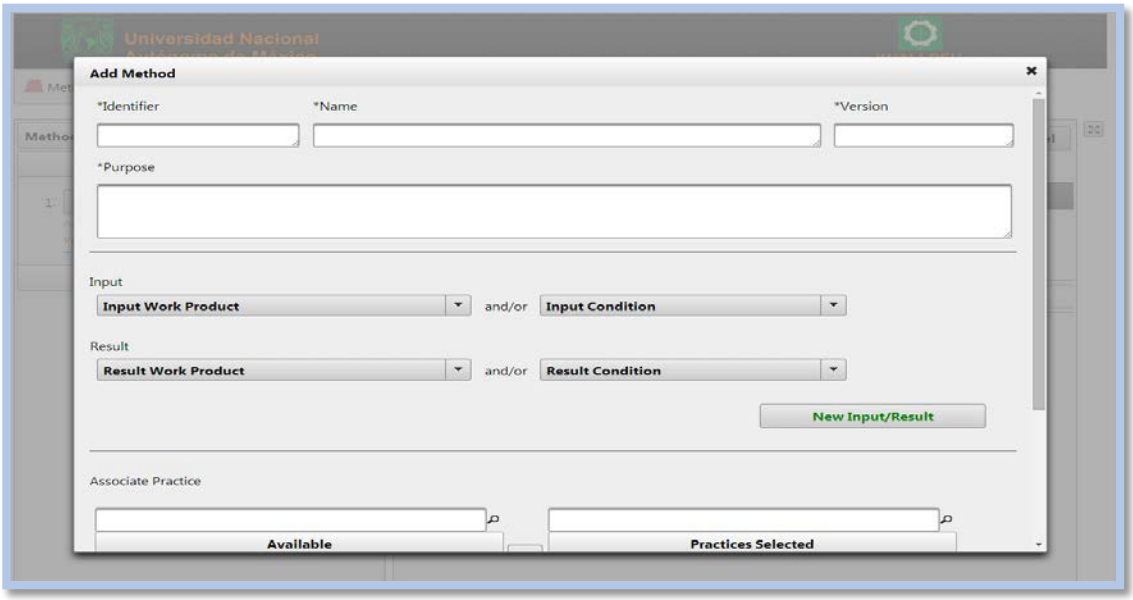


Figura 54. Add Method.

4.3.4.4.2 Read Method

La figura 55 muestra la pantalla que permite visualizar toda la información del método.

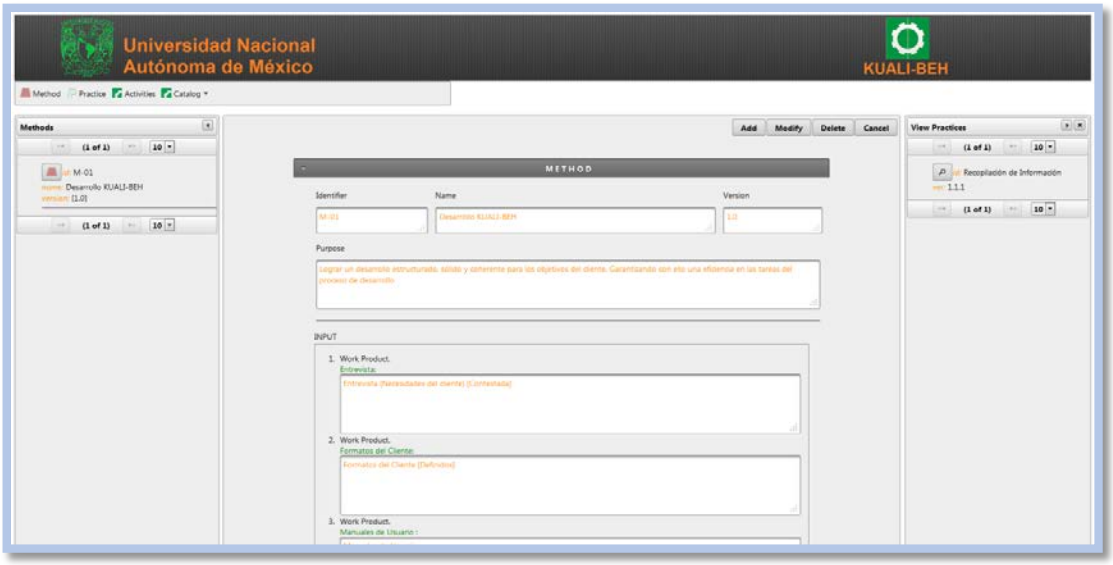


Figura 55. Read Method.

4.3.4.4.3 Modify Method

La opción Modify del método permite realizar cambios a los métodos seleccionados, incluyendo las prácticas asociadas y la versión. Ver figura 56.

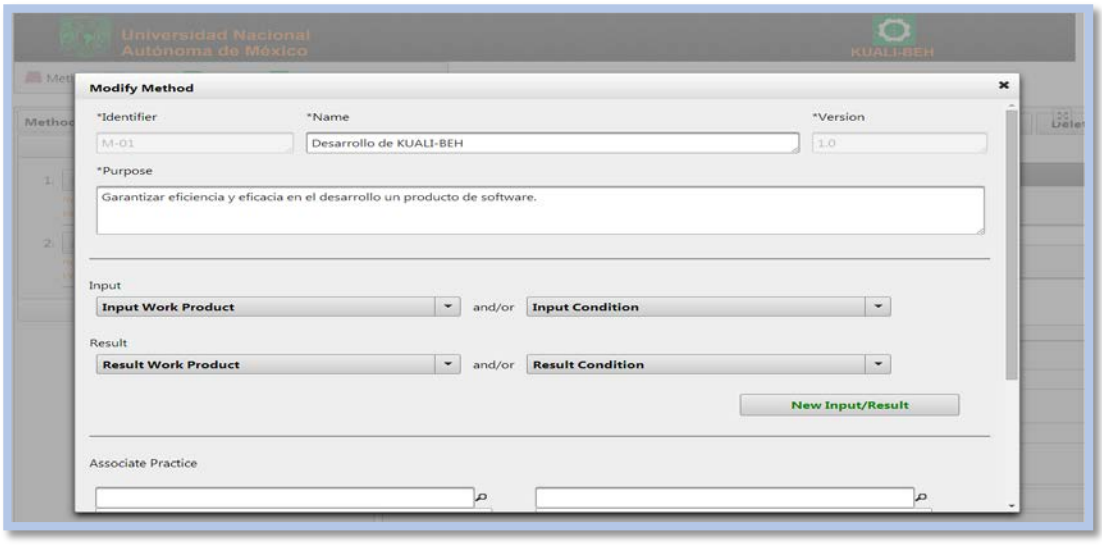


Figura 56. Modify Method.

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

APÉNDICE B

ANÁLISIS Y DISEÑO

En éste apartado se describen los resultados de las actividades de Análisis y Diseño del ‘Repositorio de métodos y prácticas de proyectos de software para KUALI-BEH’ mediante el uso de diagramas UML (Diagramas de Clase, Diagramas de Secuencia, Diagramas de Paquetes) y Modelo Entidad Relación en la representación de la base de datos.

ÍNDICE

ANÁLISIS Y DISEÑO

1. Introducción

En éste documento se describen los resultados de las actividades de Análisis y Diseño del 'Repositorio de métodos y prácticas de proyectos de software para KUALI-BEH'.

1.1 Propósito

El propósito de éste apéndice es documentar el análisis y diseño realizado en éste proyecto mediante el uso de diagramas UML

1.2 Alcance

Este documento mostrará el manejo de las ventanas contenidas en la interfaz gráfica del usuario mediante diagramas de clases. Los diagramas de secuencia detallarán la forma en que actuará el sistema con el usuario, así como la manera en que éste le responderá. Los diagramas de paquete presentan las arquitectura del sistema en general, y en particular se mostrarán aquellos que tienen que ver con el repositorio. El diagrama de Entidad Relación, describe de forma general la estructura de la base de datos en donde se almacenará toda la información que requieran los practicantes.

1.3 Definiciones, Acrónimos y Abreviaturas

La tabla 1 muestra la definición de algunos conceptos manejados en este documento.

Tabla 1. Definición de Conceptos.

Término	Definición
Diagrama de Secuencia	Un diagrama de secuencia es una forma de diagrama de interacción que muestra los objetos como líneas de vida a lo largo de la página y con sus interacciones en el tiempo representadas como mensajes dibujados como flechas desde la línea de vida origen hasta la línea de vida destino.
Diagrama de Paquetes	Muestra cómo un sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones.

1.4 Referencias

[10] Alberto Tapia Durán, tesis de maestría en ingeniería: Arquitectura de software para el entorno computacional de KUALI-BEH (2014).

[26] EjemplosTIW. Universidad Carlos III de Madrid. http://www.lab.inf.uc3m.es/~a0080802/RAI/mvc.html. 17/11/2013

[20] PostgreSQL. http://www.postgresql.org.es/sobre_postgresql, 30/9/2013

2. Representación de la Arquitectura

Este software será creado como una aplicación web, por ésta razón podrá ser utilizado por cualquier persona que cuente con conexión a internet y un navegador web (Chrome o Mozilla).

El diseño de este sistema se basará en la tecnología orientada a objetos y en particular hará uso del patrón de arquitectura Modelo-Vista-Controlador (MVC) [26]. Como lenguaje de programación se hará uso principalmente de Java, HTML y JavaScript, además utilizará una base de datos relacional implementada en PostgreSQL [20].

En la figura 1 se muestra la vista general de la arquitectura del sistema.

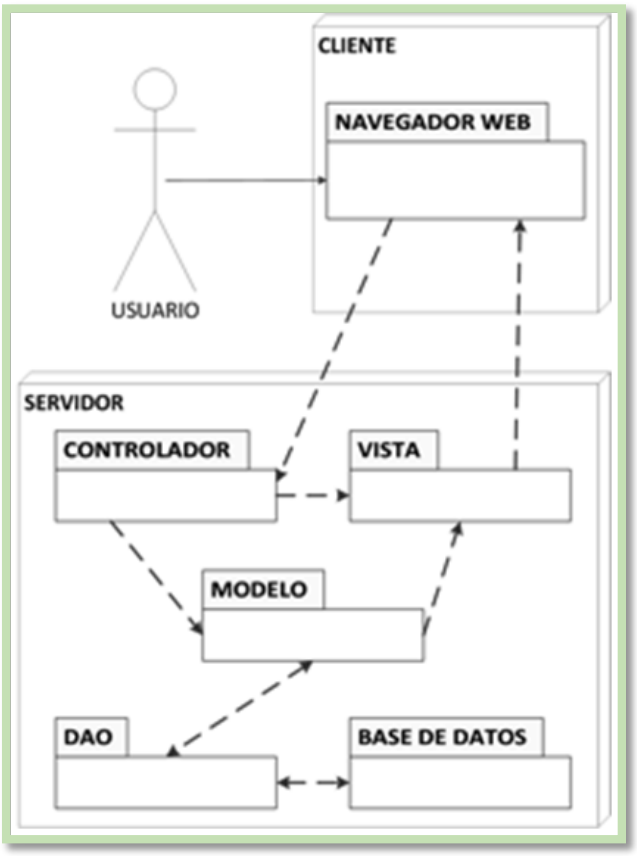


Figura 1. Diagrama de Distribución [10]

3. Modelo del Análisis

A continuación, en las figuras 2, 3 y 4, se muestran los paquetes que conforman el sistema completo del proyecto Entorno Computacional para KUALI-BEH, estos diagramas fueron tomados de la tesis de maestría en ingeniería: Arquitectura de software para el entorno computacional de KUALI-BEH [6].

