UNIVERSIDAD NACIONAL DE SAN CRISTÓBAL DE HUAMANGA

FACULTAD DE INGENIERÍA DE MINAS, GEOLOGÍA Y CIVIL

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

"ADAPTABILIDAD DE LA INFRAESTRUCTURA JAVA PLATFORM ENTERPRISE EDITION A LA ARQUITECTURA TÉCNICA DE LA METODOLOGÍA ÁGIL Y FORMAL ICONIX 2019. CASO DE ESTUDIO: REGISTRO DE INCIDENCIA DE BIENES INFORMÁTICOS DE LA UGEL HUAMANGA, 2019"

Tesis presentada por	:	Bach. Thalia Betty Atauje Pumallihua
Para optar el título profesional de	:	Ingeniero de Sistemas
Tipo de investigación	:	Descriptiva
Área de investigación	:	Ingeniería de Software
Asesor	:	Ing. Hubner Janampa Patilla

Ayacucho - Perú

2020

DEDICATORIA

A la vida por concederme entendimiento y sapiencia.

A mis padres, Juan y Victoria, quienes han sido ejemplo de perseverancia y superación.

A mis hermanos, quienes han sido el mentor, para poder alcanzar este punto de mi carrera profesional.

A mi compañero de la vida, Andy, fuente de esperanza, vida, paz, y amor.

,

AGRADECIMIENTO

En primer lugar al Mg. Ing. Hubner Janampa Patilla, por su asesoramiento, orientación y calidad profesional en la realización de esta tesis.

A la Universidad Nacional de San Cristóbal de Huamanga y a la plana docente de la Escuela Profesional de Ingeniería de Sistemas, por la contribución de sus conocimientos y saberes en mi profesionalización.

"Muchas gracias"

CONTENIDO

Pág.

CAPITULO I
DIAGNÓSTICO Y ENUNCIADO DEL PROBLEMA

CAPITULO II
REVISIÓN DE LA LITERATURA

CAPITULO III
METODOLOGÍA DE LA INVESTIGACIÓN

CAPITULO IV
ANÁLISIS Y RESULTADOS DE LA INVESTIGACIÓN

CAPÍTULO V
CONCLUSIONES Y RECOMENDACIONES

RESUMEN

En la actualidad, existen diversas metodologías e infraestructuras tecnológicas para desarrollar software a medida y de calidad, sin embargo, por ser un proceso complejo surgen diversas dificultades en la elección de una metodología y tecnologías para la implementación. Por tal motivo el propósito de esta investigación es desarrollar un software, mediante tecnologías utilizadas en los proyectos Java EE, encajados a la adaptabilidad de componentes y servicios de la infraestructura Java Platform Enterprise Edition y la arquitectura técnica de la metodología ICONIX.

El objetivo es determinar que la infraestructura Java Platform Enterprise Edition es un mecanismo de adaptabilidad a la Arquitectura Técnica de la Metodología Ágil y Formal ICONIX, 2019. Mediante el caso de estudio: Registro de Incidencia de Bienes Informáticos, UGEL Huamanga.

En la presente investigación se busca abstraer conceptos relevantes en común, entre la arquitectura técnica de la metodología ICONIX y la infraestructura Java Platform Enterprise Edition, para adaptar tecnologías utilizadas en proyectos Java EE y proponer un modelo referencial para el desarrollo ágil de software con escalabilidad, accesibilidad y manejabilidad.

PALABRAS CLAVE

Java Platform Enterprise Edition, Iconix, Patrón mvc, Spring mvc, Hibernate, HTTP y Servlets.

INTRODUCCIÓN

El Modelo Vista Controlador, es un patrón de arquitectura de software, que define en que capas estructuramos lógicamente una aplicación y define las responsabilidades y la relación entre si de cada capa. En los proyectos Java EE, existe la tecnología Spring MVC, un framework orientado a controles, que ha optado por una filosofía diferente en la cual lo más importante es el patrón MVC.

Mi motivación para proponer un modelo referencial, para el desarrollo ágil de softwrare mediante la adaptabilidad entre la infraestructura Java EE y la arquitectura técnica de la metodología Iconix, es la entrega de un producto softwrare a medida y de calidad con los principios de escalabilidad, accesibilidad, y manejabilidad.

Realizar un estudio sobre la adaptabilidad de la infraestrura Java Platform Enterprise Edition a la Arquitectura Técnica de la metodología Iconix, es importante, porque proporciona una guía de tecnologías claramente definidas para la implementación de cada capa de aplicación.

Los principales objetivos específicos son: a) Utilizar la infraestructura de Mapeo Objeto Relacional Hibernate de la Java Platform Enterprise Edition como mecanismo de adaptabilidad a la Arquitectura Técnica de la Metodología Ágil y Formal Iconix b) Utilizar la infraestructura de Framework basado en HTTP y Servlets de la Java Platform Enterprise Edition como mecanismo de adaptabilidad a la Arquitectura Técnica de la Metodología Ágil y Formal ICONIX. c) Utilizar la infraestructura de las clases controladoras del Spring MVC de la Java Platform Enterprise Edition como mecanismo de adaptabilidad a la Arquitectura Técnica de la Metodología Ágil y Formal Iconix.

CAPITULO I

DIAGNÓSTICO Y ENUNCIADO DEL PROBLEMA

1.1. DIAGNÓSTICO Y ENUNCIADO DEL PROBLEMA

La norma ISO / IEC / IEEE 12207 (2017), es el estándar que establece un análisis del ciclo de vida del software, incluye procesos y actividades que contribuyen al desarrollo y entrega de software de calidad, pero no obstante, a la hora de desarrollar un software de calidad, surgen problemas en la eleccion de una metodología y tecnologías, sino también, si dichas tecnologías se adaptan a una arquitectura del sistema para el desarrollo ágil de software, para implantar en organizaciones pequeñas y medianas, como lo son la mayoría en nuestro país.

Según Medina, et al. (2015), en una revista publicada sobre educación en ingeniería, afirma que elegir una metodología para el desarrollo de software, es una difícil decisión. Siendo las metodologías, los marcos que permiten la estructuración, planificación y control de un proyecto, sean estas las tradicionales, también conocido como pesadas, o las ágiles. Muchas veces, en lugar de facilitar las actividades requeridas para terminar el proyecto, estas se hacen complejas y retrasan los proyectos, por la cantidad de entregables e hitos que se deben realizar en cada etapa del proyecto. Estas son las razones que conllevan a reafirmar que elegir una metodología para el desarrollo de software no es una decición fácil.

Por otra parte, Según Ordax y Ocaña (2012), existen problemas originadas del modelo de una aplicación Java EE multicapa. Las aplicaciones que manejan acceso a datos, ejecutan distintas presentaciones y tienen lógica de negocio compleja, acostumbran a sufrir un problema a la hora de mantenerlas, debido a interdependencias entre todos los componentes. Dichas interdependencias obstaculizan la reutilización de código, obligando a rescribir más veces de deseadas una lógica muy parecida, para lo cual, el patrón MVC viene hacer la solución a estos problemas, separando el acceso a datos de la lógica de negocio y esta de la presentación.

Por otro lado, MVC es un patrón de arquitectura de software, que define en que capas estructuramos lógicamente una aplicación y define las responsabilidades y la relación entre si de cada capa. En los proyectos Java EE, existe la tecnología Spring MVC, un framework orientado a controles, que ha optado por una filosofía diferente en la cual lo más importante es el patrón MVC.

1.2. FORMULACIÓN DEL PROBLEMA DE INVESTIGACIÓN

1.2.1. PROBLEMA GENERAL

¿Cómo la **infraestructura Java Platform Enterprise Edition,** es un mecanismo de adaptabilidad a la Arquitectura Técnica de la Metodología Ágil y Formal ICONIX, 2019?

1.2.2. PROBLEMAS ESPECÍFICOS

a. ¿Cómo la infraestructura de **Mapeo Objeto Relacional Hibernate** de la Java Platform Enterprise Edition es un mecanismo de adaptabilidad a la Arquitectura Técnica de la Metodología Ágil y Formal ICONIX, 2019?

b. ¿Cómo la infraestructura de **Framework basado en HTTP y Servlets** de la Java Platform Enterprise Edition es un mecanismo de adaptabilidad a la Arquitectura Técnica de la Metodología Ágil y Formal ICONIX, 2019?

c. ¿Cómo la infraestructura de **las clases controladoras del Spring MVC** de la Java Platform Enterprise Edition es un mecanismo de adaptabilidad a la Arquitectura Técnica de la Metodología Ágil y Formal ICONIX, 2019?

1.3. OBJETIVOS DE LA INVESTIGACIÓN

1.3.1. OBJETIVO GENERAL

Determinar que la **infraestructura Java Platform Enterprise Edition** es un mecanismo de adaptabilidad a la Arquitectura Técnica de la Metodología Ágil y Formal ICONIX, 2019

1.3.2. OBJETIVOS ESPECÍFICOS

a. Utilizar la infraestructura de **Mapeo Objeto Relacional Hibernate** de la Java Platform Enterprise Edition como mecanismo de adaptabilidad a la Arquitectura Técnica de la Metodología Ágil y Formal ICONIX, 2019.

b. Utilizar la infraestructura de **Framework basado en HTTP y Servlets** de la Java Platform Enterprise Edition como mecanismo de adaptabilidad a la Arquitectura Técnica de la Metodología Ágil y Formal ICONIX, 2019.

c. Utilizar la infraestructura de las **clases controladoras del Spring MVC** de la Java Platform Enterprise Edition como mecanismo de adaptabilidad a la Arquitectura Técnica de la Metodología Ágil y Formal ICONIX, 2019.

1.4. HIPÓTESIS DE IVESTIGACIÓN

"Las investigaciones de tipo descriptivo no requieren formular hipótesis; es suficiente plantear algunas preguntas de investigación que, como ya se anotó, surgen del planteamiento del problema, de los objetivos y, por supuesto, del marco teórico que soporta el estudio" (Bernal, 2010, p.136).

La investigación que se desarrollara es de tipo descriptivo, por lo que no se pretende pronosticar, hallar o verificar lo planteado en los objetivos a través de hipótesis que está sujeta a pruebas estadísticas, por lo cual, se optó por no plantear hipótesis, ni probarlas.

1.5. JUSTIFICACIÓN Y DELIMITACIÓN DE LA INVESTIGACIÓN

1.5.1. IMPORTANCIA DEL TEMA

A. IMPORTANCIA TÉCNICA

La importancia del presente trabajo de investigación, reside en proponer un modelo referencial para el desarrollo ágil de software en la plataforma Java EE, considerando una metodología ágil y formal ICONIX y tomando como caso de estudio, Registro de incidencias de bienes informáticos de la UGEL Huamanga. Dicho modelo referencial permitirá la entrega de un producto softwrare a medida y de calidad con los principios de escalabilidad, accesibilidad, y manejabilidad.

B. IMPORTANCIA ECONÓMICA

La importancia del presente trabajo de investigación, reside en en proponer un modelo referencial para desarrollar software con escalabilidad, teniendo en cuenta que este termino es una propiedad de un sistema que indica su habilidad para estar preparado para el crecimiento continuo, sin perder calidad en sus servicios, esto permitirá reducir el costo de los cambios generados en etapas posteriores en el desarrollo del proyecto.

"Las inversiones en software se caracterizan por un fuerte incremento del valor añadido, de la creación de empleo y de los gastos en investigación y desarrollo (I+D), por lo que repercuten de manera considerable en la productividad y competitividad de las empresas y en la evolución de la economía en general". (García, 2007, p.110)

1.5.2. JUSTIFICACIÓN

Según Ordax y Ocaña (2012). "Las aplicaciones Java EE multicapa, suelen sufrir un problema a la hora de mantenerlas, debido a interdependencias entre todos los componentes. Dichas interdependencias dificultan la reutilización de código, obligando a rescribir más veces de deseadas una lógica muy parecida". El patrón Modelo Vista Controlador, viene hacer la solución a estos problemas, separando en las capas de acceso a datos, lógica de negocio y la presentación.

Por lo tanto, una vez que se tiene los componentes de Java EE, encajados en el diseño del patron MVC, se adapta a la arquitectura técnica de la metodología Ágil y Formal Iconix, para el proceso de implementación, permitiendo la optimización de código y haciendo más fácil el desarrollo de sosftware.

1.5.3. DELIMITACIÓN

En el presente trabajo deinvestigación, nos delimitaremos al análisis de la arquitectura técnica de la metodología ágil y formal iconix y a la infraestructura Java Platform Enterprise Edition, ya que el primero aplica componentes y servicios encajados en la arquitectura MVC, mientras el segundo se caracteriza por las prácticas de programación para desarrollar software de aplicaciones en el lenguaje de programación Java, teniendo como tecnologías utilizadas para desarrollar un proyecto java, el Hibernate, Framework basado en HTTP y Servelets y el Spring MVC.

CAPITULO II

REVISIÓN DE LA LITERATURA

2.1. ANTECEDENTES DE LA INVESTIGACIÓN

Yacuri (2018), en su investigación, "Adaptabilidad del Patrón MVC del Framework .Net a la Arquitectura Técnica de la Metodología Ágil y Formal ICONIX. Caso de estudio: Registro de Estudios Ambientales del DREMA", al concluir, llegó a la conclusión de; que el Entity Framework del Patrón MVC del framework .Net, el Motor de Vista Razor del Patrón MVC del framework. Net y las Clases Controladores C# del Patrón MVC del framework .Net son mecanismos de adaptabilidad a la Arquitectura Técnica de la Metodología Ágil y Formal ICONIX.

Acosta (2013), en su investigación, "ESTUDIO DE PATRONES DE DISEÑO EN PLATAFORMA JAVA ENTERPRISE EDITION VERSIÓN 6 PARA EL DESARROLLO DE APLICACIONES WEB", afirma que "la plataforma Java EE adopta un modelo distribuido de aplicaciones multinivel para aplicaciones empresariales, donde la lógica de la aplicación se divide en componentes según su función. Utiliza una lógica de programación desarrollada en niveles o capas, que permite encapsular o separar elementos de las aplicaciones en partes definidas, para simplificar el desarrollo y esclarecer las responsabilidades de cada uno de los componentes de un aplicativo".

2.2. MARCO TEÓRICO

2.2.1. JAVA PLATFORM ENTERPRISE EDITION

Según Galindo y Camps (2008), "J2EE, es una plataforma abierta y estándar para desarrollar y desplegar aplicaciones empresariales multicapa con n-niveles, basadas en servidor, distribuidas y basadas en componentes".

Según Perrone y Chaganti (2003), Java 2 Enterprise Edition (Java2EE), "es una plataforma de programación, para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java. Permite utilizar arquitecturas de N capas distribuidas y se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones". J2EE incluye numerosas API y herramientas Java que incluyen Enterprise JavaBeans, JavaServer Pages y Servlets.

2.2.1.1. ARQUITECTURA MULTICAPA CON N-NIVELES

Según Galindo y Camps (2008), "J2EE extiende la arquitectura propia de tres capas para fijaruna arquitectura en n-capas. Esta extensión se hace de forma normal, estimando que la capa intermedia se puede subdividir en el nivel lógico en diferentes capas, cada una responsabilidades diferenciadas".

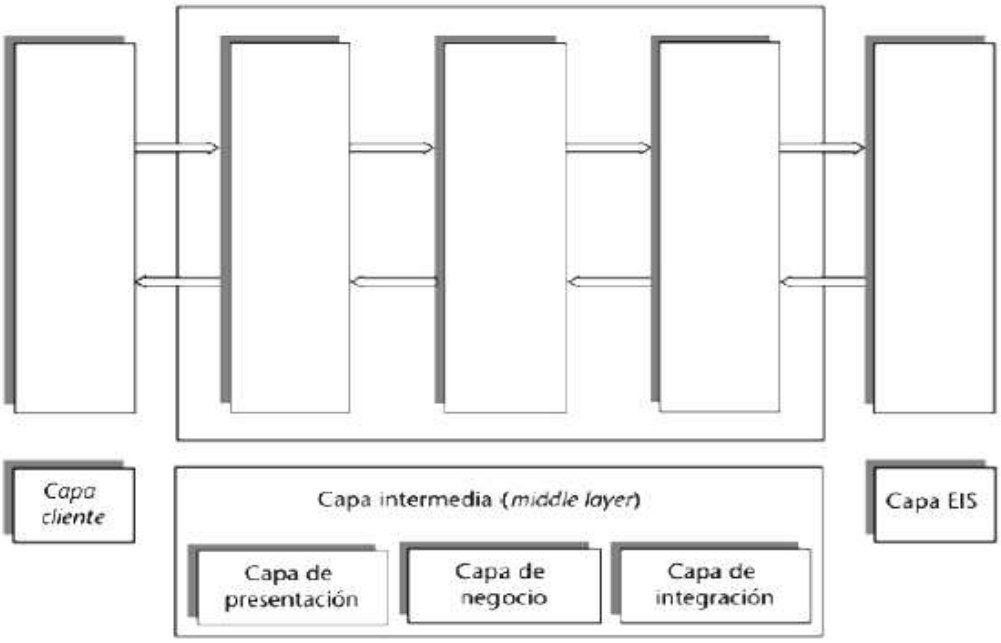


Figura 1. Capas en las que se puede dividir una aplicación J2EE (Galindo y Camps, 2008)

A. Capa Cliente

Según Galindo y Camps (2008), "Comprende a los componentes que se ejecutan en las máquinas cliente, ejemplo, páginas HTML que se ejecutan en los navegadores de los usuarios".

B. Capa Intermedia

Según Galindo y Camps (2008), “middle layer en el nivel físico, podemos tener la capa de presentación y la capa de negocio en la misma máquina física o bien en máquinas diferentes)”. Por lo genral, contiene el procesamiento de la lógica de negocio, situada entre la capa cliente y la capa EIS. En la arquitectura J2EE, se divide la capa intermedia, en tres subcapas, estas conformadas por distintos componentes que son ejecutadas en diferentes contenedores y que tienen distintas funcionalidades:

a. Capa de presentación. Esta capa opera la lógica de interacción que vienes a ser entre el usuario y la aplicación, así como tambien la forma con la que se presenta la información. Dicha capa concierne componentes de presentación las cuales se ejecutan en el contenedor web del servidor de aplicaciones.

b. Capa de negocio. Esta capa concierne el código y tambien las reglas de negocio para la aplicación. J2EE declara los componentes de negocio como EJB, las cuales se ejecutan en el contenedor de EJB del servidor de aplicaciones.

c. Capa de integración. Esta capa en la que se realiza la integración de la aplicación con los distintos sistemas con los que tiene que interactuar (bases de datos, sistemas legacy, etc.). Uno de los componentes de la capa de integración que fija J2EE, vienen a ser los EJB de entidad, la cuales también se ejecutan en el contenedor de EJB del servidor de aplicaciones. (p. 21).

C. Capa EIS

Según Galindo y Camps (2008), “En esta capa, se encuentra, todo el resto de los componentes del sistema de información que hay en la empresa”. Por ejemplo, esta capa contiene componentes de la capa de datos, las cuales residen en un servidor de datos (como podrían ser las tablas de una base de datos relacional).

Según Ordax y Ocaña (2012), las distintas capas son:

a. **Capa cliente (Client Tier):** Es la responsable de la interacción con el usuario.
b. **Capa Web (Web Tier):** Es la responsable del control de la aplicación.
c. **Capa de negocio (Businnes Tier):** Es la responsable de la lógica de la aplicación.
d. **Capa de datos (EIS Tier):** Es la responsable de la persistencia de datos y/o lógica especializada. Por ejemplo, ERPs, BBDD, etc. (p. 5)

En tal sentido según Ordax y Ocaña (2012), "a continuación, mostramos este concepto de multicapa en un diagrama".

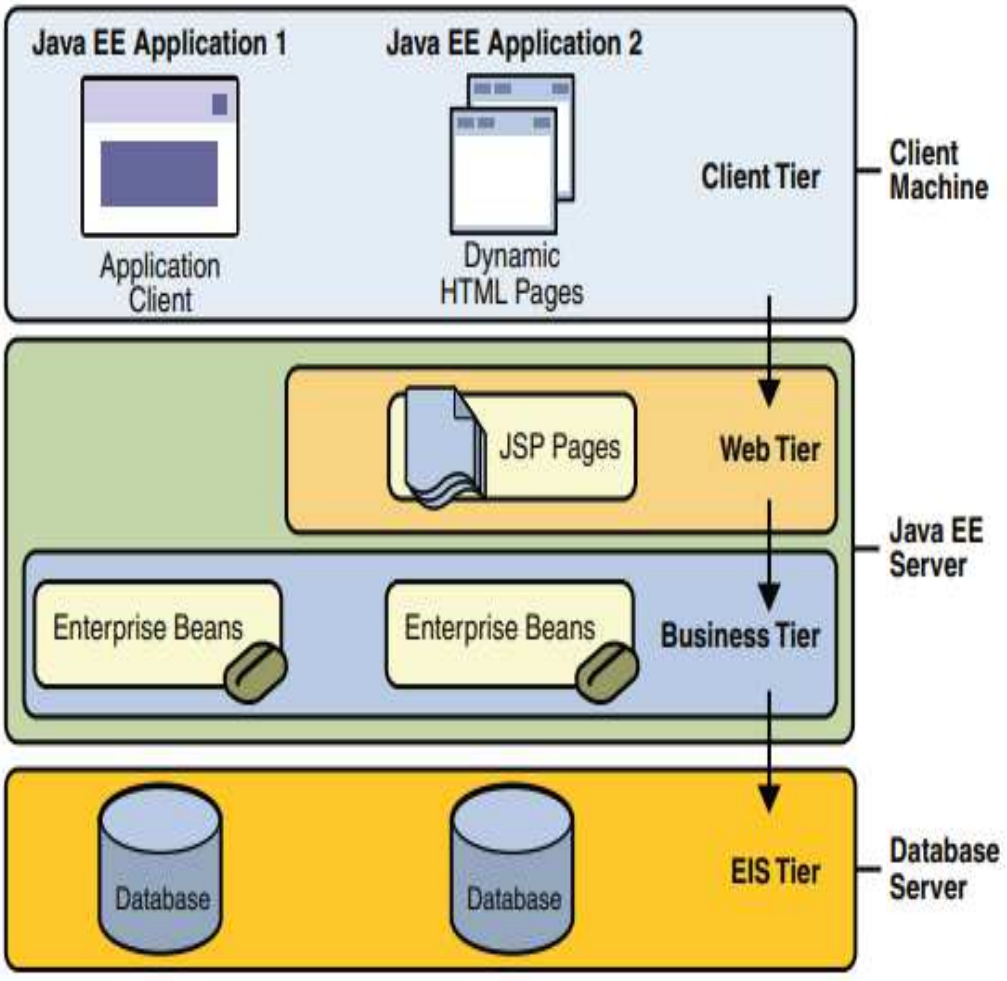


Figura 2. Arquitectura multi-nivel Java EE (Ordax y Ocaña, 2012)

2.2.2. MODELO DE APLICACIÓN JAVA EE

Según Ordax y Ocaña (2012), "la plataforma Java EE, utiliza un modelo de programación distribuido en distintas capas. La lógica de la aplicación se divide en distintos componentes, dependiendo de su funcionalidad, y estos son desplegados en distintas capas según a la cuál pertenecen".

Según Galindo y Camps (2008), "El modelo de programación de aplicaciones de J2EE, esta basado en conceptos de componentes, contenedor y servicio, establece la interelación que existe entre éstos".

2.2.1.1. COMPONENTES

Según Ordax y Ocaña (2012) "un componente, es una unidad de software que forma o compone una aplicación".

A. Componentes cliente.

Según Galindo y Camps (2008), "es un programa que se ejecuta en un entorno cliente, ya sea como applet o bien como aplicación Java stand-alone".

B. Componentes web

Según Galindo y Camps (2008), "es una pieza de software que genera una respuesta a una petición HTTP. J2EE define dos tipos de componentes web, los servlets y las Java Server Pages (JSP)".

C. Componentes de negocio

Según Galindo y Camps (2008), " es una pieza de software que implementa algún concepto de la lógica del dominio particular de negocio en el que se desarrolla la aplicación. J2EE define los Enterprise Java Beans (EJB) como componentes de negocio, y pueden ser de tres tipos: EJB de sesión, EJB de entidad y EJB de mensaje".

2.2.1.2. CONTENEDORES

Según Ordax y Ocaña (2012), “son entornos de ejecución donde se ejecutan los componentes”.

A. Contenedor de aplicaciones

Según Galindo y Camps (2008), “contiene aplicaciones Java stand-alone. Lo encontramos en las máquinas cliente con aplicaciones Java”.

B. Contenedor de applets

Según Galindo y Camps (2008), “proporciona un entorno de ejecución por applets. Lo encontramos en los navegadores de las máquinas cliente”.

C. Contenedor web

Según Galindo y Camps (2008), “proporciona un entorno de ejecución para los componentes web que define la plataforma J2EE, servlets y JSP. Lo encontramos en el servidor”.

D. Contenedor de EJB

Según Galindo y Camps (2008), “proporciona un entorno de ejecución para los EJB, los componentes de negocio que ofrece J2EE. Lo encontramos en el servidor”.

2.2.1.3. SERVICIOS

Según Ordax y Ocaña (2012,) “son funcionalidades estándar que todo contenedor debe proveer a los componentes”.