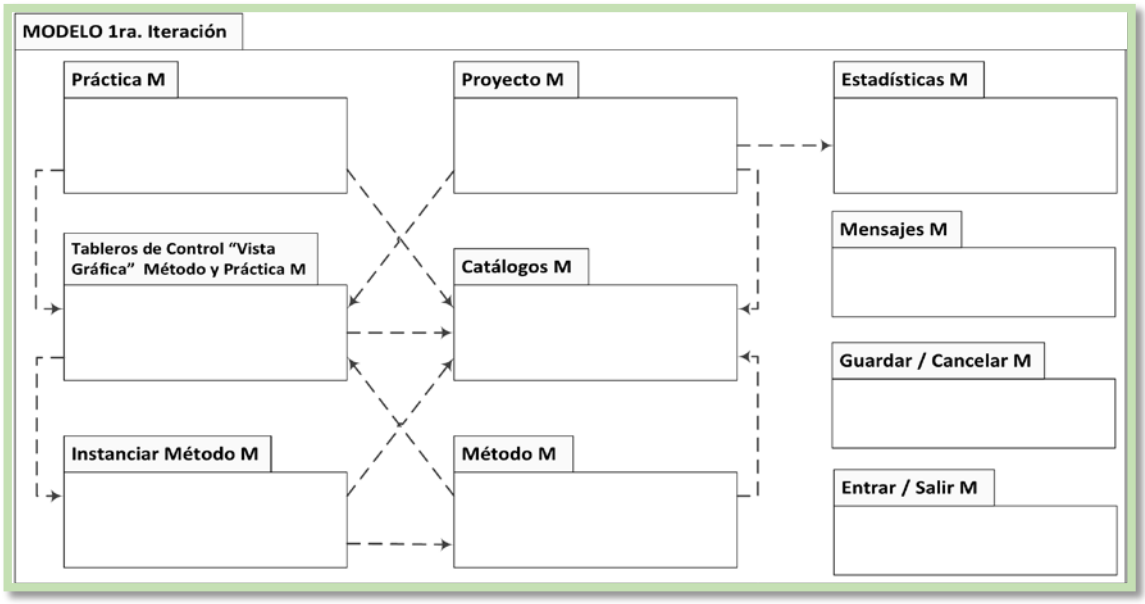


Figura 2. Modelo 1a Iteración [10]

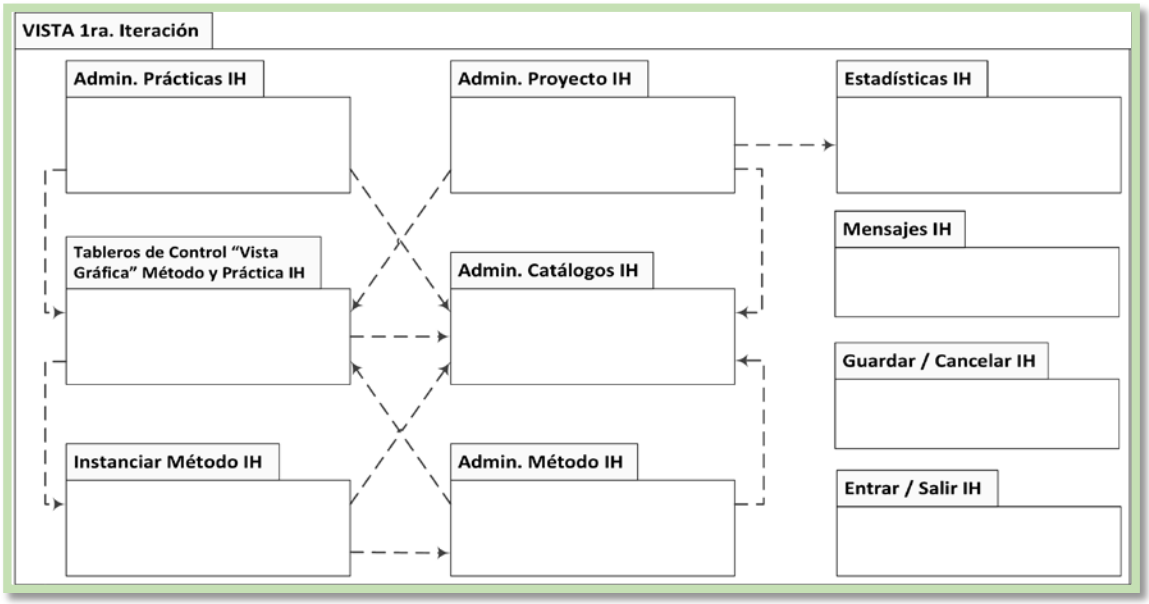


Figura 3. Vista 1a Iteración[10]

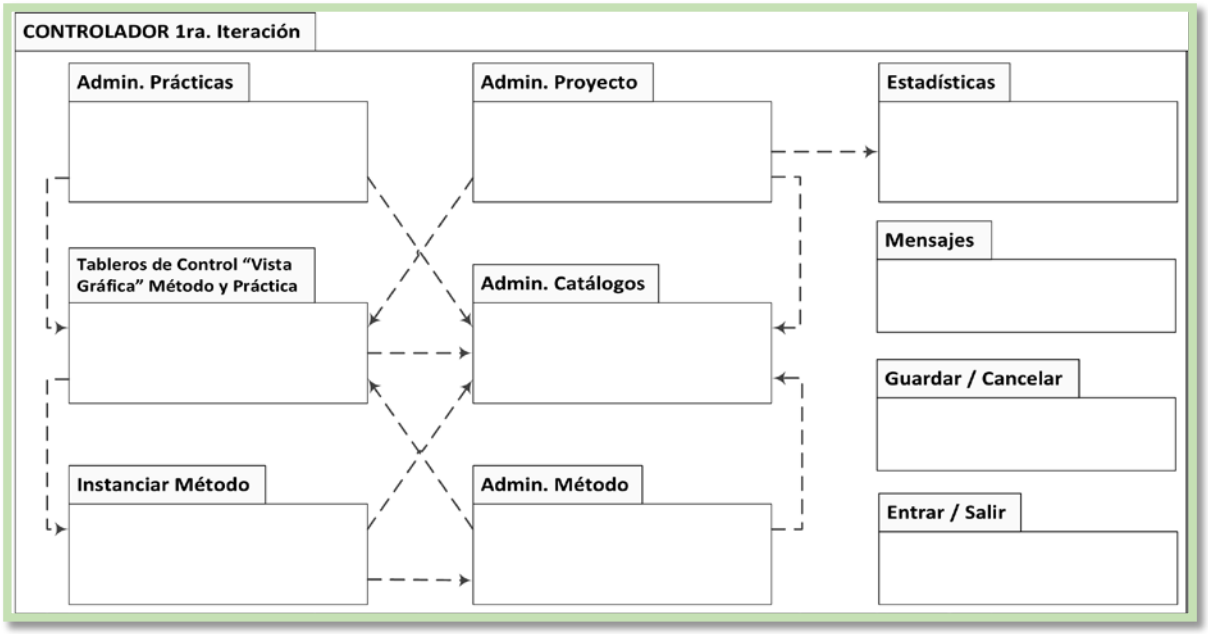


Figura 4. Controlador 1a Iteración[10]

Los paquetes correspondientes a Prácticas, Métodos, Catálogos y Guías, son los que se encuentran altamente relacionados a éste trabajo, ya que corresponden a la Vista Estática de KUALI-BEH.

4. Clases del Análisis

Esta sección contendrá los diagramas que describen el comportamiento estático del sistema y los tipos de relaciones existentes entre las clases. Se generará un diagrama de clases por cada paquete que contenga el sistema.

4.1 Identificación de las clases

Esta sección contiene los diagramas que describen el comportamiento estático del sistema y los tipos de relaciones existentes entre las clases. Se generó un diagrama de clases por cada paquete que contiene el sistema.

4.1.1 Clases de la Interfaz

La figura 5 muestra alguna de las clases y paquetes utilizados en la programación de la interfaz del sistema.

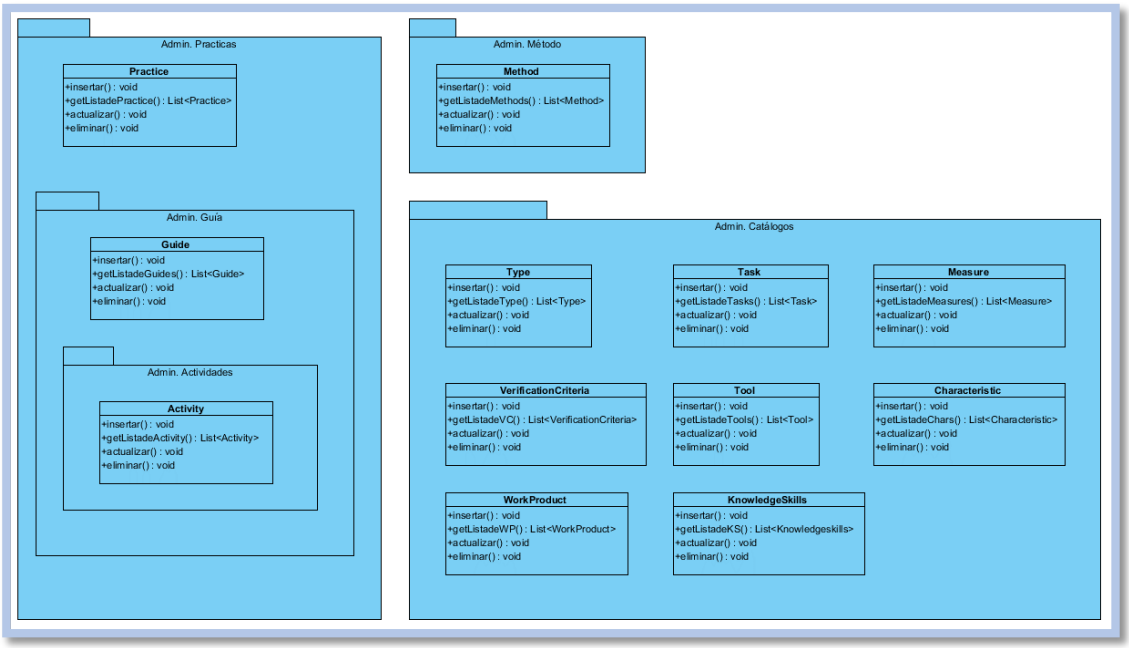


Figura 5. Clase de la Interfaz

4.1.2 Clases de entidad

Las clases de entidad manejados en el sistema se muestran en la figura 6.

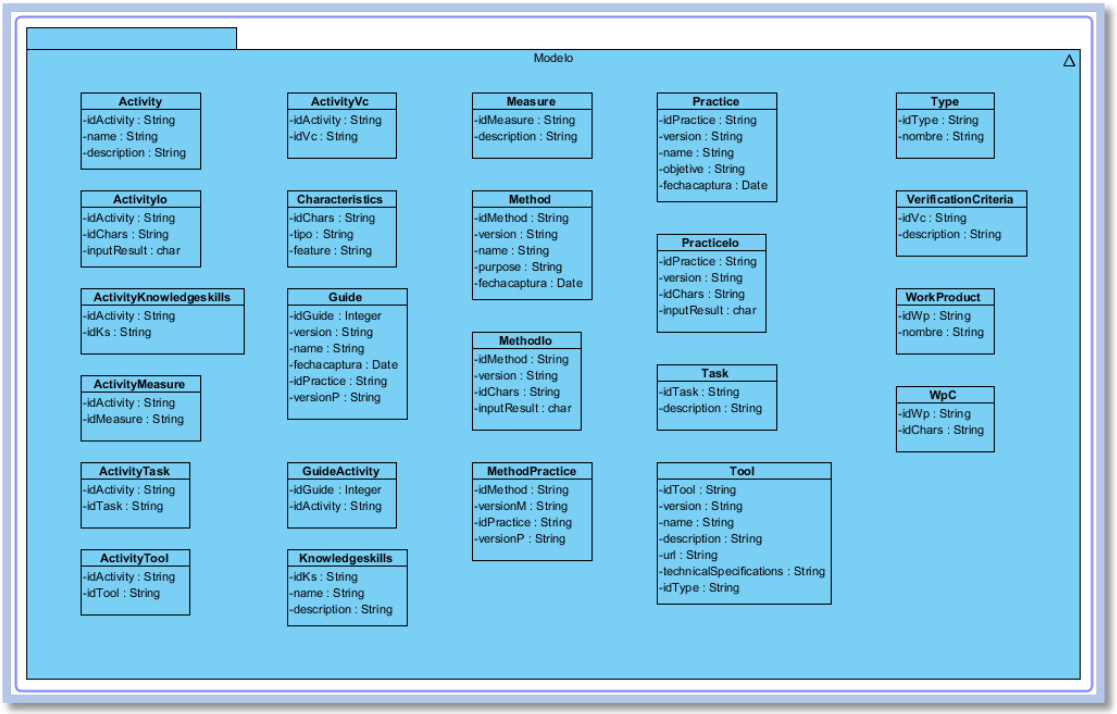


Figura 6. Clases de Entidad

4.1.3 Clases del control

La figura 7 muestra las clases de control, las cuales responden a órdenes hechas comúnmente por el usuario e invoca peticiones al modelo.

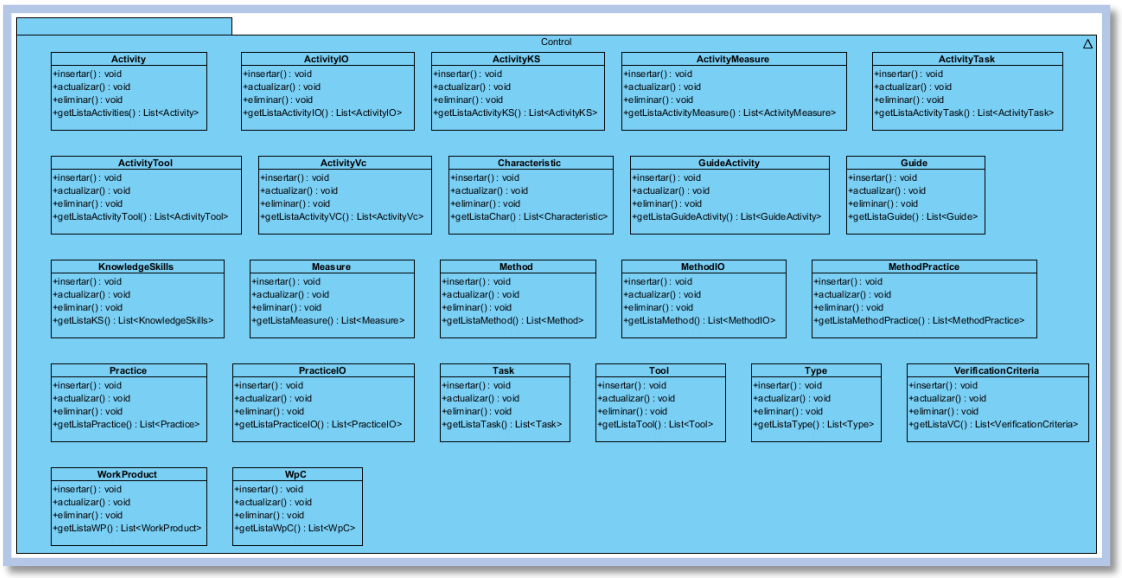


Figura 7. Clases de Control

5. Realización de Caso de Uso - Análisis

En ésta sección se incluyen los diferentes Diagramas de Secuencia referentes a cada caso de uso.

5.1 Diagramas de Secuencia

Los diagramas de secuencia muestran la forma en que el o los usuarios y los objetos de un sistema se comunican o interaccionan.

Las figuras que se presentan a continuación muestran diferentes diagramas de secuencia correspondientes a cada caso de uso perteneciente al sistema.

5.1.1 Create Catalog

La figura 8 muestra el diagrama de secuencia create catalog, en donde se detalla el proceso que sigue el sistema al dar de alta un catálogo.

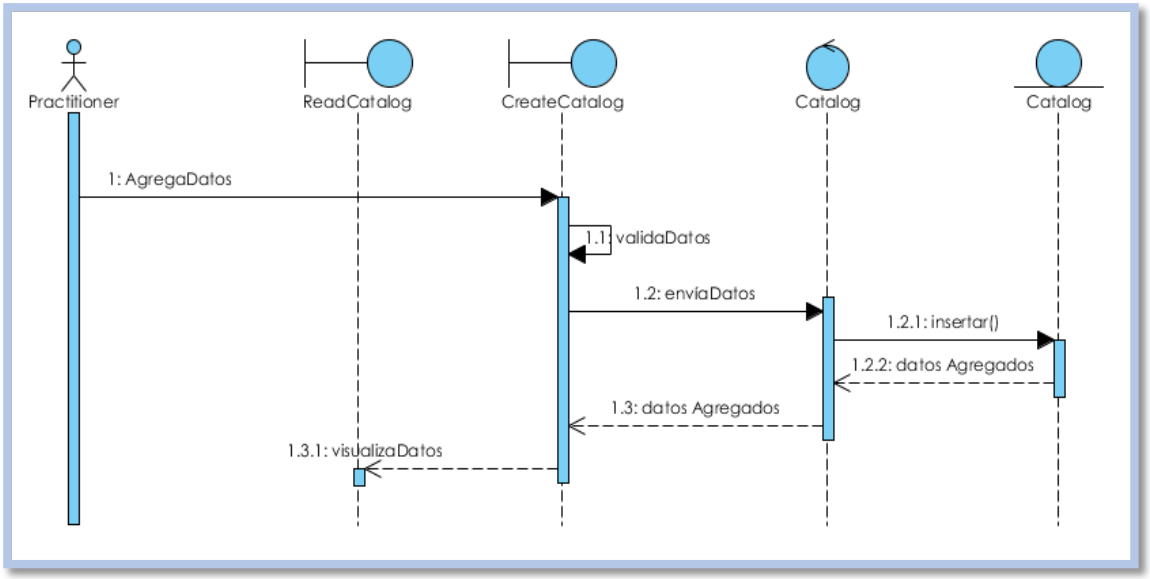


Figura 8.Diagrama de Secuencia Create Catalog

5.1.2 Read Catalog

La figura 9 muestra el diagrama de secuencia read catalog, en donde se detalla el proceso que sigue el sistema para mostrar al practicante un catálogo.

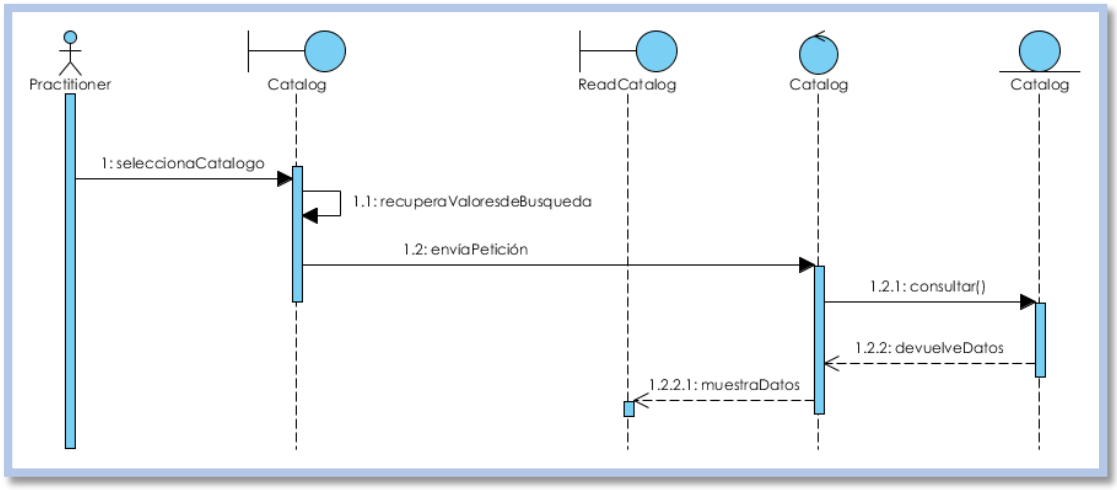


Figura 9.Diagrama de Secuencia Read Catalog

5.1.3 Update Catalog

La figura 10 muestra el diagrama de secuencia update catalog, donde se describe el proceso que sigue el sistema para permitir al practicante actualizar o modificar los datos de un catálogo.

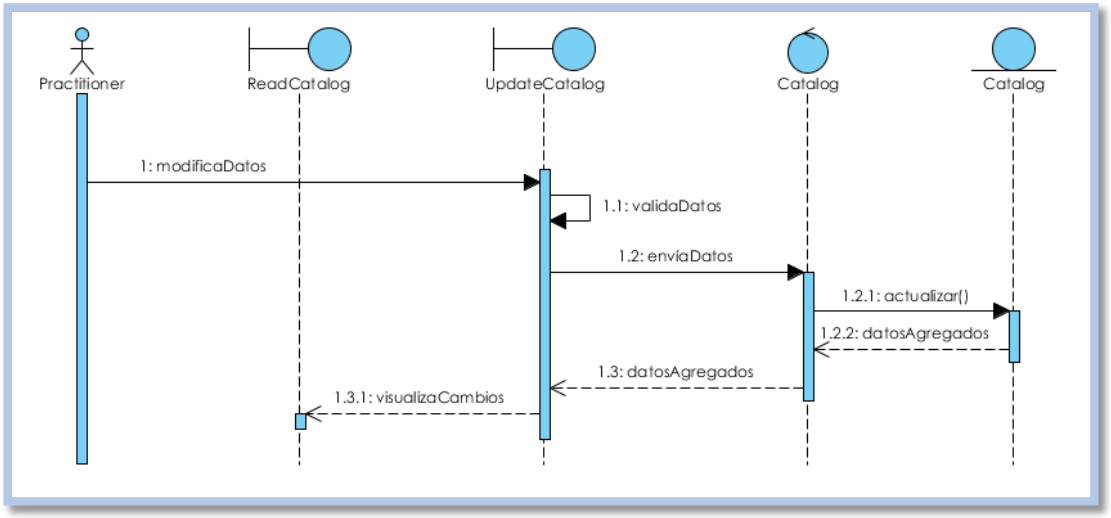


Figura 10.Diagrama de Secuencia Update Catalog

5.1.4 Delete Catalog

A continuación se presenta el diagrama de secuencia delete catalog, donde se describe el proceso que sigue el sistema para permitir al practicante eliminar los datos almacenados dentro de un catálogo. Ver figura 11.

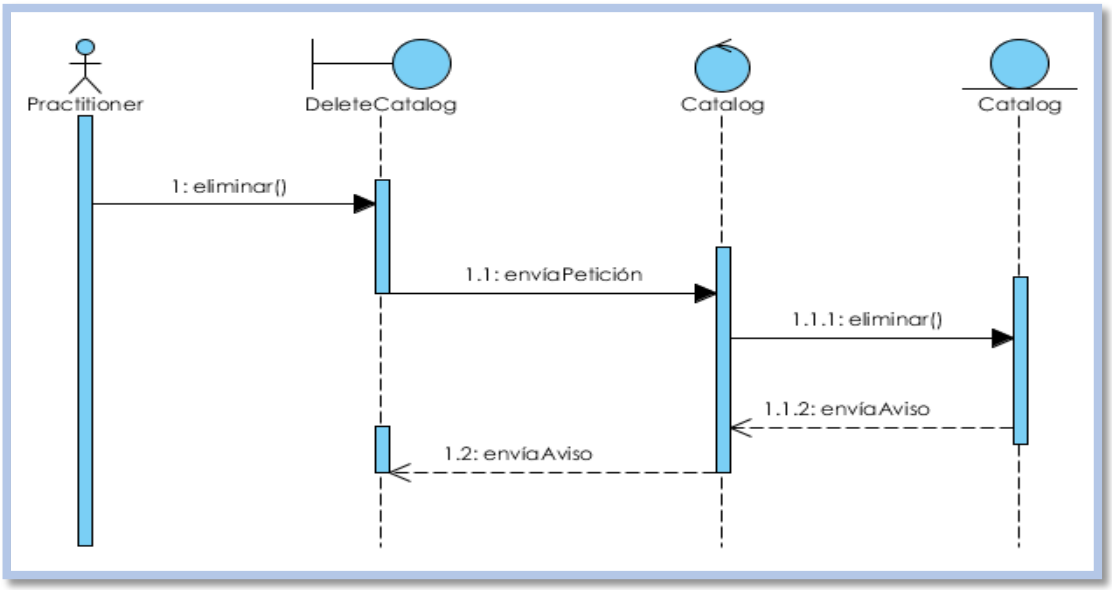


Figura 11. Diagrama de Secuencia Delete Catalog

5.1.5 Create Activity

La figura 12 muestra el diagrama de secuencia create activity, en donde se detalla el proceso que sigue el sistema para permitir al practicante dar de alta una actividad.

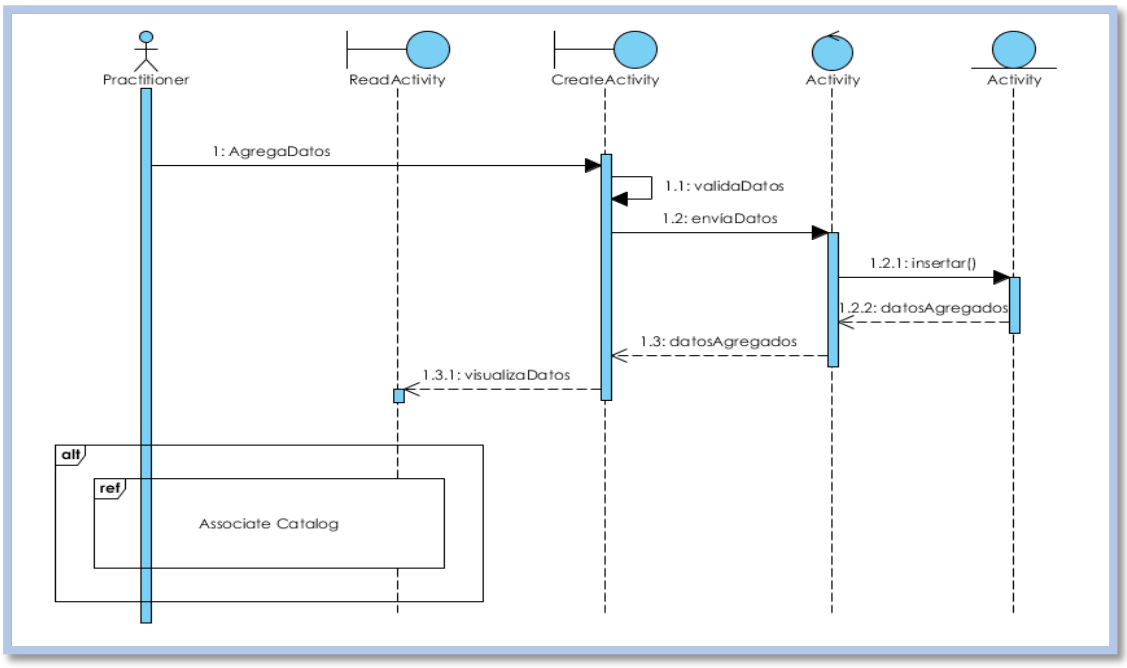


Figura 12.Diagrama de Secuencia Create Activity

5.1.6 Read Activity

La figura 13 muestra el diagrama de secuencia read activity, en donde se detalla el proceso que sigue el sistema para mostrar al practicante los datos contenidos dentro de una actividad.

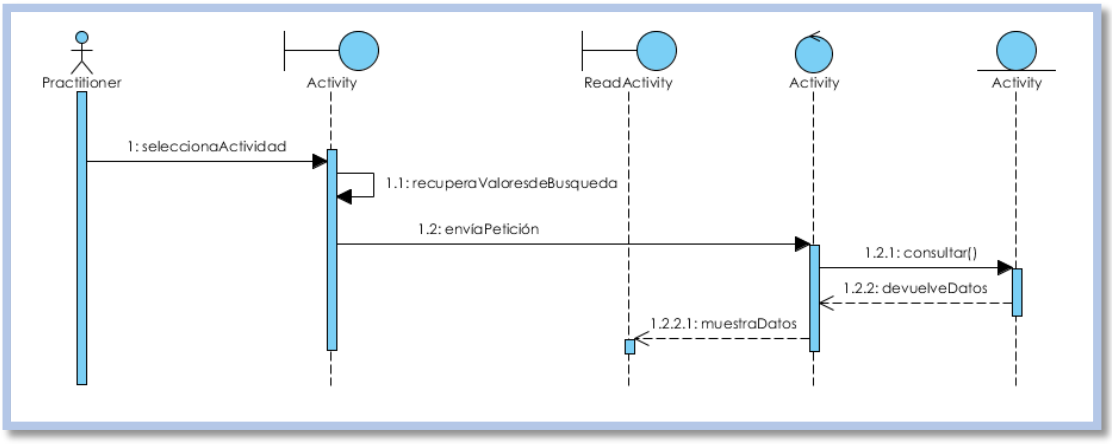


Figura 13.Diagrama de Secuencia Read Activity

5.1.7 Update Activity

La figura 14 muestra el diagrama de secuencia update activity, donde se describe el proceso que sigue el sistema para permitir al practicante actualizar o modificar los datos almacenados dentro de una actividad.