En tal sentido según, Galindo y Camps (2008), afirma:

a. “Los servicios configurables permiten a los desarrolladores especificar el comportamiento de los componentes en tiempo de despliegue de la aplicación. Entre éstos, podemos citar los servicios de transacciones, de seguridad y el servicio de nombres”.
b. “Los contenedores también ofrecen servicios no configurables, como la gestión del ciclo de vida de los componentes, la gestión de los pools de conexiones a recursos externos, el acceso a las API de J2EE, etc”. (p.19)

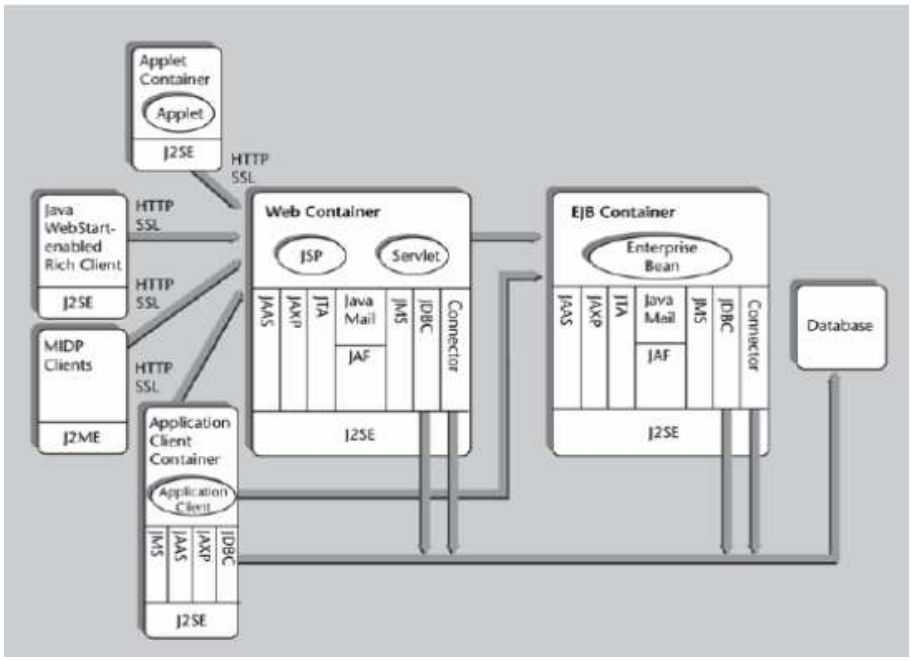


Figura 3. Componentes, contenedores y servicios de la plataforma J2EE (Galindo y Camps, 2008)

Según Ordax y Ocaña (2012), "Los distintos tipos de componentes introducido en el apartado del modelo de aplicación Java EE, encajan perfectamente en este diseño":

a. Modelo: Enterprise JavaBeans, POJOS (Palin Old Java Objects).

b. Vista: JavaServer Pages (JSP), JavaServer Faces (JSF).

c. Controlador: Java Servelets.

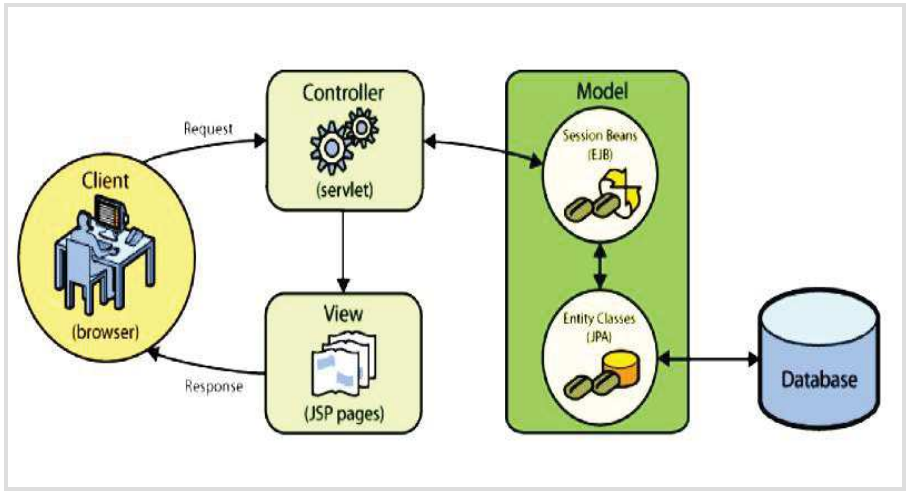


Figura 4. Flujo de aplicación Java EE (Ordax y Ocaña, 2012)

Según Ordax y Ocaña (2012), “Las aplicaciones que manejan acceso a datos, utilizan diversas presentaciones y poseen una lógica de negocio compleja, estas conllevan a un problema, a la hora de mantenerlas, debido a la existencia de interdependencias entre todos los componentes”. Por la cual, dichas interdependencias, dificultan la reutilización de código, obligando a reescribir mas veces de las deseadas una lógica muy parecida.

El patrón de diseño MVC, recuelve los problemas genaerados por las interdependencias, de manera que desacopla el acceso a datos de la lógica de negocio y esta de la presentación. De tal manera, se podrá reutilizar un acceso desde diferentes funcionalidades, o utilizar la misma funcionalidad desde distintos tipos de presentación, haciendo más fácil el próximo mantenimiento. (p. 9).

2.2.3. PATRÓN MODELO VISTA CONTROLADOR

Según Flores y Acuña, (2014), “El MVC, es un patrón de arquitectura, de software, que divide los datos de una aplicación, la interfaz gráfica de usuario, y la lógica de control en tres partes distintas: modelo, vista y controlador”.

Según Talledo (2015), “El MVC, es un patrón de arquitectura de software, que divide los datos y la aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones”. Para ello, el MVC propone separar en tres distintos componentes: el modelo, la vista y el controlador.

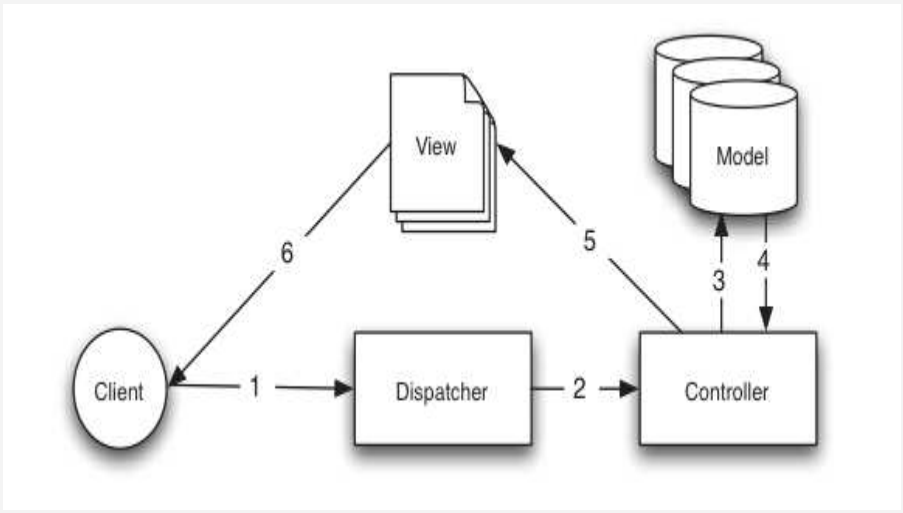


Figura 5. El modelo vista -controlador (Talledo, 2015)

Según Rosenber y Stephens (2007), se trata de un conocido patrón de diseño que se ajusta muy bien con las clases interfaz, entidad, y control utilizadas por el proceso ICONIX, y es casi un patrón de diseño para ambas aplicaciones GUI cliente-rico y para un cliente ligero basada en Web (es decir, no se limita a Spring). La premisa detrás de MVC, es que la aplicación sea separado en tres componentes distintos: modelo, vista y controlador.

2.2.3.1. Modelo

Según Talledo (2015), "el modelo, es la capa que trabaja con los datos, contiene mecanismos para acceder a la información y también para actualizar su estado". Es decir, esta capa se refiere a la gestión de accesos a la información, consultas para su respectiva respuesta y tambien del mantenimiento (actualización). Como resultado envía a "vista" aquella parte de la información que se solicita para ser mostrada.

Según Rosenber y Stephens (2007), "es un objeto en representación de los datos, por lo general a leer de una base de datos. La sesión "detrás" del modelo es totalmente detallado para el mapeo de a objetos a tablas, filas, columnas, y relaciones en la base de datos".

2.2.3.2. Vista

Según Talledo (2015), "la vista, es la que se encarga de mostrar la información gestionada. Contiene el código necesario de la aplicación, para mostrar dicha información". En caso de los sitios web, habitualmente es mediante código HTML, CSS y JavaScript, en el caso del navegador que será el encargado de interpretarlo. Pero habrá una parte que se ejecutará por anticipado en el servidor como viene a ser el caso de los scripts, por ejemplo, de PHP.

Rosenber y Stephens (2007), "es el límite entre el ordenador y el usuario. En una aplicación web, la vista generalmente se refiere tanto a la página web como a la plantilla (por ejemplo, JSP o un archivo Velocity) que crea la página web".

2.2.3.3. Controlador

Rosenber y Stephens (2007), " son el pegamento entre la vista y el modelo. Cuando una solicitud es recibida, el controlador busca (o actualiza) la información del modelo, decide que vista mostrar al usuario, y maneja los datos necesarios para la vista". En Spring, un objeto controlador, interpreta la entrada del usuario y lo transforma el resultado en un modelo que se mostrará al usuario en la Vista.

Según Talledo (2015), "También puede enviar órdenes a su vista asociada si se solicita un cambio en la forma en que se presenta de modelo (por ejemplo, desplazamiento o scroll por un documento o por los diferentes registros de una base de datos)". Por lo tanto, se dice que el "controlador", viene a ser el intermediario entre la "vista" y el "modelo".

2.2.4. MAPEO OBJETO RELACIONAL

Del Busto y Enriquez (2012), " Es una técnica de programación para convertir datos del sistema de tipos utilizado en un lenguaje de programación orientado a objetos al utilizado en una base de datos relacional". En la práctica el mapeo objeto relacional, crea una base de datos orientada a objetos sobre la base de datos relacional, posibilitnado la utlización de las características de la orientación a objetos en especial la herencia y el polimorfismo.

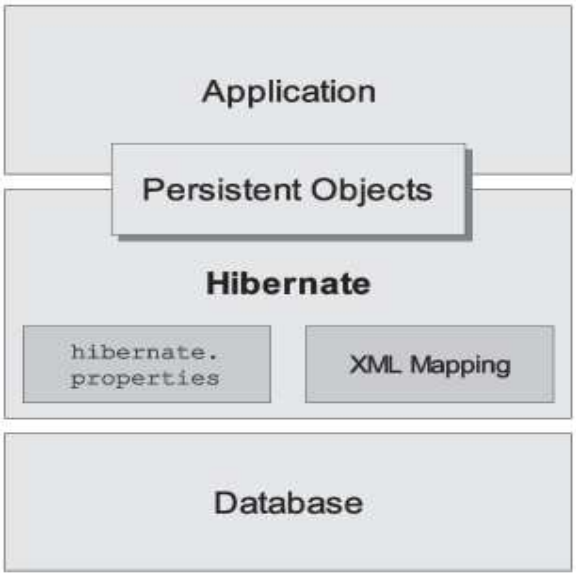


Figura 7: Arquitectura orm (ORM ,2013).

El principal objetivo de un ORM, es delegar en utilidades extremas (por ejemplo, el Hibernate) la tarea de crear una correspondencia entre objetos y tablas Según (Ceballos, 2015).

2.2.5. HIBERNATE

Según Rios (2015), " 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".

Hibernate es un proyecto ambicioso que tiene como objetivo proporcionar una solución completa al problema de la gestión de datos persistentes en Java. Hoy, hibernate no es solo un servicio ORM, sino también una colección de herramientas de gestión de datos que se extienden mucho más allá de ORM. (Bauer et al, 2015).

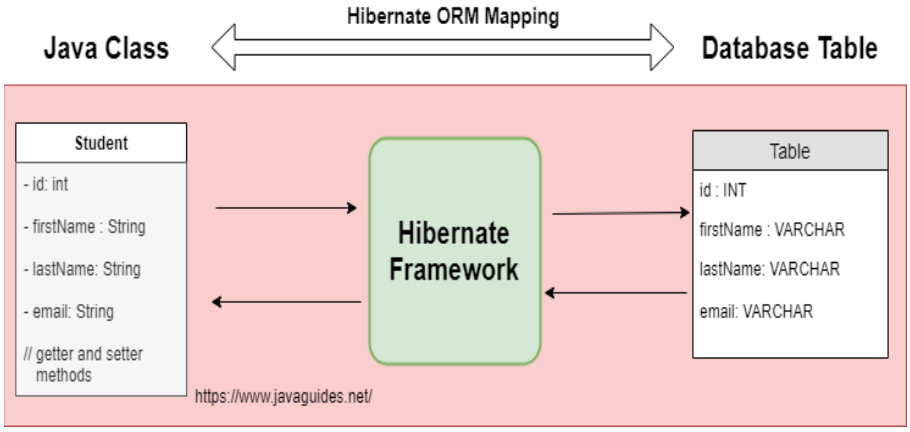


Figura 8. Mapeo de Hibernate orm: (Fadatare, 2018)

2.2.6. SPRING

Según Sergio (2015), "Spring es una plataforma para el desarrollo de aplicaciones Java, con el que puedes crear una aplicación web JEE completa, ya que

cuenta con varios módulos que soportan la estructura completa de una arquitectura JEE". Algunos módulos muy utilizados son: Spring MVC, Spring Tiles, para la capa de presentación. Spring JDBC, Spring loC, Spring AOP para la capa de procesamiento.

Según Rosenberg y Stephens (2007), es generalmente considerada como un Framework ligero de aplicaciones J2EE, a pesar de eso no se limita necesariamente a J2EE. Sin embargo, Spring permite diferentes Frameworks a ser "enchufado" para manejar mapeo relacional de objetos (ORM), vistas/plantillas, etc. En el lado ORM de las cosas, Spring tiene soporte para JDBC, Hibernate, Java Data Objects (JDO), y iBatis. En el lado de las Vistas, Spring tiene soporte para una serie de plantillas/soluciones del contenido web, incluyendo JSP, Velocity, y Struts.

Según Johnson, et al. (2010), "Spring Framework es un importante marco de desarrollo de aplicaciones de código abierto, que hace que el desarrollo de Java / J2EE (TM) sea más fácil y más productivo".

A. SPRING MVC

Según Hemrajani (2006), "Spring Web MVC Framework, es un marco robusto, flexible y bien diseñado para el desarrollo rápido de aplicaciones web, utilizando el patrón de diseño MVC. Los beneficios obtenidos al usar este módulo Spring son similares a los que obtiene del resto del Spring Framework".

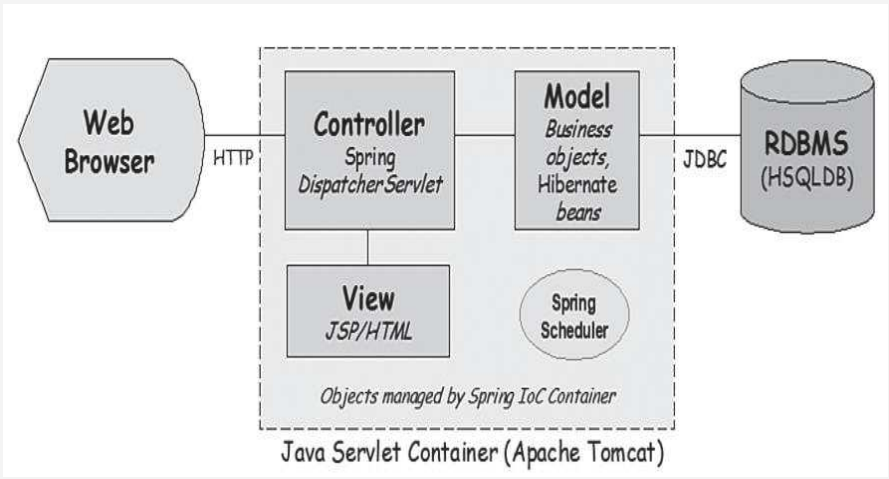


Figura 9. Diagrama de arquitectura de alto nivel. (Hemrajani, 2006).

Según la Figura 9. Se puede ver, todas las solicitudes HTTP entrantes de un navegador web son manejadas por Controladores. Un controlador, como su nombre indica, controla la vista y el modelo al facilitar el intercambio de datos entre ellos. El beneficio clave de este enfoque es que el modelo puede preocuparse solo acerca de los datos y no tiene conocimiento de la vista. La vista, por otro lado, no tiene conocimiento del modelo y la lógica de negocios y simplemente representa los datos que se le pasan (como una página web, en nuestro caso). (Anil Hemrajani, 2006).

El Framework Web se centra alrededor de Servlet de Java, llamado DispatcherServlet. Como su nombre lo sugiere, este servlet despacha los requisitos a los controladores y proporciona algunas funciones adicionales de las que las aplicaciones web pueden hacer un mejor uso (Rosenber y Stephens,2007).

2.2.7. FRAMEWORK BASADO EN HTTP Y SERVLETS

A. FRAMEWORK

Según Rodriguez D. (2012). “es una implementación compuesta de un conjunto de componentes y librerías empleados para desarrollar una estructura estándar de una aplicación, que, sumado a una metodología de uso y su correspondiente documentación nos permitirá diseñar, analizar, implementar y desplegar aplicaciones de forma estándar, de alta calidad y rapidez”.

B. HTTP

Según Galindo y Camps (2008), “HTTP y el paradigma petición-respuesta La capa web de una aplicación J2EE está desarrollada sobre el protocolo http siguiendo el paradigma de petición-respuesta”. Dicho ello los componentes web reciben peticiones http del servidor web, las procesan y generan una respuesta según esta petición. Existe un servidor en la capa web que recibe las peticiones HTTP de los clientes. “Si éstas van dirigidas a un componente web, las pasa al contenedor web, que se encargará de procesarlas y retornar la respuesta al cliente. Si la petición no va dirigida a ningún componente web (por ejemplo, una petición de una imagen, de una página HTML estática o de una película de Flash), el servidor web envía la respuesta al cliente sin pasar por el contenedor web”. (p. 29).

C. SERVLETS

Según Galindo y Camps (2008), "Un servlet es un objeto Java que extiende la funcionalidad de un servidor web, recibe peticiones HTTP y genera contenido dinámico como respuesta a estas peticiones". Los servlets se encuentran dentro de un contenedor web y se cargan y ejecutan dinámicamente como respuesta a las peticiones de los clientes. Su funcionamiento se basa en el paradigma de petición-respuesta, sobre el protocolo htttp.

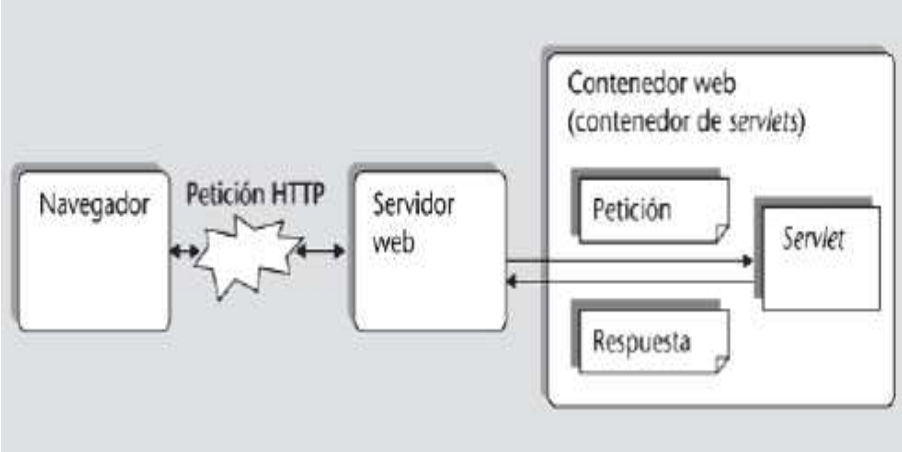


Figura 10. Paradigma de petición respuesta (Galindo y Camps, 2008)

2.2.8. METODOLOGÍA ICONIX

"El proceso ICONIX, es una metodología de desarrollo de software, orientado a objetos, utilizando UML. Es una metodología de desarrollo práctico impulsado por casos de uso y consiste en flujos de trabajos dinámicos y estáticos". El flujo de trabajo dinámico fluye de modelos de casos de uso a diagramas de robustez, y luego de diagramas de robustez a diagramas de secuencia. Por un lado, el flujo de trabajo estático fluye de los modelos de dominio al diagrama de clase. (Maria Virvou, Saeko Matsuura. 2012, p.173).

Porras (2011), señala que "la metodología ICONIX se encuentra entre el proceso unificado (PU) y el enfoque (XP). ICONIX está dirigido por casos de uso, igual que el proceso unificado, pero sin la sobrecarga de proceso unificado; es relativamente pequeño y ligero".

"El Proceso de Iconix se divide en flujos de trabajo dinámicos y estáticos, que son altamente repetitivo; usted puede ir a través de una repetición de todo el proceso para una pequeña cantidad de casos de uso (quizás un par de paquetes de valor, que no es una cantidad enorme teniendo en cuenta que cada caso de uso es sólo un par de párrafos), todo el camino a la fuente de código y pruebas unitarias. Por esta razón, el proceso Iconix es muy adecuado para proyectos ágiles, donde se necesita información rápida sobre factores tales como los requisitos, el diseño, y las estimaciones" (Rosenberg y Stephens, 2007, p. 35).

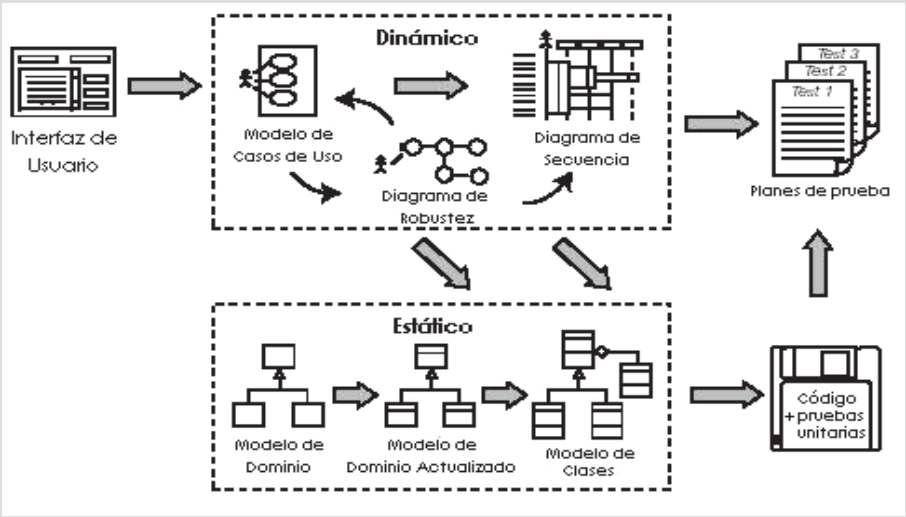


Figura 11. Flujo de trabajo Iconix (Rosenberg y Scott,2007)

Rosenberg y Stephens (2007), "muestra las fases de ICONIX como sigue: análisis de requisitos, diseño preliminar, diseño detallado, implementación y pruebas, lo cual se detalla a continuación".

2.2.8.1. FASES DE LA METODOLOGÍA ICONIX

A. REQUISITOS

Esta actividad asegura que el sistema sea tal como se detalla, concuerda con los requisitos. La revisión de requisitos incluecasi siempre al cliente, el analista de negocios y un miembro del equipo de desarrollo, un usuario final y, un administrador implicado en el proyecto.

En tal sentido Rosenberg y Stephens (2007), afirma que en la fase de requisitos se tiene:

a. Requisitos funcionales

Rosenberg y Stephens (2007), "Definir lo que el sistema debe ser capaz de hacer. Dependiendo de la forma en que su proyecto está organizado, ya sea que usted participe en la creación de los requisitos funcionales o que los requisitos serán "dictados desde lo alto" de un cliente o un equipo de analistas de negocios". (p.10).

b. Modelo de Dominio

Rosenberg y Stephens (2007), "Entender el espacio del problema en términos inequívocos (sin ambigüedad)". (p.10).

c. Requisitos de comportamiento

Rosenberg y Stephens (2007), "Define la forma en que el usuario y el sistema interactúan (es decir, escribir el primer proyecto de casos de uso). Le recomendamos que empezar con un prototipo GUI (lo que llamamos "guión" GUI) e identificar todos los casos de uso que vamos a poner en práctica, o, al menos, llegar a una primera lista de casos de uso, que espera razonablemente cambiar a medida que se explora los requisitos con mayor profundidad". (p.10).

d. Etapa 1: Revisión de Requisitos

Rosenberg y Stephens (2007), "Asegúrese de que la descripción de los casos de uso coincida con las expectativas de sus clientes. Tenga en cuenta que se deben revisar los casos de uso en pequeños lotes, justo antes de diseñarlos". (p.10).

B. ANÁLISIS/DISEÑO PRELIMINAR

Esta fase nos ayuda a cerciorarnos que los diagramas de robustez, el modelo de dominio y la descripción de casos de uso coincida entre sí. Dicha revisión viene a ser el "puente" entre el diseño preliminar y el diseño detallado, se elabora para cada paquete de casos de uso. Las personas implicadas en esta acción es el mismo grupo de la revisión de requisitos.

En tal sentido Rosenberg y Stephens (2007), afirma que en la fase de análisis/diseño preliminar se tiene:

a. "Análisis de robustez: Dibuje un diagrama de robustez (una "imagen del objeto" de los pasos en un caso de uso), reescribiendo los casos de uso a medida que avanza". (p.10)
b. "Actualice el modelo de dominio mientras escribe los casos de uso y dibuje el diagrama de robustez. Aquí descubrirá algunas clases "perdidas", corregirá las ambigüedades, y añadirá atributos a los objetos de dominio (por ejemplo: identificar que un libro tiene un Título, autor, sinopsis, etc.)". (p.10)
c. "Nombre todas las funciones lógicas del software (controladores) necesitadas para hacer que los casos de uso funcionen". (p.10)
d. "Reescriba el primer borrador de casos de uso". (p.10)

C. DISEÑO DETALLADO

Esta fase, viene a ser un paso entre el diseño y la implementación, para tal fin se utiliza tres criterios: hacer coincidir los diagramas de secuencia con la descripción de los casos de uso, revisar y verificar la continuidad de los mensajes para un buen diseño. Si se diseña cada caso de uso aplicado los tres criterios, entonces el diseño está listo para la codificación.