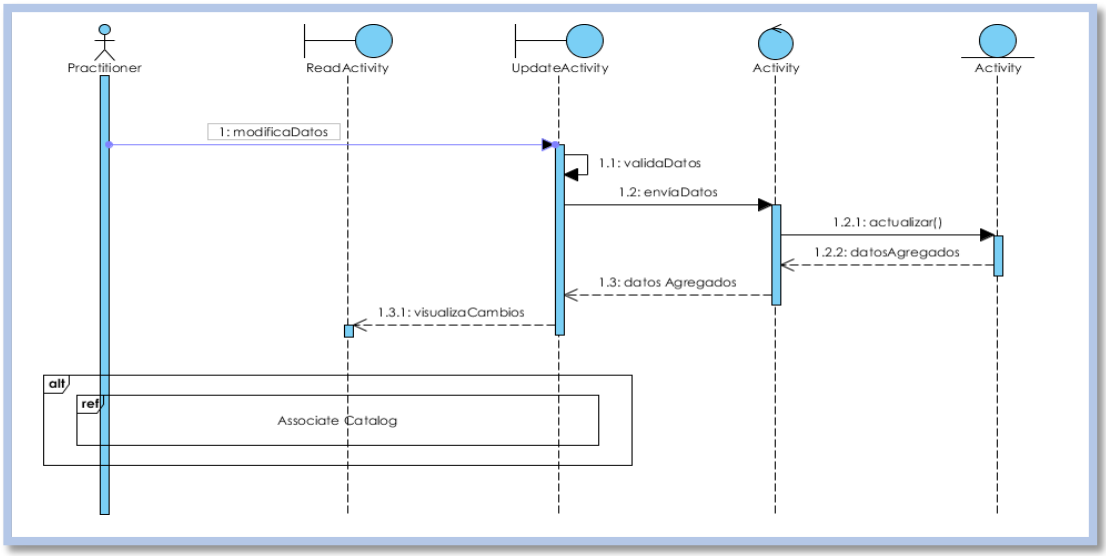


Figura 14.Diagrama de Secuencia Update Activity

5.1.8 Delete Activity

A continuación se presenta el diagrama de secuencia delete activity, en donde se describe el proceso que sigue el sistema para permitir al practicante eliminar los datos almacenados de una actividad. Ver figura 15.

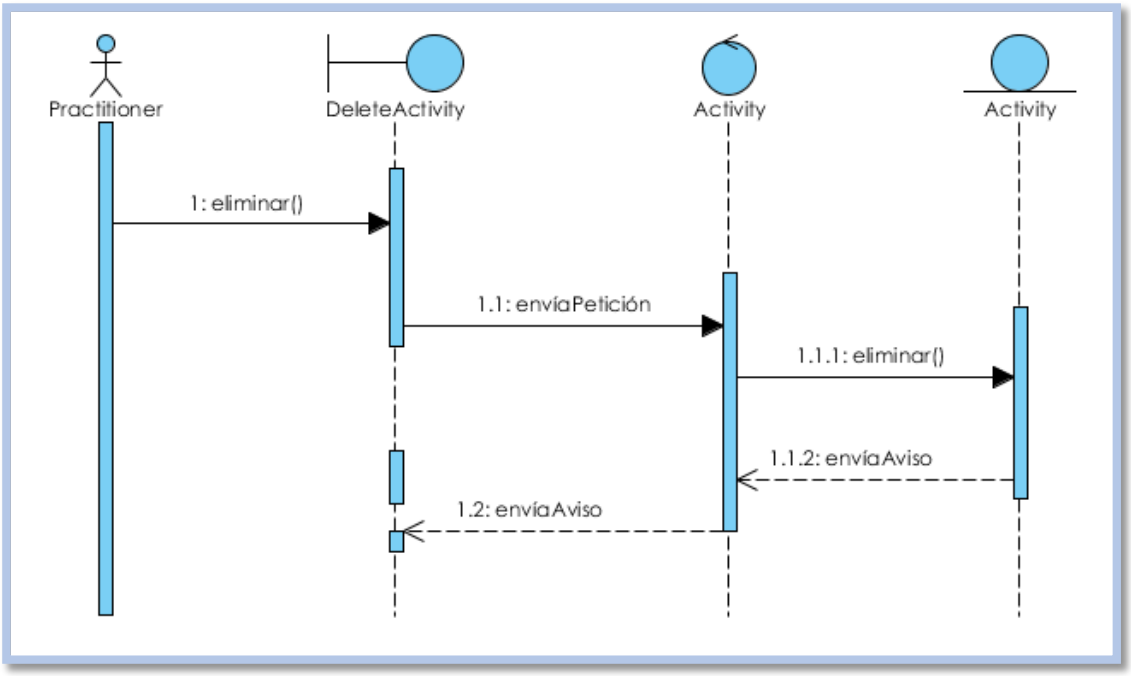


Figura 15.Diagrama de Secuencia Delete Activity

5.1.9 Associate Catalog

La figura 16 muestra un diagrama de secuencia en donde se puede observar el proceso de asociación de catálogos a las actividades correspondientes.

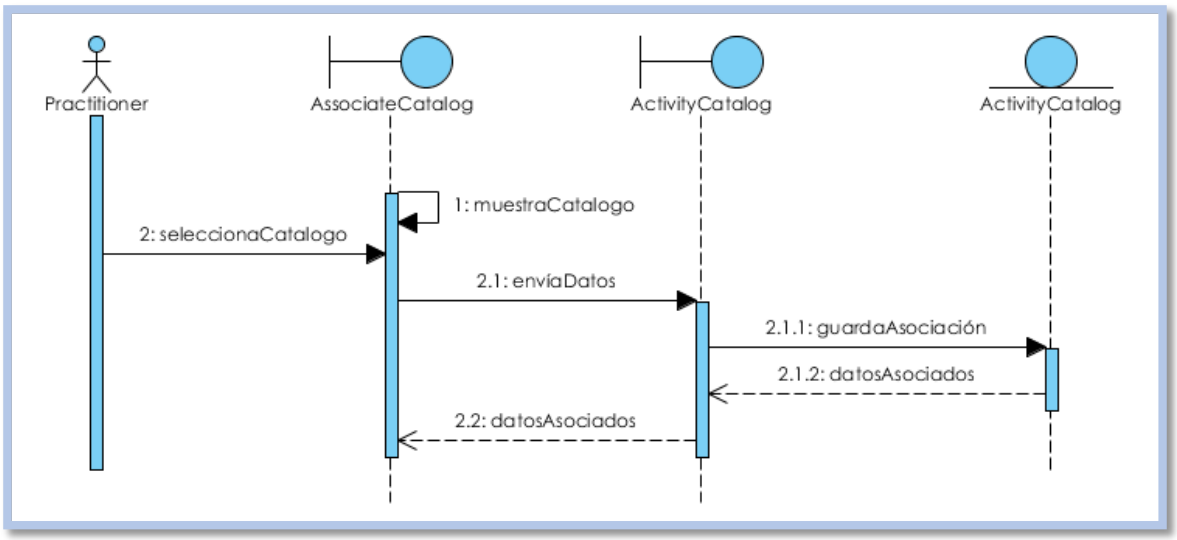


Figura 16.Diagrama de Secuencia Associate Catalog

5.1.10 Create Practice

La figura 17 muestra el diagrama de secuencia create practice, en donde se detalla el proceso que sigue el sistema para permitir al practicante crear una nueva práctica.

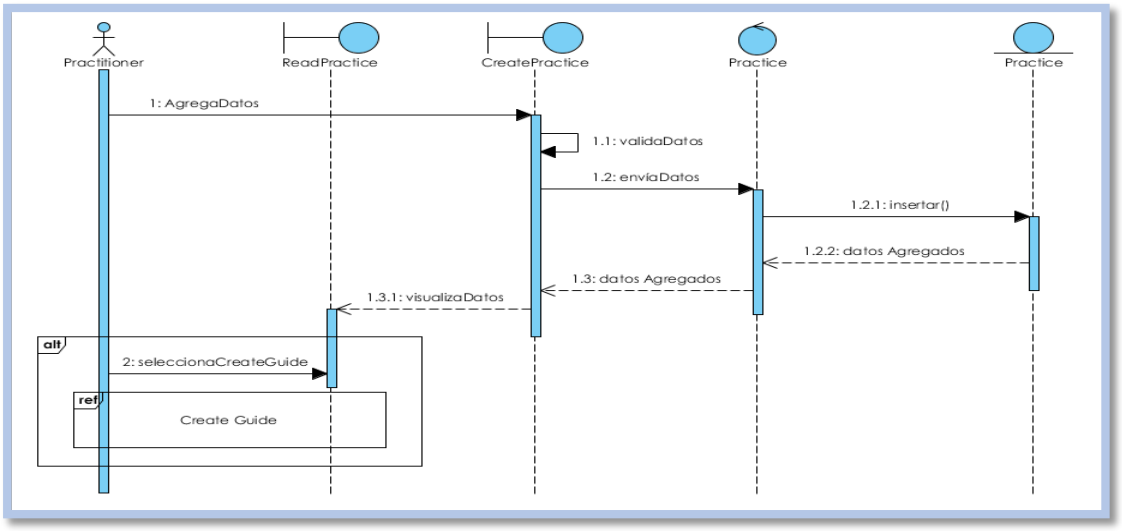


Figura 17.Diagrama de Secuencia Create Practice

5.1.11 Read Practice

La figura 18 muestra el diagrama de secuencia read practice, en el cual se presenta el proceso que sigue el sistema para mostrar al practicante los datos contenidos dentro de una práctica.

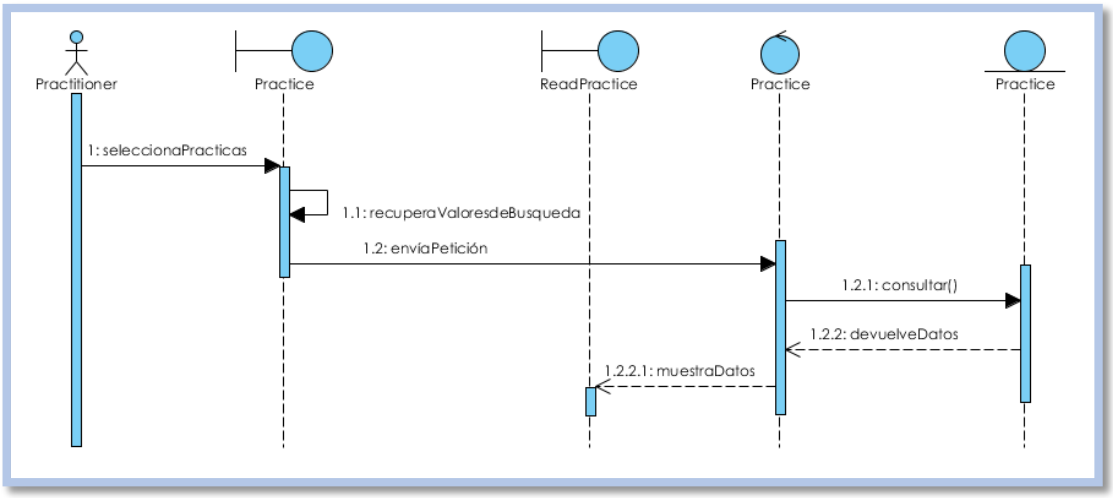


Figura 18.Diagrama de Secuencia Read Practice

5.1.12 Update Practice

La figura 19 muestra el diagrama de secuencia update practice, en el que se describe el proceso que permite al practicante actualizar o modificar los datos almacenados en una práctica dentro el sistema.

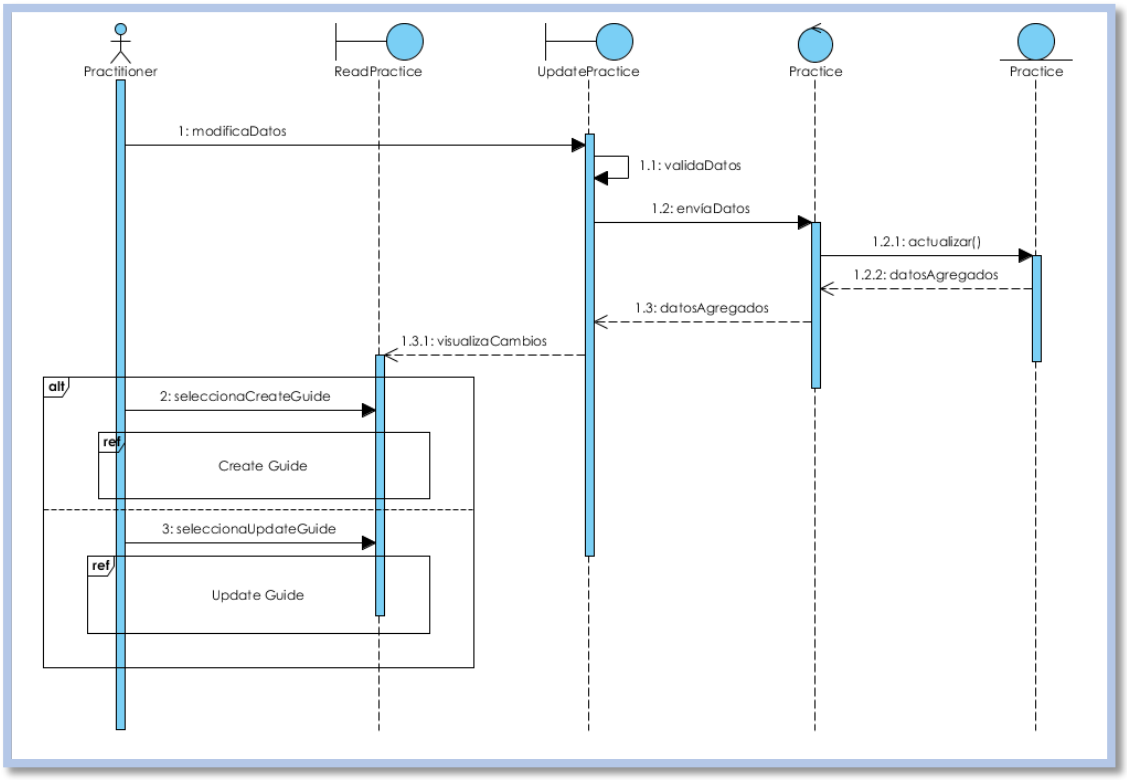


Figura 19.Diagrama de Secuencia Update Practice

5.1.13 Delete Practice

A continuación se presenta el diagrama de secuencia delete practice, en donde se describe el proceso que sigue el sistema para permitir al practicante eliminar los datos almacenados de una práctica. Ver figura 20.

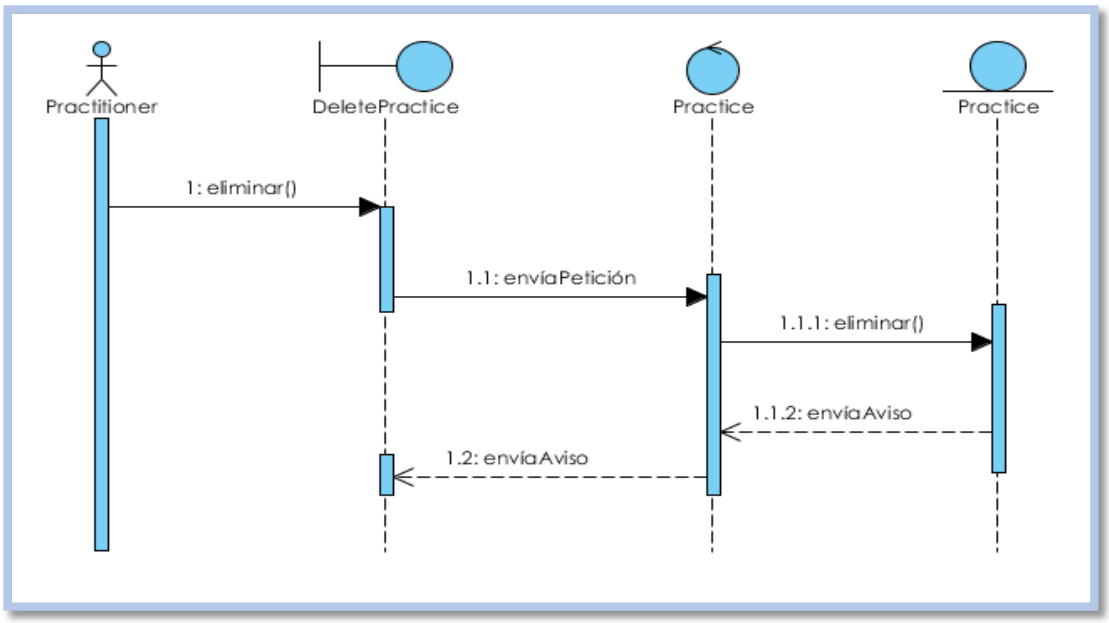


Figura 20.Diagrama de Secuencia Delete Practice

5.1.14 Create Guide

La figura 21 muestra el diagrama de secuencia create guide, en donde se detalla el proceso que sigue el sistema para permitir al practicante crear una nueva guía.

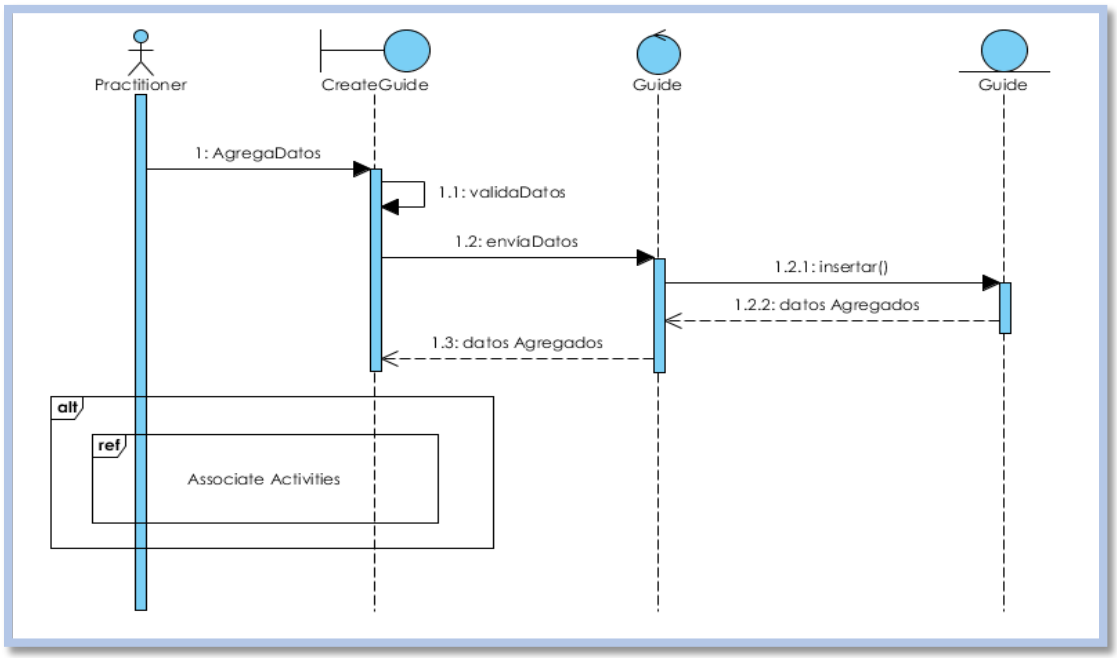


Figura 21.Diagrama de Secuencia Create Guide

5.1.15 Read Guide

La figura 22 muestra el diagrama de secuencia read practice, en el cual se presenta el proceso que sigue el sistema para mostrar al practicante los datos contenidos dentro de una guía.

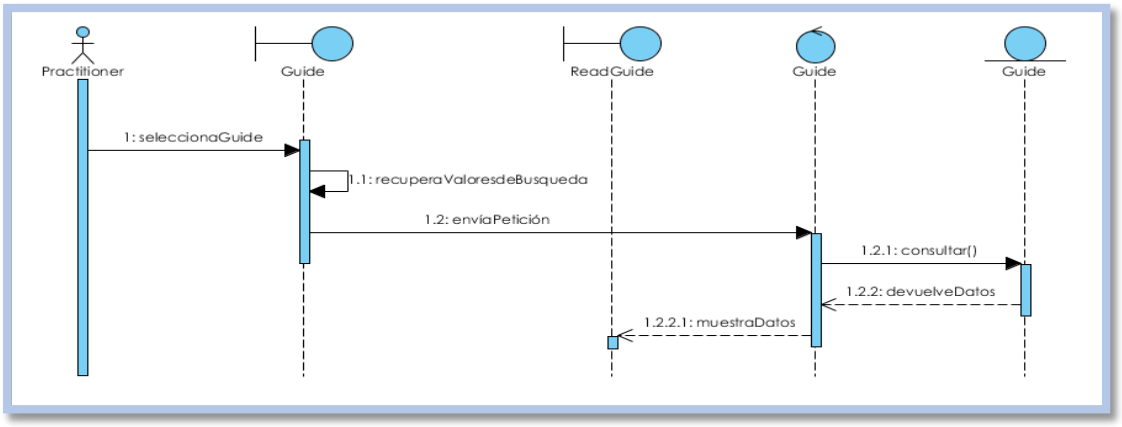


Figura 22.Diagrama de Secuencia Read Guide

5.1.16 Update Guide

La figura 23 muestra el diagrama de secuencia update guide, en el que se describe el proceso que permite al practicante actualizar o modificar los datos almacenados en una guía dentro el sistema.

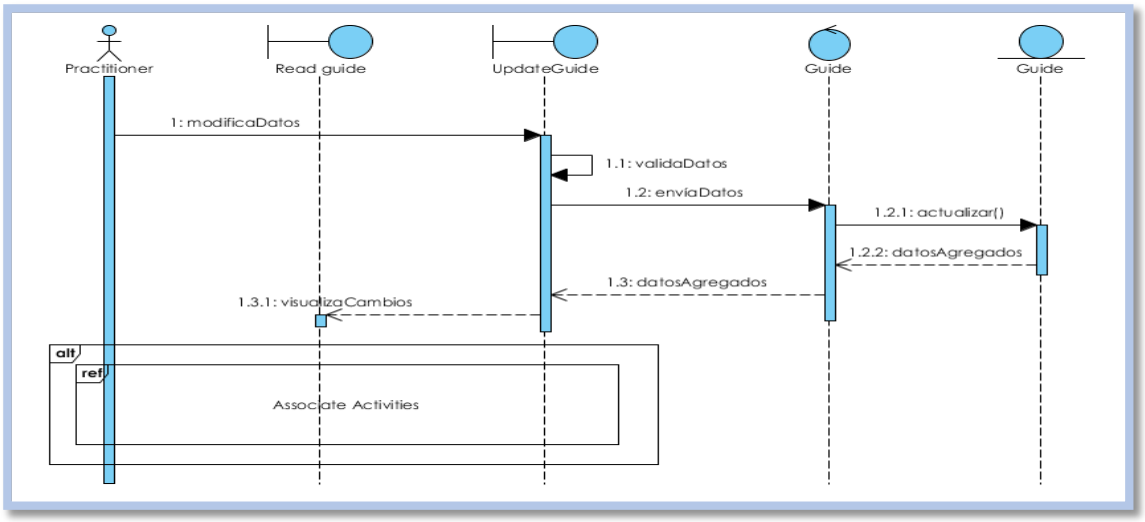


Figura 23.Diagrama de Secuencia Update Guide

5.1.17 Delete Guide

A continuación se presenta el diagrama de secuencia delete guide, en donde se describe el proceso que sigue el sistema para permitir al practicante eliminar los datos almacenados en una guía. Ver figura 24.

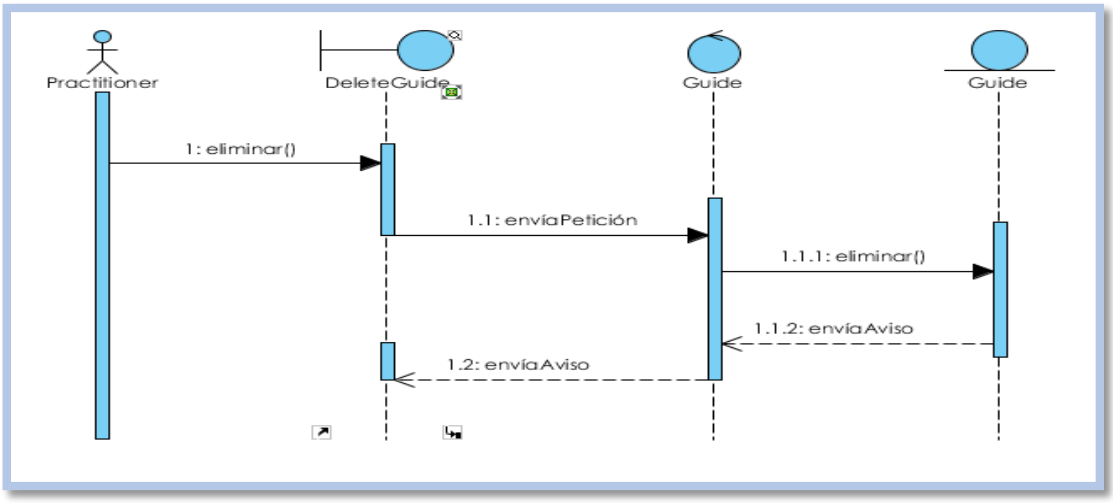


Figura 24.Diagrama de Secuencia Delete Guide

5.1.18 Associate Activities

La figura 25 muestra el proceso de asociación de las actividades con las guías correspondientes.