En tal sentido Rosenberg y Stephens (2007), afirma que en la fase de diseño detallado se tiene:

a. "Análisis de robustez: Dibuje un diagrama de robustez (una "imagen del objeto" de los pasos en un caso de uso), reescribiendo los casos de uso a medida que avanza". (p.11)
b. "Actualice el modelo de dominio mientras escribe los casos de uso y dibuje el diagrama de robustez. Aquí descubrirá algunas clases "perdidas", corregirá las ambigüedades, y añadirá atributos a los objetos de dominio (por ejemplo: identificar que un libro tiene un Título, autor, sinopsis, etc.)". (p.11)
c. "Nombre todas las funciones lógicas del software (controladores) necesitadas para hacer que los casos de uso funcionen". (p.11)

D. IMPLEMENTACIÓN

Durante la implementación se han hecho cambios al diseño, entonces el código no concuerda con los diagramas de diseño. ICONIX, usando la herramienta Enterprise Architec, proporciona una técnica para mantener y actualizar el diseño, este será la base del diseño del resto del proyecto.

En tal sentido Rosenberg y Stephens (2007), afirma que en la fase de diseño detallado se tiene:

a. **"**Código/pruebas unitarias: Escriba el código y las pruebas unitarias. (o, dependencia de sus preferencias, escriba las pruebas unitarias y luego el código)". (p.11)

b. **"**Integración y pruebas de escenario: Base las pruebas de integración en los casos de uso, de modo que así probara ambos cursos, el básico y el alterno". (p.11)

c. **"**Ejecute una revisión de código y actualice el modelo para prepararse para la próxima ronda de trabajo". (p.11)

2.2.8.2. ARQUITECTURA TÉCNICA

Según Rosenberg y Stephens (2007), "La Arquitectura técnica (también conocida como arquitectura del sistema y arquitectura del software) generalmente describe el sistema que se intenta construir en términos de estructura". el objetivo de arquitectura técnica (AT) es obtener un sentido general del sistema que vas a desarrollar. ¿Será un sistema basado en Internet? O un sistema en VB. ¿NET o Java Swing, para un cliente muy rico? Es necesario utilizar un framework de aplicación específico (por ejemplo, un framework de una compañía estándar).

No hay una notación estándar o un formato para documentar la AT, la profundidad y el formato de la arquitectura técnica-y los convenios para crearla-varían mucho de empresa a empresa (p.159).

En tal sentido, Rosenberg y Stephens (2007), señala 10 principios para hacer una arquitectura técnica, los cuales se resumen como una lista de concejos como sigue:

a. Separe la funcionalidad, los datos, y la arquitectura del sistema.
b. Entender la razón por la cual estamos creando una arquitectura.
c. Base objetivamente la arquitectura en los requisitos.
d. Considere factores como la escalabilidad, seguridad y disponibilidad.
e. Considere la posibilidad de internacionalización y localización.
f. Plantee preguntas a todas las personas implicadas.
g. Si no tienes las respuestas que necesitas, pregunta otra vez-y otra vez mas
h. Considere la comprobabilidad.
i. Explorar sistemas externos con los que tendrás que interactuar.
j. Tener el valor para creer que su arquitectura es correcta y la fuerza para empujar su aprobación a lo largo del proyecto.

A. ARQUITECTURA EN CAPAS

Según Rosenberg y Stephens (2007), "el sostenimiento de la arquitectura es una metáfora visual del software que hace que un sistema se divide en bandas (capas) en un diagrama de arquitectura".

Podemos clasificar en términos generales a las capas como, ya sea horizontal (aplicable a muchos, si es que no a todas, los dominios de negocios) o vertical (aplicable a través de un subconjunto o un único dominio). (p.162)

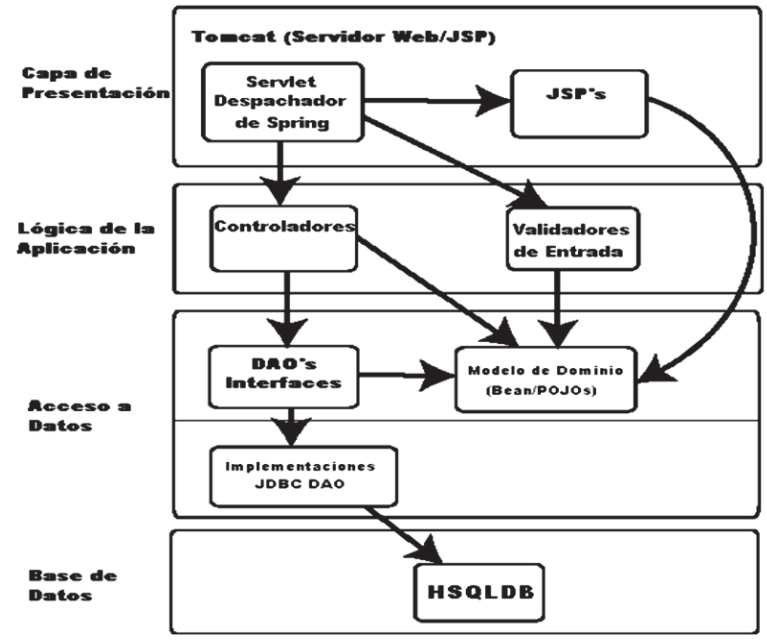


Figura 12. Arquitectura en capas (Rosenberg y Scott,2007)

Según Rosenberg y Stephens (2007), el diagrama parece estar mezclando dos aspectos de la arquitectura: el modelo de despliegue y el modelo de paquete/componente.

En tal sentido, Rosenberg y Stephens (2007), separa en dos diagramas.

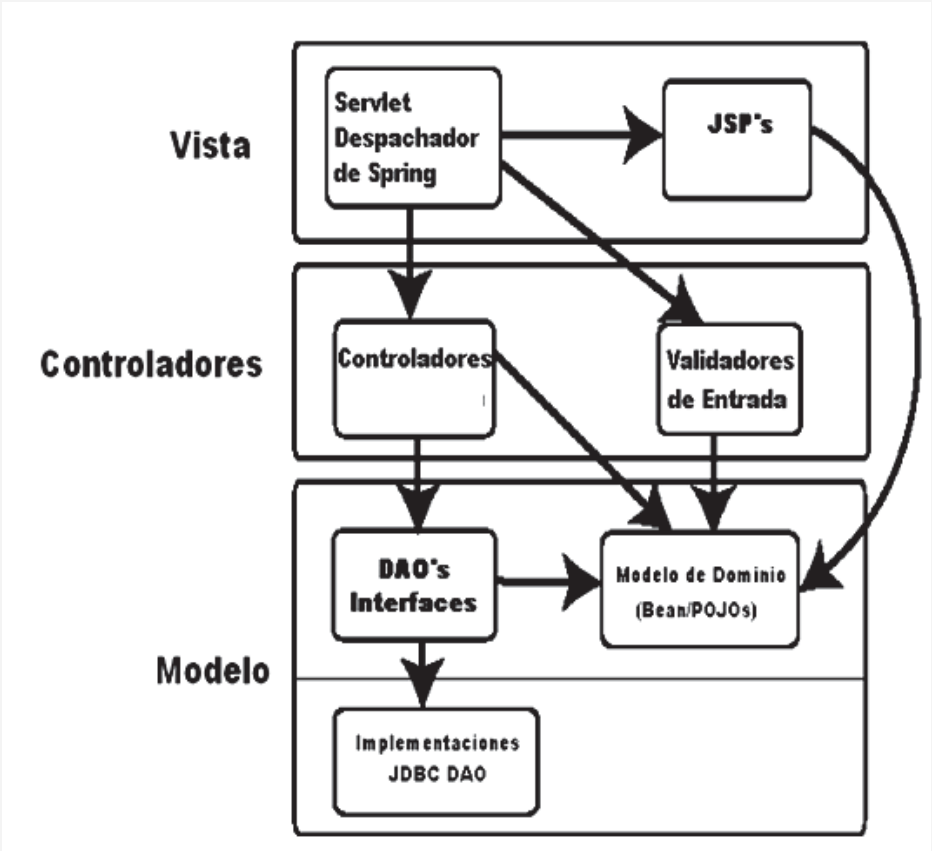


Figura 13. Arquitectura de paquetes/componentes (Rosenberg y Scott,2007)

En la Figura 13, se ha renombrado cada una de las capas, para que sea más evidente que se trata de un clásico mapeo en la arquitectura MVC.

2.2.10. SISTEMA GESTOR DE BASE DE DATOS

Según Osorio (2008), "Un sistema de Manejo de Bases de Datos (DBMS), es un conjunto de herramientas interrelacionados y/o una serie de programas que permiten a los usuarios tener acceso a esta información, en genral para hacer consultas o actualizar".

Según Gómez y De Abajo (1998), ‘’Viene a ser un conjunto de herramientas queayudan a los usuarios en la gestión de información que se encuentra almacenada en una base de datos”.

Figura 14. componentes y tipos de sistemas gestores de bases de datos.

A. MYSQL

Según Thibaud, (2006), “Es un sistema de gestión de bases de datos relacionales (SGBDR) en un entorno de red, especialmente en arquitecturas cliente/servidor”. Trabaja con muchas herramientas y es compatible con varios lenguajes de programación. Es el más compatible con el servidor de páginas web Apache y el lenguaje de páginas web..

2.2.11. LENGUAJE DE PROGRAMACIÓN ORIENTADA A OBJETOS

Según Joyanes (2002), “Es un método de implementación, en el que los programas se organizan como colecciones cooperativas de objetos, los cuales cada uno ellos representan una instancia de alguna clase, que vienen a ser miembros de una jerarquía de clases unidas mediante relaciones de herencia”.

Según Deitel (2004), “La programación orientada a objetos, encapsula datos (atributos) y métodos (Comportamientos) dentro de objetos; Los datos y los métodos de un objeto se encuentran íntimamente ligados entre sí”. Los objetos se caracterizan por ocultar la información, lo cual significa que aunque los objetos sepan cómo comunicarse, atreves de interfaces bien definidas, por lo general a los objetos no se les

permite conocer cómo estan implementados otros objetos, los detalles de implementación están ocultos dentro de los mismos objetos.

2.2.12. IDE NETBEANS

Para Ávila J. (2016), NetBeans es un entorno de desarrollo integrado (IDE) para el desarrollo principalmente con Java, pero también con otros lenguajes, en particular, PHP, C/, C++ y HTML5. Tambien es una plataforma de aplicaciones de ventana para las alicaciones de escritorio Java y otros.

2.2.13. TECNOLOGÍAS DE INTERNET

Según Joyanes (2008), “El internet es conocida como la red de redes y tiene base en la tecnología cliente/servidor”. Las personas que hacen uso de la red, controlan sus tareas mediante aplicaciones web tales como software de navegador. Toda la información incluyendo mensajes de correo y las páginas web son almacenandos en los servidores.

Para Ceballos (2006), “Es una red de redes informáticas, distribuidas anivel mundial, por el cual se intercambian información, por medio de la familia de protocolos TCP/IP. Puede imaginarse Internet como una gran nube con ordenadores conectados”.

A. APLICACIÓN WEB

Según Luján (2001), “Es un tipo especial de aplicación cliente / servidor, en el que el cliente, como el servidor web y el protocolo se comunican mediante el http, están estandarizados y no han de ser creados por el programador de aplicaciones”.

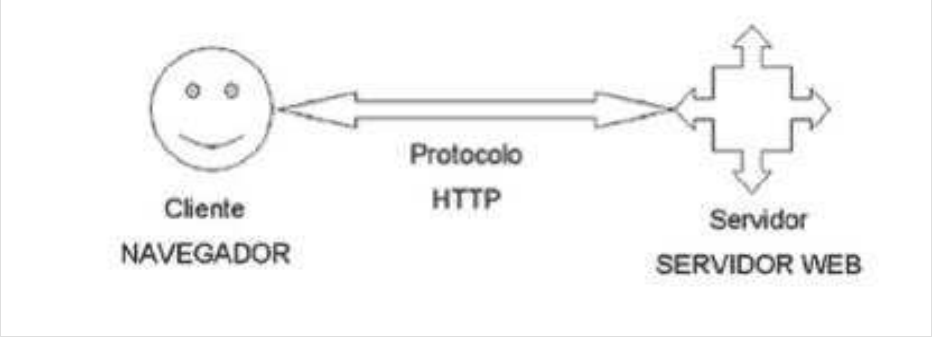


Figura 15. Esquema básico de una aplicación web (Luján, 2001)

B. PROTOCOLO

Según Antonio (2001), "Proviene del griego <<protókolon>>, que significa <<la primera hoja o tapa, encolada, de un manuscrito importante con notas sobre su contenido>>"

Según Gutiérrez y Tena (2003), "Un protocolo es un conjunto definido de etapas, que implica a dos o más partes, designado para realizar una tarea específica".

C. TCP

Según Colección Esencial (2011), "Este protocolo, descompone el mensaje en paquetes y asegura la fiabilidad de la transmisión. Cuando los paquetes llegan a su destino, se agrupan automáticamente para formar un único mensaje, idéntico al original".

Según Romero (1997), "TCP (Transmission Control Protocol), utiliza mensajes IP para lograr una transferencia de datos libre de errores. Ambos establecen un diálogo con otro sistema a base de enviar servicios de mensajes". (p.22)

2.2.14. SERVIDOR APACHE TOMCAT

Según Ordax (2012), "Es un servidor gratuito de código abierto, producto de software que implementa todas las especificaciones Java EE, de manera que al desplegar o instalar una aplicación Java EE en el servidor, sabemos seguro que va a encontrarse con todos los contenedores y servicios definidos por la especificación y que seguramente utiliza y necesita la aplicación".

2.2.10. ISO 12207

Según la ISO / IEC / IEEE 12207: 2017, "proporciona procesos que pueden emplearse para definir, controlar y mejorar los procesos del ciclo de vida del software dentro de una organización o un proyecto".

2.2.15. POBLACIÓN

Según Hernández (2013), "Se entiende por población al conjunto total de individuos, objetos o medidas que poseen algunas características comunes observables en un lugar y en un momento determinado".

Para Chávez (2007), la población "es el universo de estudio de la investigación, sobre el cual se pretende generalizar los resultados, constituida por características o estratos que le permiten distinguir los sujetos, unos de otros".

2.2.16. MUESTRA

Según Supo (2015), "una muestra es una parte de la población que tenemos que estudiar para llevar sus conclusiones desde la muestra hacia la población, a este procedimiento se le conoce como inferencia y se hará efectivo únicamente si hemos seleccionado una muestra representativa".

Según Balestrini (2006), señala que: "Una muestra es una parte representativa de una población, cuyas características deben producirse en ella, lo más exactamente posible".

Según Tamayo y Tamayo (1997), "Afirma que la muestra es el grupo de individuos que se toma de la población, para estudiar un fenómeno estadístico".

A. MUESTREO POR CONVENIENCIA

Según Bavaresco (2006), "Los sujetos de una investigación específica, son seleccionados para el estudio sólo porque son más fáciles de reclutar y el investigador no está considerando las características de inclusión de los sujetos que los hace representativos de toda la población".

CAPITULO III

METODOLOGÍA DE LA INVESTIGACIÓN

3.1. TIPO Y NIVEL DE INVESTIGACIÓN

3.1.1. TIPO DE INVESTIGACIÓN

Según Supo (2012), "los estudios sin intervención del investigador, son denominados estudios observacionales, porque no existe intervención de ningún tipo por parte del investigador". Los datos que se observan son registrados y reflejan la evolución natural de los acontecimientos. Por ende **la investigación es de tipo observacional**.

Así mismo según Supo (2012), "Los estudios de sin intervención, incluyen a los estudios descriptivos, porque no buscan modificar los resultados o datos obtenidos, apareciendo el análisis estadístico de los datos, no se busca demostrar nada". Por ello **la investigación es de tipo Descriptivo.**

3.1.2. NIVEL DE INVESTIGACIÓN

Según Hernández Sampieri Et. Al (2010), "los **estudios descriptivos** buscan especificar las propiedades, las características y los perfiles importantes de personas, grupos, comunidades o cualquier otro fenómeno que se someta a análisis. Es decir, únicamente pretenden medir o recoger información de manera independiente o conjunta sobre las variables a las que se refieren". Esto viene a ser el objetivo, mas no como se relacionan éstas.

De acuerdo al autor Carrasco (2006), señala que, "la **investigación descriptiva** se soporta principalmente en técnicas como la encuesta, la entrevista, la observación y revisión documental. Este tipo de investigación estudia, analiza, describe y especifica situaciones y propiedades de personas, grupos, comunidades o cualquier otro fenómeno u objeto que sea sometido al análisis". Estas consideraciones conllevan a que el nivel de investigación sea descriptivo.

Los trabajos de investigación de grado, pregrado y muchas maestrías, vienen a ser estudios de carácter preponderantemente descriptivo. En dichos estudios se presentan, identificación de hechos, situaciones, rasgos característicos de un objeto de estudio, o se diseñan productos, modelos, prototipos, y guías.

3.2. DISEÑO DE LA INVESTIGACIÓN

Según Hemández Et. Al (2010), "los **diseños no experimentales** se definen; como aquella investigación que se realiza sin manipular deliberadamente las variables, se trata de estudios donde no hacemos variar en forma intencional las variables independientes para ver su efecto sobre otras variables".

"Lo que hacemos en la investigación no experimental es observar fenómenos tal como se dan en su contexto natural, para posteriormente analizarlos" (p.191).

Según Carrasco (2008), el diseño de **investigación transversal descriptivo**, se emplea para analizar y conocer las características, cualidades internas y externas, propiedades y rasgos esenciales de los hechos y fenómenos de un hecho realizado a momento determinado del tiempo.

De acuerdo a Hernández et al (2010), una investigación de **diseño transversal,** "es cuando se recolectan datos en un solo momento o en un tiempo único y su propósito es describir variables y analizar los hechos tal como se dan. Los instrumentos de recolección de datos, son usados durante el proceso de forma única".

En esta investigación se tenía que analizar, procesos de desarrollo de software, con las tecnologías arriba indicadas, conocer características de las herramientas, y entender funcionalidades para encontrar los métodos necesarios para la construcción del software; por lo tanto, el diseño de investigación es **no experimental y transversal descriptivo**.

3.3. POBLACIÓN Y MUESTRA

3.3.1. POBLACIÓN

Conjunto de librerías desarrolladas por Sun Microsystems. Oracle Corporation y la Comunidad Java Open Source.

3.3.2. MUESTRA

Conjunto de librerías desarrolladas por Sun Microsystems. Oracle Corporation y la Comunidad Java Open Source, en específico Hibernate, Spring MVC y Servlets.

3.4. VARIABLES E INDICADORES

3.4.1. DEFINICIÓN CONCEPTUAL DE LAS VARIABLES

Según Supo (2012), cuando pasamos a un nivel descrptivo, la variable de estudio se convierte en la variable de interés, pero solamente por la función que cumple. No ha dejado de ser la variable de estudio, sigue siendo el objeto de interés o la característica de interés en nuestra población.

3.4.1.1. VARIABLE DE ESTUDIO

A. Java platform enterprise edition

Es una plataforma de desarrollo de aplicaciones para crear sistemas empresariales robustos. J2EE incluye numerosas API y herramientas de Java, incluidos Enterprise JavaBeans, JavaServer Pages y Servlets (Perrone y Chaganti 2003).

3.4.1.2. INDICADORES DE LA VARIABLE

a. Hibernate

Según Rios (2015), 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".

b. Framework basado en http y servlets

"Provee herramientas para la extensión y personalización de aplicaciones web y servicios web REST".

c. Spring MVC

"Spring Web MVC Framework, es un marco robusto, flexible y bien diseñado para el desarrollo rápido de aplicaciones web, utilizando el patrón de diseño MVC" (Hemrajani, 2006).

3.4.1.3. VARIABLE DE ESTUDIO

A. Metodología ágil y formal iconix

Es una metodología de desarrollo de software, orientado a objetos, utilizando UML. Es una metodología de desarrollo práctico impulsado por casos de uso y consiste en flujos de trabajos dinámicos y estáticos. (Virvou y Matsuura, 2012, p.173).

3.4.1.4. INDICADORES DE LA VARIABLE

a. Arquitectura técnica

Según Rosenberg y Stephens, (2007), "Es conocida como arquitectura del sistema y arquitectura del software, generalmente describe el sistema que se intenta construir en términos de estructura".

3.4.2. DEFINICIÓN OPERACIONAL DE LAS VARIABLES DE ESTUDIO

variable de estudio:

X: Java Platform Enterprise Edition

Indicadores:

X1: Hibernate

X2: Framework basado en http y servlets

X3: Spring mvc

variable de estudio:

Y: Metodología Ágil Y Formal Iconix

indicadores:

Y1: Arquitectura Técnica

3.5. TÉCNICAS PARA EL TRATAMIENTO DE DATOS E INFORMACIÓN

3.5.1. TÉCNICAS

Según Arias F. (2006), el Análisis documental, es una técnica que consiste en la separación e interpretación de los contenidos de un documento.

Se consideró la técnica Análisis documental, para documentar las arquitecturas y patrones de diseño e implementación para construir los componentes software que usen modelos arquitectónicos, capas de abstracción, interfaces, metodologías y tecnologías que permiten el acceso a las bases de datos distribuida.

3.5.2. HERRAMIENTAS PARA EL TRATAMIENTO DE DATOS E INFORMACIÓN

Las herramientas tecnológicas a utilizan, fueron seleccionadas, considerando la facilidad y rapidez que nos brinda en el tratamiento de los datos e información obtenidos. Se ha tomado las tecnologías según la tabla.

Tabla 1

Herramientas tecnológicas para el tratamiento de datos e información.

TECNOLOGÍA	FABRICANTE	SERVICIO
S.O WINDOWS 10	Microsoft Corporation	Línea de sistemas operativos producida por Microsoft Corporation.
NETBEANS 8.2	Sun Microsystems	NetBeans es una plataforma de desarrollo de software escrito en Java.
MYSQL	Oracle	Es un sistema de gestión de base de datos relacional que ofrece velocidad al realizar operaciones y facilidad en su instalación y configuración.
DRAW.IO	Diagrams.net	es una herramienta de creación y edición de diagramas libre que permite la integración con diversas plataformas.

Fuente: Elaboración propia.

3.5.4. TÉCNICAS PARA APLICAR ICONIX

Revisando el marco teórico desarrollado en el capítulo II, formularemos el proceso, que sonconsideradas en las fases para desarrollar un software, usando la metodología Iconix, como se observan en las siguientes tablas.

Tabla 2

Análisis de requisitos.

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLE
Identificar requisitos	• Requisitos funcionales y no funcionales • Casos de prueba	• Entrevista • Definir lo que el sistema debe hacer • Escribir al menos un caso de prueba de para cada requisito	Usuario Cliente Analista

Identificar objetos del mundo real y dibujar modelo de dominio	• Modelo de dominio	• Identificar clases clave del negocio. • Identificar sustantivos y depurar. • Identificar objetos en requisitos funcionales y asignar al modelo de dominio. • Utilizar agregación y generalización.	Analista
Realizar prototipo de interfaz gráfica	• Prototipo GUI	• Utilizar historia de eventos del (storyboards) • Utilizar los requisitos funcionales • Diseñar interfaz gráfica básica	Programador Analista
Descubrir casos de uso	• Lista de casos	• Utilizar requisitos funcionales • Entrevistas	Usuario, cliente y analista
Dibujar y empaquetar casos de uso	• Diagrama de casos de uso • Paquete de casos casos de uso	• Identificar roles y responsabilidades de actores • Asociar actores con casos de uso Relacionar casos de uso • Agrupar lógicamente casos de uso	Analista
Asignar requisitos funcionales a los casos de uso	• Relación entre requisitos funcionales y casos de uso	• Asignar requisitos funcionales a los casos de uso	
Escribir el primer borrador de · casos de uso	• Primer borrador de casos de uso	• Utilizar glosario de objetos del modelo de dominio • Utilizar la regla de dos párrafos • Escribir el caso de uso como Primer flujos de evento/respuesta • Escribir el caso de uso con estructura sustantivo-verbosustantivo • Escribir caso de uso en voz activa • Referenciar por su nombre las pantallas	

Fuente: Porras (2011).

Tabla 3

Revisión de requisitos (primer hito).

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLE
Revisar el modelo de dominio	Modelo de dominio	• Identificar al menos 80% de clases clave de dominio del problema	Usuario Cliente Analista Programador
Revisar el prototipo GUI	Prototipo GUI	• Diseñar con precisión la GUI relacionada al caso de uso	
Revisar moelo de casos de uso	Primer borrador de casos de uso	• Eliminar clases fuera del dominio del problema • Cambiar descripción de voz pasiva activa • Describir todos los cursos alternos • Asociar todos los requisitos a los casos de uso • Describir que intenta hacer el usuario para cada caso de uso	

Fuente: Porras (2011).

Tabla 4

Diseo preliminar.

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLE
Rescribir el primer borrador para cada caso de uso	Caso de uso desambiguado	• Rescribir el caso de uso durante el análisis de robustez.	Analista
Identificar el primer corte de objetos que completan escenarios para cada caso de uso	Diagrama de robustez	• Copiar la descripción del caso de uso en el diagrama de robustez. • Usar las clases del modelo del dominio.	

Fuente: Porras (2011).

Tabla 5

Revisión de diseño preliminar (segundo hito)

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLE
Revisar descripción de casos de uso	Casos de uso	• Coincidir la descripción del caso de uso con el diagrama de robustez	Usuario Cliente Analista Programador
Revisar diagrama de robustez	Diagrama de robustez	• Coincidir el diagrama de robustez con descripción del caso de uso • Verificar que el diagrama de robustez tiene todos los cursos alternos	
Revisar modelo de dominio actualizado	Modelo de dominio actualizado	• Coincidir los objetos entidad del diagrama de robustez con el modelo de dominio actualizado	

Fuente: Porras (2011).