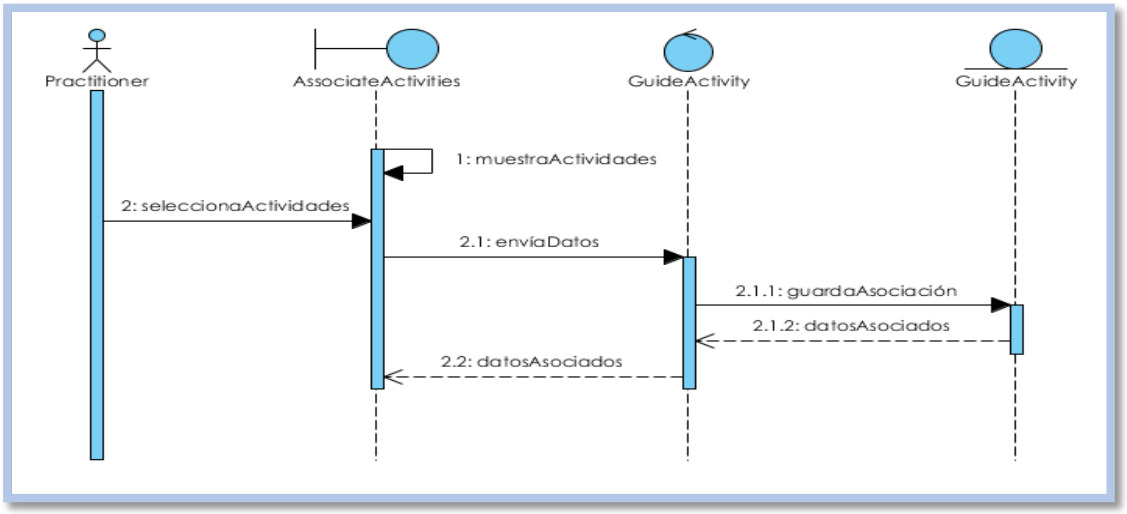


Figura 25.Diagrama de Secuencia Associate Activities

5.1.19 Create Method

La figura 26 muestra el diagrama de secuencia create method, en el cual se detalla el proceso que sigue el sistema para permitir al practicante crear un nuevo método.

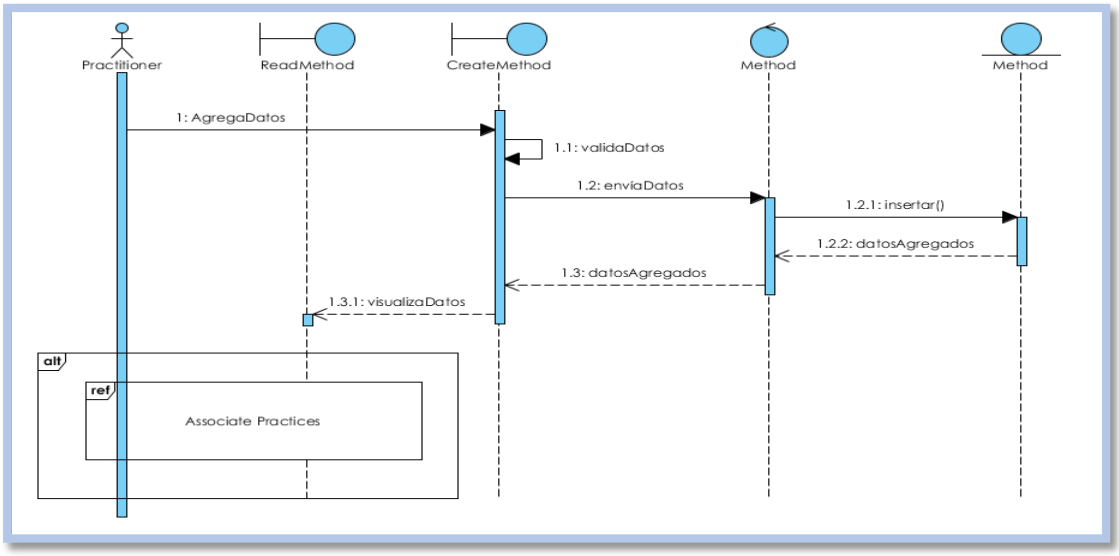


Figura 26.Diagrama de Secuencia Create Method

5.1.20 Read Method

La figura 27 muestra el diagrama de secuencia read method, en el que se detalla el proceso que sigue el sistema para mostrar al practicante los datos contenidos dentro de un método.

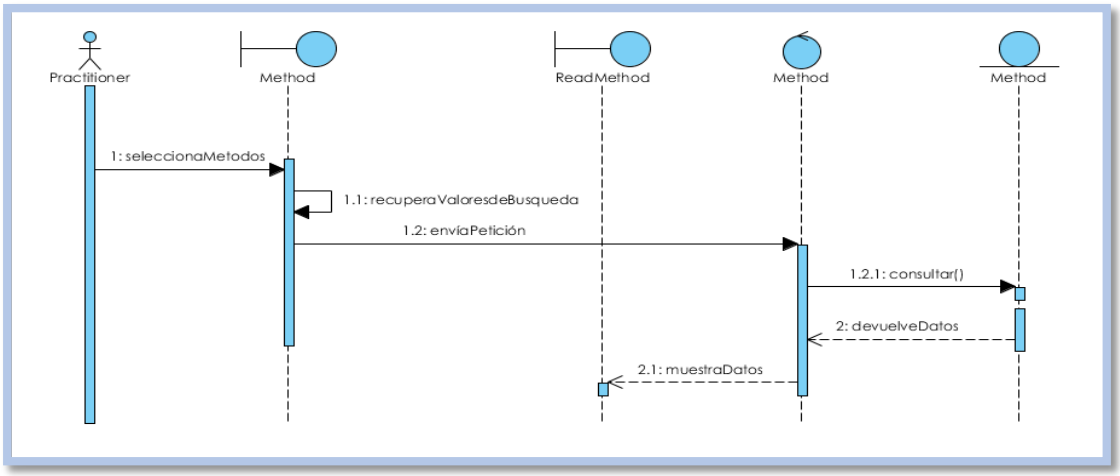


Figura 27.Diagrama de Secuencia Read Method

5.1.21 Update Method

La figura 28 muestra el diagrama de secuencia update method, en el que se describe el proceso que permite al practicante actualizar o modificar los datos almacenados en un método dentro el sistema.

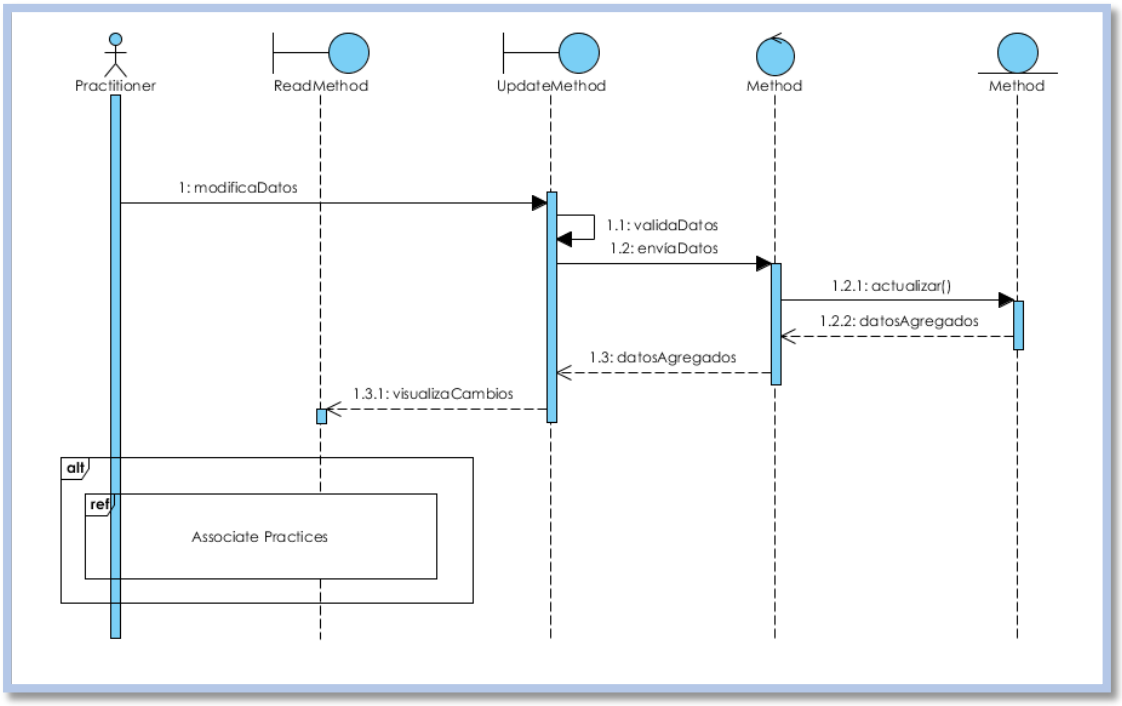


Figura 28.Diagrama de Secuencia Update Method

5.1.22 Delete Method

A continuación se presenta el diagrama de secuencia delete method, en donde se describe el proceso que sigue el sistema para permitir al practicante eliminar los datos almacenados de un método. Ver figura 29.

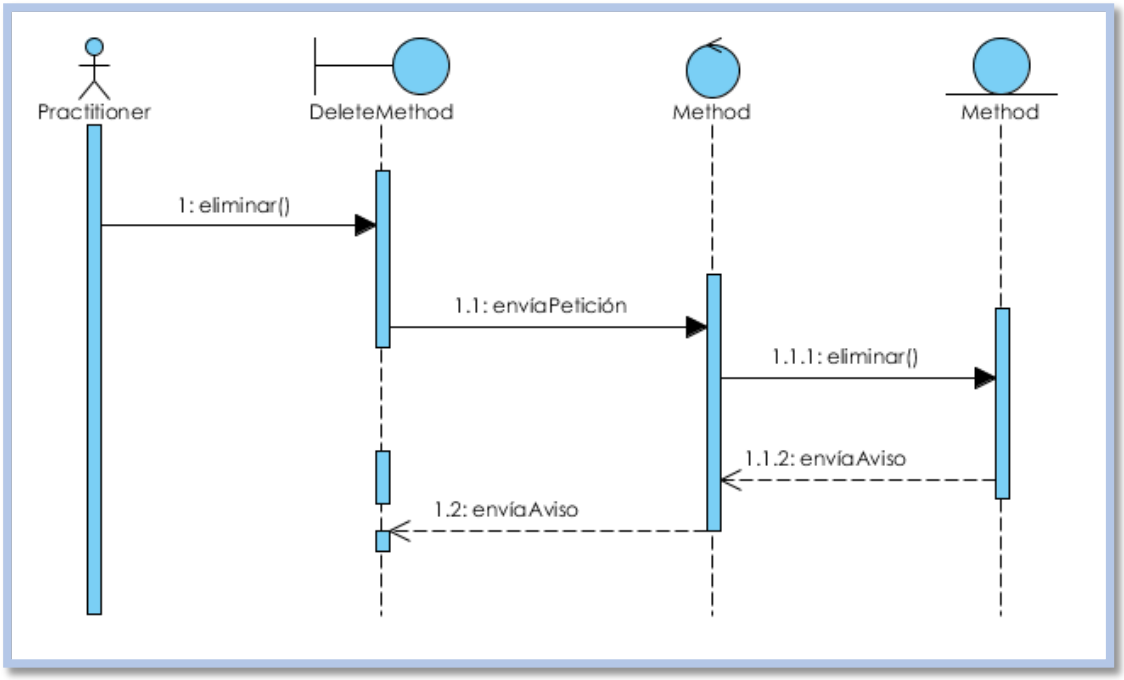


Figura 29.Diagrama se Secuencia Delete Method

5.1.23 Associate Practices

La figura 30 muestra el proceso de asociación de las prácticas con los métodos respectivos.

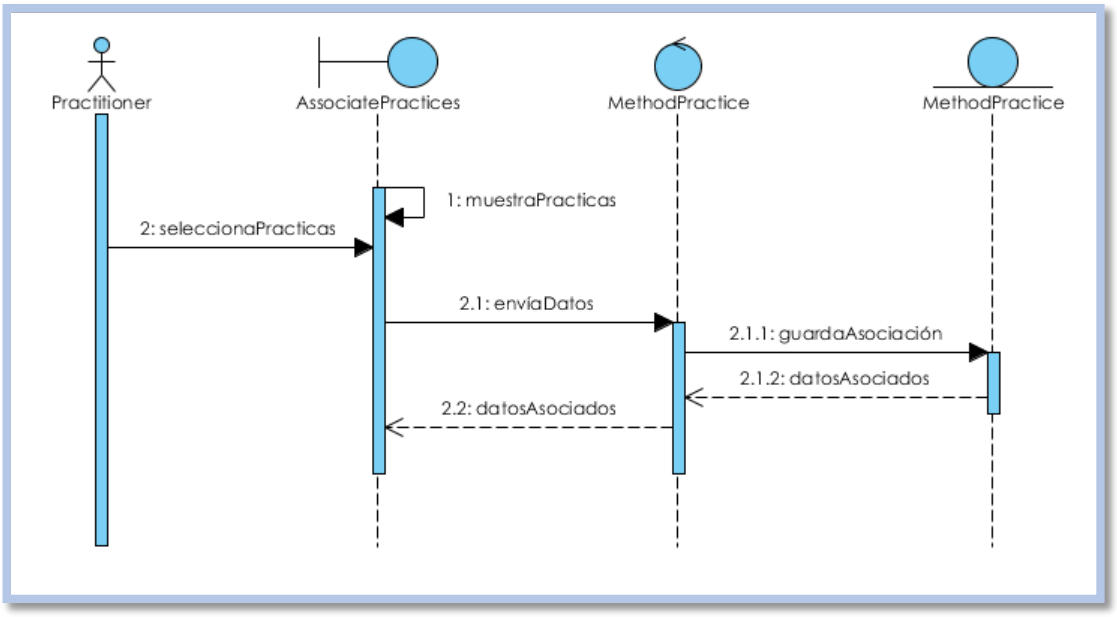


Figura 30.Diagrama de Secuencia Associate Practices

6. Diseño de la Base de Datos

Como ya hemos mencionado, el modelo relacional es el que mejor se adaptó para satisfacer las necesidades del equipo en la organización de los datos que se manejarán en la herramienta. Además de ser un modelo que representa de forma sencilla la estructura lógica del negocio. Razones por las cuales se optó por su uso.