Tabla 6

Arquitectura técnica

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLE
Diseñar diagrama de componentes	Diagrama de componentes	• Entrevistas • Características del negocio • Utilizar la arquitectura MVC • Integrar frameworks	Cliente Analista Programador Arquitecto de software
Diseñar diagrama de despliegue	Diagrama de despliegue	• Entrevistas • Características del negocio	

Fuente: Porras (2011).

Tabla 7

Diseño.

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLE
Dividir modelo de dominio actualizado para cada caso de uso	Parte de modelo de dominio actualizado	• Cincidir las clases entidad del diagrama de robustez con parte del modelo de dominio actualizado y dibujarlo	Diseñador
Dibujar un diagrama de secuencia para cada caso de uso	Diagrama de secuencia	• Copiar la descripción del caso de uso • Copiar objetos entidad, interfaz y actores del duagrama de robustez • Verificar que un mensaje del diagrama de secuencia es verbo en el caso de uso • Hacer refactoring al diagrama de secuencia antes de codificar	Programador Diseñador
Actualizar el diagrama de clases de un caso de uso	Diagrama de clase	• Asignar operaciones a las clasez a partir de mensajes del diagrama de secuencia • Establecer multiplicidad en las clases • Depurar las clases, operaciones y atributos del diagrama de clases	
Extraer controladores para pruebas unitarias	Lista de controladores	• Identificar controladores para la lógica del negocio desde un diagrama de robustez	

Fuente: Porras (2011).

Tabla 8

Revisión crítica del diseño (tercer hito).

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLE

Revisar diagrama de secuencia	Diagrama de secuencia	• Verificar que el diagrama de secuencia coincide con la descripción de casos de uso • Verificar que el diagrama de secuencia representa los cursos básico y alterno • Verificar en los mensajes que los atributos y valores de retorno son correctos	Diseñador Programador
Revisar diagrama de clases	Diagrama de clases	• Verificar que el nombre, atributos y operaciones se asinaron correctamente a las clases • Asignar requisitos no funcionales a los casos de uso y clases	
Revisar modelo de dominio actualizado	Modelo de dominio actualizado	• Verificar nombres y atributos del modelo de dominio actualizado	
Revisar lista de pruebas unitarias	Lista de controladores	• Actualizar la lista de controladores	

Fuente: Porras (2011).

Tabla 9

Implementación.

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLE
Implementar la base de datos física	Base de datos física	• Escribir el script usando el modelo de dominio actualizado • Ejecutar el script usando un DBMS	Programador

Implementar código para clases entidad	Código fuente	• Escribir o generar código fuente con una herramienta usando el modelo de dominio actualizado	Programador
Implementar código para las interfaces	Código fuente	• Generar código fuente usando una herramineta	Programador
Crear pruebas unitarias para cada controlador	Prueba unitaria	• Escribir código fuente para una prueba unitaria usando una herramienta	Programador
Implementar código fuente para cada controlador	Código fuente	• Escribir el código fuente siguiendo el flujo normal del diagrama de secuencia usando una herramienta • Actualizar el diagrama de secuencia con la codificación	Programador
Ejecutar pruebas unitarias para cada controlador	Reporte de pruebas unitarias	• Ejecutar el modulo de cada prueba unnitaria • Modificar código fuente si la prueba unitaria muestra resultado incorrecto	Programador
Ejecutar pruebas de aceptación para cada caso de uso	Reporte de pruebas de aceptación	• Utilizar los casos de prueba de aceptación • Ejecutar el modulo de un caso de uso • Modificar código fuente si la prueba de asecptación muestra resultado incorrecto	Programador Usuario Cliente

Fuente: Porras (2011).

CAPITULO IV

ANÁLISIS Y RESULTADOS DE LA INVESTIGACIÓN

4.1. ANÁLISIS DE ADAPTABILIDAD DE LA INFRAESTRUCTURA JAVA PLATFORM ENTERPRISE EDITION A LA ARQUITECTURA TÉCNICA DE LA METODOLGÍA ÁGIL Y FORMAL ICONIX

Se consolidará las mejores prácticas utilizadas entre la arquitectura técnica de la metodología ICONIX y la infraestructura Java Platform Enterprise Edition, basado en componentes y servicios, para adaptar tecnologías utilizadas en proyectos Java EE, con el objetivo de brindar un modelo referencial para el desarrollo ágil de software con escalabilidad, accesibilidad y manejabilidad.

4.1.1. ESTRATEGIA DE ADAPTACIÓN

Con el objetivo de brindar un conjunto de buenas prácticas que permitan el desarrolla ágil de software, se plantea realizar un esquema de adaptabilidad con la infraestructura Java EE y la Arquitectura técnica de la metodología ICONIX, mediante una matriz de cruce, con los componentes que conforman las distintas capas de la arquitectura MVC, con la finalidad de adecuar las tecnologías, Hibernate, Spring mvc y Framework basado en HTTP y Servlets.

Se consolida los distintos componentes del modelo de aplicación de la Java EE, basados en la arquitectura MVC, que según Según Ordax y Ocaña (2012), Los distintos tipos de componentes introducido en el apartado del modelo de aplicación Java EE, encajan perfectamente en este diseño: Modelo: Enterprise JavaBeans, POJOS (Palin Old Java Objects), Vista: JavaServer Pages (JSP), JavaServer Faces (JSF) y Controlador: Java Servelets.

Se consolida los distintos componentes de la arquitectura técnica de la metodología ICONIX, basados en la arquitectura MVC, de acuerdo al diagrama de componentes de un clásico mapeo en la arquitectura MVC.

Tabla 10

Componentes en la infraestructura java platform enterprise edition, en base al patrón MVC.

JAVA PLATFORM ENTERPRISE EDITION	PATRÓN MVC		
	MODELO	**VISTA**	**CONTROLADOR**
Componentes del Modelo de Aplicación	• Enterprise JavaBeans (EJB) • Entity clases (JPA) • POJOS (Palin Old Java Objects).	• JavaServer Pages (JSP) • JavaServer Faces (JSF)	• Java Servelets

Fuente: Elaboración propia (2020).

Tabla 11

Componentes en la arquitectura técnica de la metodología Iconix, en base al patrón MVC.

METODOLOGÍA ICONIX	ARQUITECTURA MVC		
	MODELO	**VISTA**	**CONTROLADOR**
Arquitectura de Paquetes/Componentes	• DAO`s Interfaces • Modelo de dominio (Bean/POJOs) • Implementaciones JDBC DAO	• Servlet • JSP`s	• Controladores • Validadores de entrada

Fuente: Elaboración propia (2020).

Tabla 12

Correspondencia de la infraestructura Java Platform Enterprise Edition y la arquitectura técnica de la metodología ágil y formal ICONIX, basado en el patrón MVC.

ASPECTOS A COMPARAR		JAVA PLATFORM ENTERPRISE EDITION	METODOLOGÍA ICONIX
		Componentes del Modelo de Aplicación	**Arquitectura de Paquetes/Componentes**
PATRÓN MVC	MODELO	• Enterprise JavaBeans (EJB) • Entity clases (JPA) • POJOS (Palin Old Java Objects).	• DAO`s Interfaces • Modelo de dominio (Bean/POJOs) • Implementaciones JDBC DAO
	VISTA	• JavaServer Pages (JSP) • JavaServer Faces (JSF)	• Servlet • JSP`s
	CONTROLADOR	• Java Servelets	• Controladores • Validadores de entrada

Nota: Correspondencia de componentes en cada capa de aplicación
Fuente: Elaboración propia (2020).

Tabla 13

Adaptabilidad de la infraestructura Java Platform Enterprise Edition a la arquitectura técnica de la metodología ágil y formal Iconix.

PATRÓN MVC	JAVA PLATFORM ENTERPRISE EDITION	METODOLOGÍA ICONIX	ADAPTABILIDAD DE TECNOLOGÍA
	Componentes del Modelo De Aplicación	**Arquitectura de Paquetes/Componentes**	
MODELO	• Enterprise JavaBeans (EJB) • Entity clases (JPA) • POJOS (Plain Old Java Objects).	• DAO`s Interfaces • Modelo de dominio (Bean/POJOs) • Implementaciones JDBC DAO	Hibernate
VISTA	• Java Server Pages (JSP) • Java Server Faces (JSF)	• Servlet • JSP`s	Framework basado en HTTP y Servlets
CONTROLADOR	• Java Servlets	• Controladores • Validadores de entrada	Spring mvc

Fuente: Elaboración propia (2020).

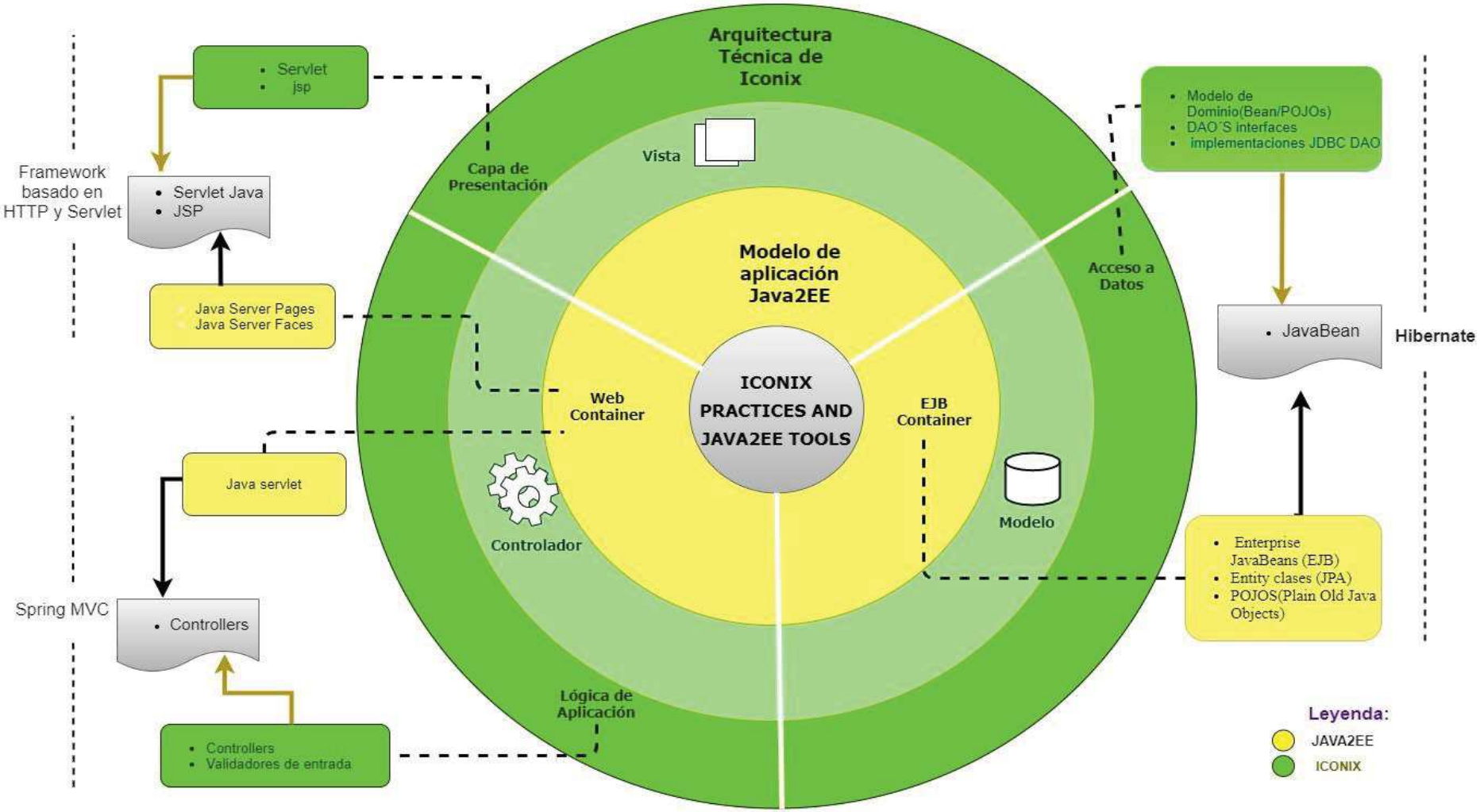


Figura 16. Adaptabilidad de la infraestructura Java Platform Enterprise Edition a la arquitectura técnica de la metodología ágil y formal Iconix.

A. INTERPRETACIÓN EN LA CAPA MODELO

Según Talledo (2015), el modelo, es la capa que trabaja con los datos. Deberá contener mecanismos para acceder a la información y también para actualizar su estado.

Se propone utilizar el mapeo Mapeo Objeto Relacional Hibernate, en la implementación de todo el modelo del software, precisamente, para crear las clases entidades a partir de la base de datos relacional, adpatándose, así como como un indicador para la dimensión modelo de la variable Java2EE.

i. Java Bean

Según Rashmi (2014), un JavaBean o también conocido como Bean, es una clase simple en Java que cumple con ciertas normas con los nombres de sus propiedades y métodos.