Como sistema gestor de base de datos se decidió emplear PostgreSQL en su versión 9.2. Ésta decisión fue tomada ya que PostgreSQL logra satisfacer las necesidades para la realización de la herramienta. Además de ser un sistema gestor de base de datos es de código abierto, soporta el modelo entidad relación, utiliza un modelo cliente/servidor, y es uno de los gestores más seguros del mercado.

6.1 Diagrama Entidad Relación

El Diagrama Entidad Relación es una herramienta que utiliza un lenguaje gráfico, el cual se usa para el modelado de datos. Básicamente un diagrama entidad relación consiste en un conjunto de Entidades, las cuales representan objetos del mundo real en la mayoría de las veces, éstas entidades se relacionan unas con otras con el objetivo de resolver un problema en particular.

La figura 31 muestra el diagrama Entidad Relación que se utilizó para la creación de la base de datos. Este diagrama corresponde a la vista estática de KUALI-BEH.

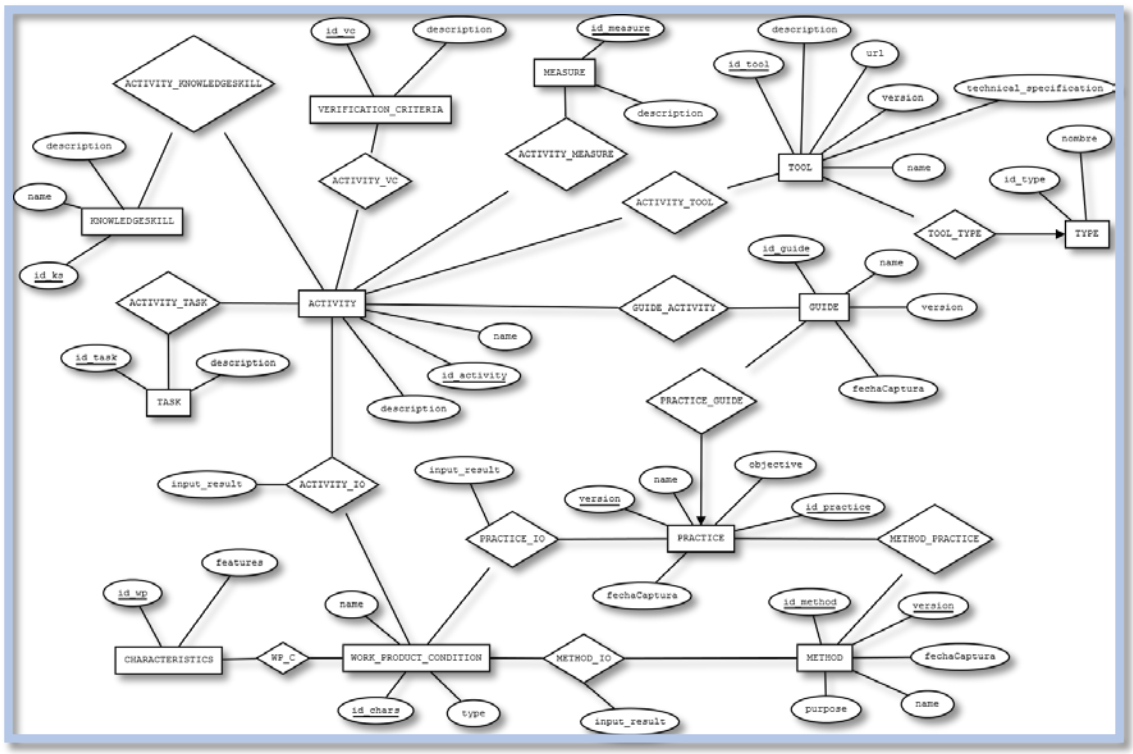


Figura 31. Diagrama Entidad Relación de la Vista Estática de KUALI-BEH

En la figura 32 se presenta el diagrama de tablas de la base de datos de la vista estática de KUALI-BEH.

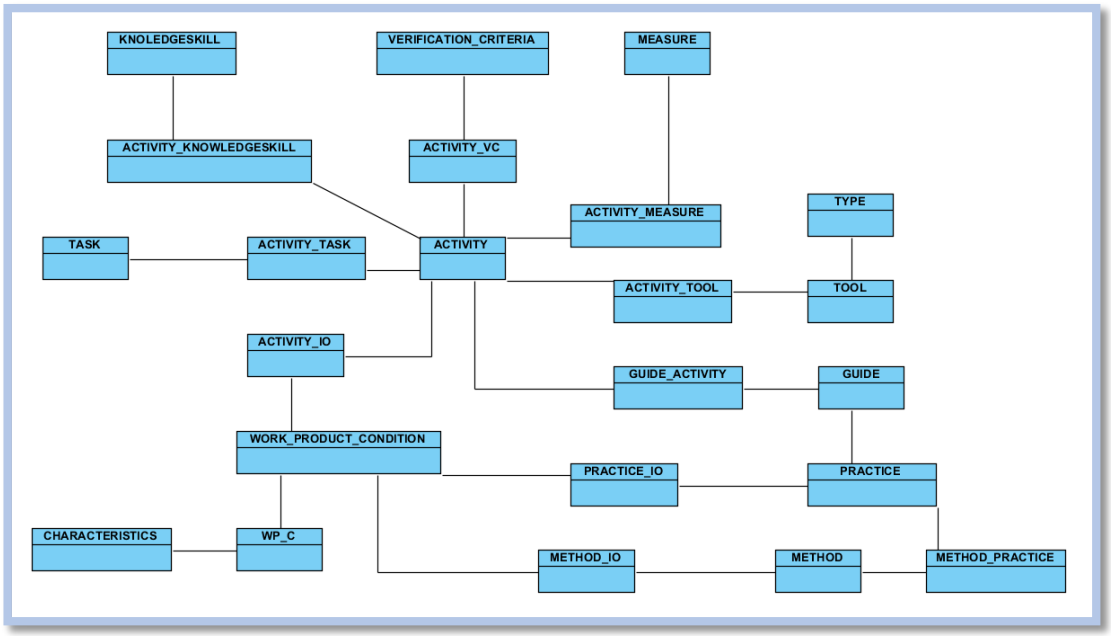


Figura 32. Diagrama de Tablas

La tabla 2 muestra una breve descripción de las entidades y atributos contenidos en el diagrama entidad relación de la vista estática.

Tabla 2. Descripción diagrama Entidad-Relación.

ENTIDAD	DESCRIPCIÓN Y ATRIBUTOS
KNOWLEDGESKILL	Almacena información de los conocimientos y habilidades que deberá tener el encargado de realizar alguna actividad específica, se asigna un nombre y un atributo a la información. **Atributos:** ▪ **id_ks.** Identificador ▪ **name.** Nombre ▪ **description.** Contiene una breve descripción de la información.
VERIFICATION_CRITERIA	Contiene los diferentes criterios que permitirán saber si una actividad se llevó a cabo correctamente. **Atributos:** ▪ **id_vc.** Identificador. ▪ **description.** Breve descripción de los criterios de verificación.
MEASURE	Almacena las métricas que se utilizarán para verificar el grado de éxito que se tuvo en la realización de las actividades. Atributos: ▪ **id_measure.** Identificador. ▪ **description.** Breve descripción de la

	métrica.
TOOL	Resguarda las herramientas necesarias para la ejecución de las actividades. Contiene los elementos necesarios para su descripción, si se tratase de algún tipo de software, incluye los atributos: versión, url y especificaciones técnicas para su especificación. **Atributos:** ▪ **id_tool.** Identificador. ▪ **name.** Nombre de la herramienta ▪ **description.** Descripción de la herramienta. ▪ **version.** Versión de la herramienta si se tratase de algún software. ▪ **technical_specification.** Especificaciones técnicas de necesarias para instalar la herramienta si se tratase de un software ▪ **url.** Dirección web donde encontrar la herramienta para su descarga o para obtener información de la misma.
TYPE	Especifica qué tipo de herramienta será la que se almacenará. **Atributos:** ▪ **id_type.** Identificador. ▪ **nombre.** Nombre.
GUIDE	Almacena la información de las guías, éstas guías contienen información que ayudarán a la realización de las prácticas. **Atributos:**

	▪ **id_guide.** Identificador. ▪ **name.** Nombre de la Guía. ▪ **version.** Se le asigna una versión a cada guía y ésta pude sufrir modificaciones en cualquier momento que se requiera. ▪ **fechaCaptura.** Este atributo es usado por el sistema para resguardar el historial por tiempo de las modificaciones hechas a las guías.
WORK_PRODUCT_ CONDITION	Almacena los productos de trabajo o condiciones que serán las entradas o resultados de las actividades, prácticas o métodos. **Atributos:** ▪ **id_chars.** Identificador. ▪ **name.** Nombre. ▪ **type.** Puede ser Producto de Trabajo o Condición.
CHARACTERISTICS	Contiene una descripción de las características de los productos de trabajo o condiciones. **Atributos:** ▪ **id_wp**. Identificador. ▪ **features.** Características.
ACTIVITY	Almacena un conjunto de pasos o procedimientos a seguir. Las características tienen asociadas los conocimientos y habilidades, criterios de verificación, métricas, herramientas, productos de trabajo y condiciones necesarios para su realización. **Atributos:** ▪ **id_activity.** Identificador.

	▪ **name.** Nombre. ▪ **description.** Descripción.
PRACTICE	Contiene la información de la práctica. Tiene asociada las guías necesarias para su ejecución. **Atributos:** ▪ **id_practice**. Identificador. ▪ **name.** Nombre de la práctica. ▪ **version.** Cada práctica contiene una versión, la cual puede ser modificada en el momento que sea necesario. ▪ **objective.** Contiene una breve descripción de la práctica. ▪ **fechaCaptura.** Es usado por el sistema para guardar un historial por tiempo de las diferentes prácticas almacenadas.
METHOD	Almacena la información de los métodos. Tiene asociadas las prácticas necesarias para su ejecución. **Atributos:** ▪ **id_method.** Identificador del método. ▪ **name.** Nombre del método. ▪ **versión.** Versión del método. ▪ **purpose.** Propósito u objetivo del método. ▪ **fechaCaptura.** Fecha usada por el sistema para guardar un historial del método utilizando la fecha de su creación o modificación.

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

APÉNDICE C

PRUEBAS

En este apartado se incluye la documentación de las pruebas de funcionalidad y usabilidad aplicadas al Repositorio de Métodos y Prácticas de Proyectos de Software para KUALI-BEH, los resultados obtenidos de las mismas y las correcciones realizadas conforme a los resultados arrojados.

ÍNDICE

PRUEBAS

1. Introducción

En este documento se incluyen los planes de pruebas aplicados al Repositorio de Métodos y Prácticas de Proyectos de Software para KUALI-BEH, así como también los resultados obtenidos de las mismas y las correcciones realizadas conforme a los resultados arrojados.

1.1 Propósito

Se realizaron pruebas de funcionalidad y usabilidad con el propósito de verificar y validar que se hayan cumplido los requerimientos tanto funcionales como no funcionales, señalar los defectos y aciertos que contenía el sistema, corregirlos y dar como resultado un sistema eficaz, eficiente, seguro y de utilidad para el usuario.

1.2 Alcance

Se realizaron pruebas de funcionalidad con la finalidad de verificar que se cumplieran los requisitos funcionales y no funcionales del sistema. Para ello se

aplicaron pruebas a cada uno de los casos de uso definidos y, con base en los resultados, se analizó que éstos se cumplieran conforme a lo establecido.
También se aplicaron pruebas de usabilidad, en las que se contó con la participación de practicantes de Ingeniería de Software, los cuales usaron datos reales de proyectos en los que se encontraban trabajando en ese momento. Los resultados obtenidos permitieron realizar la validación del sistema y comprobar que tan atractivo es para el usuario en cuanto al diseño de la interfaz y facilidad de uso.

2. Descripción general

En éste Apéndice se documenta las pruebas que se realizaron a la herramienta Repositorio de Métodos y Prácticas de Proyectos de Software para KUALI-BEH.

3. Requerimientos para las pruebas

Los requerimientos de las pruebas para los practicantes y demás personas que la realizaron fueron: Tener una computadora con por lo menos un browser (Mozilla o Chrome) y conexión a internet.
A los usuarios que realizaron las pruebas de usabilidad se les proporcionó un documento describiendo los pasos a seguir para la realización de las pruebas, así como los datos de prueba que se utilizarían para cada funcionalidad. Este documento también contenía cuestionarios para recabar las opiniones y resultados de las pruebas. Dicho documento se presenta en el anexo A de éste trabajo.

4. Reporte de Pruebas del Sistema

A continuación se anexa un resumen de las pruebas que se llevaron a cabo.

Las tablas 1, 2, 3, 4 y 5 (CRUD Catalog, CRUD Activity, CRUD Practice, CRUD Guide y CRUD Method respectivamente) contienen una descripción de las pruebas de funcionalidad realizadas, los resultados esperados y los resultados obtenidos por cada caso de uso.
Más adelante se presentan las pruebas de usabilidad que se aplicaron al sistema.

4.1 Casos de Prueba

Durante el desarrollo del sistema se aplicaron pruebas unitarias por cada caso de uso para comprobar que realmente se cumpliera con lo esperado, cabe señalar que el manejo de Frameworks agilizó que los resultados de las pruebas fueran los esperados y que la integración de los elementos se facilitara. Una vez integrado el sistema se procedió a realizar las pruebas de integración, las tablas 1, 2, 3, 4 y 5 muestran los resultados obtenidos al término de las últimas pruebas realizadas.

La tabla 1 contiene un resumen de las pruebas de usabilidad aplicadas al caso de uso CRUD Catalog, en donde se verifica que se obtengan los resultados esperados.

Tabla 1. CRUD Catalog

Caso de Uso	Descripción	Resultados Esperados	Resultados Obtenidos
Create Catalog	Los practicantes podrán dar de alta información en algún catálogo.	El sistema almacenará diferentes catálogos dentro del sistema.	Al comienzo del desarrollo del proyecto se obtuvieron problemas, principalmente con la conexión a la base de datos, debido a no tener experiencia en el manejo de los

			Frameworks, sin embargo una vez resueltas estas problemáticas se procedió a la reutilización de código y se agilizó el desarrollo.
Read Catalog	Los practicantes podrán consultar la información de algún catálogo.	El sistema deberá mostrar toda la información del catálogo seleccionado al practicante, de forma completa y correcta.	Se obtuvieron los resultados esperados.
Update Catalog	Los practicantes podrán actualizar la información de algún catálogo.	El sistema deberá permitir al practicante actualizar la información deseada del catálogo seleccionado.	Se obtuvieron los resultados esperados.
Delete Catalog	Los practicantes podrán eliminar la información correspondiente a algún catálogo.	El sistema deberá eliminar correctamente de la base de datos toda la información del catálogo que el practicante desee.	Se obtuvieron los resultados esperados.

La tabla 2 muestra los resultados esperados y los resultados obtenidos en las pruebas de funcionalidad del caso de uso CRUD Activity.

Tabla 2. CRUD Activity.

Caso de Uso	Descripción	Resultados Esperados	Resultados Obtenidos
Create Activity / Associate Catalog	El practicante podrá almacenar una nueva actividad dentro del sistema y al mismo tiempo si lo desea podrá asociar a la misma los catálogos correspondientes.	El sistema almacenará en la base de datos la información de la actividad que el practicante desee.	Se presentaron algunos problemas con las asociaciones de los catálogos, principalmente en la interfaz debido a la falta de experiencia en el manejo de Framewoks, una vez que se resolvieron éstos problemas se aplicó la reutilización de código y se replicó en las posteriores asociaciones que presenta el sistema.
Read Activity	Los practicantes podrán consultar la información de alguna actividad.	El sistema deberá mostrar toda la información de la actividad seleccionada al practicante, de forma completa y correcta, incluyendo los catálogos asociados a la misma	Se obtuvieron los resultados esperados.

Update Activity / Associate Catalog	Los practicantes podrán actualizar la información de alguna actividad.	El sistema deberá permitir al practicante actualizar la información deseada de la actividad seleccionada, así como también actualizar los catálogos asociados a ella.	Se obtuvieron los resultados esperados.
Delete Activity	Los practicantes podrán eliminar la información correspondiente a una actividad.	El sistema deberá eliminar correctamente de la base de datos la información de toda la actividad que el practicante desee, incluyendo sus asociaciones.	Se obtuvieron los resultados esperados.

Los resultados de las pruebas para el caso de uso CRUD Practice se presentan en la tabla 3. En estas pruebas se espera que el sistema realice las acciones de crear, leer, modificar y eliminar una práctica.

Tabla 3. CRUD Practice.

Caso de Uso	Descripción	Resultados Esperados	Resultados Obtenidos
Create Practice	Los practicantes podrán dar de alta información de las	El sistema almacenará los datos de la práctica	Se obtuvieron los resultados esperados.

	prácticas que requieran.	correctamente en la base de datos	
Read Practice	Los practicantes podrán consultar la información de la práctica que deseen	El sistema deberá mostrar toda la información de la práctica seleccionada al practicante, de forma completa y correcta.	Se obtuvieron los resultados esperados.
Update Practice	Los practicantes podrán actualizar la información de las prácticas.	El sistema deberá permitir al practicante actualizar la información deseada de la práctica seleccionada.	Se presentaron conflictos debido al manejo de versiones y sus respectivas asociaciones. Se realizaron varias pruebas hasta lograr obtener los resultados esperados.
Delete Practice	Los practicantes podrán eliminar toda la información correspondiente a alguna práctica.	El sistema deberá eliminar correctamente de la base de datos toda la información de la práctica que el practicante desee.	Se obtuvieron los resultados esperados.

La tabla 4 presenta las pruebas realizadas para el caso de uso CRUD Guide. Se muestran los resultados esperados y los resultados obtenidos.

Tabla 4. CRUD Guide.

Caso de Uso	Descripción	Resultados Esperados	Resultados Obtenidos
Create Guide / Associate Activities	El practicante podrá almacenar una nueva guía dentro del sistema y al mismo tiempo si lo desea podrá asociar a la misma las actividades correspondientes.	El sistema almacenará en la base de datos la información de la guía que el practicante desee.	Se aplicaron varias pruebas debido a que se obtenían errores en la interfaz correspondiente, se probaron diversos componentes y temas del Framework utilizado hasta obtener los resultados esperados.
Read Guide	Los practicantes podrán consultar la información de alguna guía.	El sistema deberá mostrar toda la información de la guía seleccionada al practicante, de forma completa y correcta, incluyendo las actividades asociadas a la misma	Se obtuvieron los resultados esperados.
Update Guide / Associate Activities	Los practicantes podrán actualizar la información de alguna guía.	El sistema deberá permitir al practicante actualizar la información deseada de la guía seleccionada, así	Se observaron diferentes problemas en la interfaz, causadas por la manipulación de información de varias guías al

		como también actualizar las actividades asociadas a ella.	mismo tiempo. Se aplicaron diversas pruebas y se corrigieron los errores encontrados hasta que se obtuvieron los resultados esperados.
Delete Guide	Los practicantes podrán eliminar la información correspondiente a una actividad.	El sistema deberá eliminar correctamente de la base de datos la información de toda la guía que el practicante desee, incluyendo sus asociaciones.	Se obtuvieron los resultados esperados.

En la tabla 5 se muestran los resultados obtenidos de las pruebas realizadas al caso de uso CRUD Method.

Tabla 5. CRUD Method.

Caso de Uso	Descripción	Resultados Esperados	Resultados Obtenidos
Create Method / Associate Practices	El practicante podrá almacenar un nuevo método dentro del sistema y al mismo tiempo si lo desea	El sistema almacenará en la base de datos la información del método que el	Se obtuvieron los resultados esperados.

	podrá asociar a éste las prácticas correspondientes.	practicante desee así como las asociaciones con las prácticas correspondientes.	
Read Method	Los practicantes podrán consultar la información de algún método.	El sistema deberá mostrar toda la información del método seleccionado al practicante, de forma completa y correcta, incluyendo las prácticas asociadas a la misma	Se obtuvieron los resultados esperados.
Update Method / Associate Practices	Los practicantes podrán actualizar la información de los métodos.	El sistema deberá permitir al practicante actualizar la información deseada del método seleccionado, así como también actualizar las prácticas asociadas a éste.	Se obtuvieron los resultados esperados.
Delete Guide	Los practicantes podrán eliminar toda la información correspondiente a	El sistema deberá eliminar correctamente de la base de datos la	Se obtuvieron los resultados esperados.

	un método.	información de todo el método que el practicante desee, incluyendo sus asociaciones.	

Después de haber aplicado las pruebas de funcionalidad y verificar que se obtuvieran los resultados esperados, se procedió a aplicar las pruebas de usabilidad.

4.2 Pruebas de Usabilidad

Se realizaron pruebas de usabilidad con el propósito de señalar los aciertos y defectos del sistema. Los resultados permitieron medir la aceptación del sistema por parte de los usuarios, de la misma manera ayudaron a identificar mejoras a la interfaz y a las funcionalidades de la herramienta.

Estas pruebas, en una primera instancia, fueron realizadas por el mismo equipo de trabajo y por usuarios ajenos a la Ingeniería de Software. Los datos utilizados fueron con base a proyectos reales que se habían realizado con anterioridad. Los resultados obtenidos permitieron identificar defectos y mejoras, ocasionando modificaciones y en consecuencia se liberó una nueva versión del sistema aún más completa que la anterior.

Estas mismas pruebas se realizaron nuevamente, esta vez con usuarios reales, practicantes activos de la Ingeniería de Software. Los datos que se utilizaron fueron de proyectos reales de desarrollo de software sobre los que se encontraban trabajando en ese momento.

A continuación se mencionan de manera general los puntos que se tomaron en cuenta para la realización de éstas pruebas.

- Manejo intuitivo de la interfaz, incluso sin la necesidad de capacitación previa.
- Interfaz de usuario intuitiva a la vista de los practicantes.
- Fácil aprendizaje en el uso de la herramienta.
- Distribución correcta y adecuada de los elementos de la vista estática en el sistema.
- Utilidad en la realización de proyectos reales de desarrollo de software.
- Correcto almacenaje y resguardo de los conocimientos de los practicantes.
- Permitir el reúso de los conocimientos y conceptos almacenados en el sistema.
- Herramienta de fácil acceso por parte de los practicantes.

La tabla 6 presenta de manera general los puntos sobresalientes de las pruebas realizadas.

Tabla 6. Opiniones Generales.

Complejidad en el Sistema	Uno de los principales objetivos planteados para el sistema, es que éste no fuera complejo de usar, que lo usuarios lograrán entender la funcionalidad de los botones, ventanas, recuadros y de todos los elementos de la interfaz lo más rápido posible y de forma intuitiva. En las primeras pruebas realizadas, el sistema mostró complejidad y una interfaz poco usable. Los usuarios mostraron confusión en la realización de los ejercicios y no lograron identificar con claridad la funcionalidad de los componentes. Con base a los resultados obtenidos se hicieron mejoras a

	la interfaz, se modificaron los colores, ubicación, tamaño y textura de botones. Se modificó la composición de las ventanas y se realizó una interfaz más estética y sencilla, además de otros aspectos más que causaban conflicto a los usuarios. Cómo resultado se obtuvo una nueva versión del sistema. Los resultados de las pruebas más recientes, en la que participaron practicantes de ingeniería de software reales, las pruebas arrojaron resultados favorables. Algunos de ellos tuvieron algunos conflictos al inicio de los ejercicios, sin embargo conforme avanzaban en las pruebas, las fueron realizando de manera fluida. Los usuarios lograron identificar la mayor parte de los componentes de la interfaz y comprender su funcionamiento de manera intuitiva.
Funciones del sistema integradas correctamente.	Las diversas funciones del sistema deben estar correctamente integradas, es decir, todo el funcionamiento del sistema tiene que tener sentido y lógica. Esto era parte del objetivo a realizar en el sistema. Las primeras pruebas arrojaron resultados favorables en general. Se notaron componentes faltantes en ciertas partes de la interfaz y que en cierto momento se requerirían o que podrían ayudar a trabajar con mayor fluidez dentro del sistema. Sin embargo los componentes que ya estaban establecidos trabajaron correctamente y eran de utilidad. En la nueva versión del sistema y con los cambios ya realizados, los practicantes encontraron las funciones del sistema correctamente integradas.

Necesidad de un técnico para usar el sistema.	Para las distintas pruebas de usabilidad se utilizaron diferentes tipos de practicantes, desde personas que no tenían ningún conocimiento en KUALI-BEH, hasta practicantes reales de Ingeniería de Software con conocimientos básicos de KUALI-BEH. En ambos casos los practicantes coincidieron en no necesitar la ayuda de algún técnico para la realización de las pruebas. El sistema les resultó lo suficientemente intuitivo para su uso. Los usuarios que no tenían conocimiento alguno de KUALI-BEH únicamente necesitaron de una breve explicación de los ejercicios a realizar y de la finalidad de los mismos. Se observó que en general algunos tuvieron pequeños conflictos en un inicio por la poca experiencia en el proyecto, sin embargo durante el avance de las pruebas se fueron adaptando. El sistema es intuitivo de usar, sin embargo se recomienda dar a los usuarios una breve introducción de lo que es KUALI-BEH, así como de su objetivo, esto con la finalidad de mejorar, facilitar y comprender el uso de sistema.
Seguridad al usar el sistema.	En las primeras pruebas realizadas, los usuarios se notaron un tanto temerosos y desconfiados al usar el sistema, sin embargo, esto se debía a los errores que ya hemos mencionado anteriormente. Al realizar las modificaciones pertinentes, en general, todos los practicantes al término de las pruebas se notaron seguros al usar el sistema. Mencionaron que notaban cierto

	temor en un inicio por ser la primera vez en que lo manipulaban, sin embargo, con el paso de los ejercicios, retomaban la confianza hasta llegar a trabajar con fluidez.

Debido a los resultados de las diferentes pruebas realizadas, se aplicaron mejoras al sistema hasta obtener la versión final. Aunque en un inicio no se obtuvieron resultados favorables, los resultados de las pruebas finales fueron positivos.

El sistema logró ser intuitivo al usar, sin embargo se recomienda dar a los usuarios una breve introducción de lo que es KUALI-BEH antes de su uso, esto con la finalidad de mejorar, facilitar y comprender el uso de sistema.

ÍNDICE DE TABLAS