Un JavaBean tiene un constructor, tiene declarado todos sus atributos como privados y para cada uno de ellos un método setter y getter. Mediante estos JavaBeans, se desarrolla el modelo de objetos para la aplicación (p. 313).

ii. Hibernate

Es una herramienta de Mapeo objeto-relacional (ORM) para la plataforma Java, que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación (Rios, 2015).

B. INTERPRETACIÓN EN LA CAPA VISTA

Según Talledo (2015), la vista es la encargada de mostrar la información solicitada. Contendrá el código de la aplicación necesaria para mostrar dicha información. En entornos web será, habitualmente, mediante código HTML, CSS y JavaScript, en el caso del navegador que será el encargado de interpretarlo.

Se propone utilizar el HTML5, en la creación de la interfaz, precisamente, de acuerdo a las funciones que cumple las clases servlet Java, HTTP y la JSP, se adapta como indicador el Framework basado en HTTP y Servlets para la dimensión Vista de la variable Java2EE.

i. Servlet Java

Los servlets Java son clases Java diseñadas para responder a solicitudes HTTP en el contexto de una aplicación web.

ii. JSP

Las JSP se puede ver como una extensión de HTML que le ofrece la capacidad de incluir fragmentos de código Java dentro de las páginas HTML. Estos fragmentos de código Java generan contenido dinámico, que está incluido dentro del otro contenido HTML/XML. Una JSP se convierte a un servlet Java y se ejecuta en el servidor. Las sentencias JSP incluidas en la JSP pasan a formar parte del servlet generado a partir de la JSP. El servlet resultante se ejecuta en el servidor.

iii. Framework basado en http y servlet

Provee herramientas para la extensión y personalización de aplicaciones web y servicios web REST.

C. INTERPRETACIÓN EN LA CAPA CONTROLADOR

Rosenber y Stephens (2007), los controladores son el "pegamento" entre la vista y el modelo. En Spring, un objeto Controlador interpreta la entrada del usuario y transforma el resultado en un modelo que se mostrará al usuario en la Vista.

Se propone utilizar el Spring MVC, adaptándose como indicador para la dimensión Controlador de la variable Java2EE.

i. Controllers

Una manera para implementar un controlador, en una palicación web orientado a MVC es con un spring MVC anotado con @controller clase (sí, usando un DispacherServlet) con este se usa el @RequestMethodGET/POST.

ii. Spring MVC

Spring es una implementación del patrón de diseño "front controller". Todas las peticiones HTTP se canalizan a través del front controller. En casi todos los frameworks MVC que siguen este patrón, el front controller no es más que un servlet cuya implementación es propia del framework. En el caso de Spring, la clase DispatcherServlet.

4.2. CASO DE ESTUDIO: REGISTRO DE INCIDENCIA DE BIENES INFORMÁTICOS DE LA UGEL HUAMANGA

4.2.1. ARQUITECTURA J2EE A LA ARQUITECTURA TÉNICA DE ICONIX

Tabla 14

Modelo de la arquictectura J2EE adaptado a la capa modelo de la arquitectica técnica iconix.

MODELO DE LA ARQUICTECTURA J2EE ADAPTADO A LA CAPA MODELO DE LA ARQUITECTICA TÉCNICA ICONIX

```
package com.ugel.entidades;
import javax.persistence.*;

@Entity
@Table(name = "incidencia", schema = "bdincidenciapractica", catalog = "")
public class IncidenciaEntity {
    private int idIncidencia;
    private String descripcion;
    private String fechaincidencia;
    private String denominacion;
    private int bieninformaticoIdBienInformatico;
    private int tecnicoIdTecnico;

    @Id
    @Column(name = "idIncidencia")
    public int getIdIncidencia() {
        return idIncidencia;
    }

    public void setIdIncidencia(int idIncidencia) {
        this.idIncidencia = idIncidencia;
    }

    @Basic
    @Column(name = "descripcion")
    public String getDescripcion() {
```

```
    return descripcion;
}

public void setDescripcion(String descripcion) {
    this.descripcion = descripcion;
}

@Basic
@Column(name = "fechaincidencia")
public String getFechaincidencia() {
    return fechaincidencia;
}

public void setFechaincidencia(String fechaincidencia) {
    this.fechaincidencia = fechaincidencia;
}

@Basic
@Column(name = "denominacion")
public String getDenominacion() {
    return denominacion;
}

public void setDenominacion(String denominacion) {
    this.denominacion = denominacion;
}

@Basic
@Column(name = "bieninformatico_id_BienInformatico")
public int getBieninformaticoIdBienInformatico() {
    return bieninformaticoIdBienInformatico;
}

public void setBieninformaticoIdBienInformatico(int bieninformaticoIdBienInformatico) {
    this.bieninformaticoIdBienInformatico = bieninformaticoIdBienInformatico;
}
```

```
@Basic
@Column(name = "tecnico_id_Tecnico")
public int getTecnicoIdTecnico() {
    return tecnicoIdTecnico;
}

public void setTecnicoIdTecnico(int tecnicoIdTecnico) {
    this.tecnicoIdTecnico = tecnicoIdTecnico;
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;

    IncidenciaEntity that = (IncidenciaEntity) o;

    if (idIncidencia != that.idIncidencia) return false;
    if (bieninformaticoIdBienInformatico != that.bieninformaticoIdBienInformatico)
    return false;
    if (tecnicoIdTecnico != that.tecnicoIdTecnico) return false;
    if (descripcion != null ? !descripcion.equals(that.descripcion) : that.descripcion != null)
    return false;
    if (fechaincidencia != null ? !fechaincidencia.equals(that.fechaincidencia) :
    that.fechaincidencia != null)
     return false;
    if (denominacion != null ? !denominacion.equals(that.denominacion) :
    that.denominacion != null)
    return false;

    return true;
}

@Override
public int hashCode() {
```

```
        int result = idIncidencia;
        result = 31 * result + (descripcion != null ? descripcion.hashCode() : 0);
        result = 31 * result + (fechaincidencia != null ? fechaincidencia.hashCode() : 0);
        result = 31 * result + (denominacion != null ? denominacion.hashCode() : 0);
        result = 31 * result + bieninformaticoIdBienInformatico;
        result = 31 * result + tecnicoIdTecnico;
        return result;
    }
}
```

Nota: Generación de la clase Incidencia a partir de base de datos relacional. Hibernate.
Fuente: Elaboración propia (2020).

Tabla 15

Vista de la arquictectura J2EE adaptado a la capa vista de la arquitectica tecnica iconix.

VISTA DE LA ARQUICTECTURA J2EE ADAPTADO A LA CAPA VISTA DE LA ARQUITECTICA TECNICA ICONIX

```
<body>
<nav class="navbar navbar-inverse sidebar" role="navigation">
    <div class="container-fluid">
        <div class="navbar-header">
            <button type="button" class="navbar-toggle" data-toggle="collapse"
                    data-target="#bs-sidebar-navbar-collapse-1">
                <span class="sr-only">UGEL-HUAMANGA</span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            <img class="candado" src="imgs/uge.jpg" width="100" height="100"/>
             <a class="navbar-brand" th:href="@{/home}" style="color:blue">
            <b>UGEL-HUAMANGA</b></a>
        </div>

        <div class="collapse navbar-collapse" id="bs-sidebar-navbar-collapse-1">
            <ul class="nav navbar-nav">
                <li><a th:href="@{/home}">Home<span style="font-size:16px;"
```

```
            class="pull-right hidden-xs showopacity glyphicon glyphicon-ome"></span></a>
        </li>
        <li class="dropdown">
          <a      href="#"      class="dropdown-toggle"      data-toggle="dropdown">BIEN
           INFORMATICO <span class="caret"></span><span style="font-size:16px;"
          class="pull-right hidden-xs showopacity glyphicon glyphicon-cog"></span></a>
           <ul class="dropdown-menu forAnimate" role="menu">
             <li><a th:href="@{/listaEquipo}">mantener Bien informatico</a></li>
              <li><a th: href="@{/addEquipoInformatico}">Agregar Bien
             informatico</a></li>
           </ul>
        </li>
        <li class="dropdown active">
       <a   href="#"   class="dropdown-toggle"   data-toggle="dropdown">INCIDENCIA
        <span
         class="caret"></span><span style="font-size:16px;"
          class="pull-right hidden-xs showopacity glyphicon glyphicon-cog"></span></a>
           <ul class="dropdown-menu forAnimate" role="menu">
             <li><a th:href="@{/listaIncidencia}">mantener Incidencia</a></li>
             <li><a th:href="@{/addIncidencia}">Agregar Incidencia</a></li>
           </ul>
        </li>

        <li class="dropdown">
<a href="#" class="dropdown-toggle" data-toggle="dropdown">OFICINA <span
 class="caret"></span><span
    style="font-size:16px;"
       class="pull-right hidden-xs showopacity glyphicon glyphicon-cog"></span></a>
           <ul class="dropdown-menu forAnimate" role="menu">
             <li><a th:href="@{/listaOficina}">mantener oficina</a></li>
             <li><a th:href="@{/addOffice}">Agregar oficina</a></li>
           </ul>
        </li>
          <li><a th:href="@{/listaEstadoBI}">ESTADO BIEN INFORMATICO<span
  style="font-size:16px;"
 class="pull-right hidden-xs showopacity glyphicon glyphicon-user"></span></a></li>
```

```
    <li><a th:href="@{/listaTipoBI}">TIPO BIEN INFORMATICO<span style="font-
    size:16px;"
        class="pull-right hidden-xs showopacity glyphicon glyphicon-home"></span></a>
      </li>
       <li><a th:href="@{/listaEncargado}">ENCARGADO<span style="Font size:16px;"
          class="pull-right hidden-xs showopacity glyphicon glyphicon-user"></span></a>
        </li>
        <li><a th:href="@{/listaTecnico}">TECNICO<span style="font-size:16px;"
      class="pull-right hidden-xs showopacity glyphicon glyphiconenvelope"></span></a>
        </li>
        <li><a th:href="@{/reporte}">REPORTES<span style="font-size:16px;"
      class="pull-right hidden-xs showopacity glyphicon glyphicon-envelope"></span></a>
        </li>
        <li>
          <form th:action="@{/logout}" method="post">
            <button class="btn  btn-primary btn-block" type="submit">SALIR</button>
          </form>
        </li>
      </ul>
    </div>
  </div>
</nav>
<div class="main ">

  <div class="container">
    <h2>Incidencias</h2>
    <div class="row">
      <div class="col-md-6">

        <div id="custom-search-input">
          <div class="input-group col-md-12">
            <input id="searchTerm" type="text" onkeyup="doSearch()" class="form-
            control input-lg" placeholder="Buscar" />
            <span class="input-group-btn">
            <button class="btn btn-info btn-lg" type="button">
              <i class="glyphicon glyphicon-search" style="color: #328cd8"></i>
```

```
                </button>
              </span>
              </div>
            </div>
            <br/>
          </div>
        </div>
        <table class="table" id="regTable">
          <tr>
            <th>Descripcion</th>
            <th>Fecha Incidencia</th>
            <th>Denominacion</th>
            <th></th>
            <th></th>
          </tr>
          <tr th:each="incidencia : ${incidencia}">
            <td th:text="${incidencia.descripcion}"></td>
            <td th:text="${incidencia.fechaincidencia}"></td>
            <td th:text="${incidencia.denominacion}"></td>

       <td > <a th:href="@{/editIncidencia/{id}(id=${incidencia.idIncidencia})}"><span
         class="glyphicon glyphicon-pencil"></span> Editar</a>  </td>
          <td><ath:href="@{/deleteIncidencia/{id}(id=${incidencia.idIncidencia})}"
          style="color:red"><span
       class="glyphicon glyphicon-trash" style="color: red"></span> Eliminar</a>  </td>
      <!--<td > <a th:href="@{/delete/{id}(id=${oficina.idOficina})}">Eliminar</a>  </td>-->
      <!--<td > <a th:href="@{/edit/{id}(id=${oficina.idOficina})}">Editar</a>  </td>-->
            </tr>
        </table>
    </div>
</div>
</body>
</html>
```

Nota: Vista: HTML5. Registro de incidencia de bienes informáticos.
Fuente: Elaboración propia (2020).

Tabla 16

Controlador de la arquictectura J2EE adaptado a la capa controlador de la arquitectica tecnica iconix

CONTROLADOR DE LA ARQUICTECTURA J2EE ADAPTADO A LA CAPA CONTROLADOR DE LA ARQUITECTICA TECNICA ICONIX

```
@Controller
public class IncidenciaController {

   @Autowired
   @Qualifier("TecnicoServiceImpl")
   private TecnicoService TecnicoServiceImpl;

   @Autowired
   @Qualifier("EquipoInformaticoServiceImpl")
   private EquipoInformaticoService EquipoInformaticoService;
   @Autowired
   @Qualifier("InicidenciaService")
   private IncidenciaService InicidenciaService;



   @GetMapping("/listaIncidencia")
   public ModelAndView listaEquipo() {
      ModelAndView mav = new ModelAndView("Incidencia");
      //mav.addObject("oficinaIntro2",new OficinaEntity());
      mav.addObject("incidencia",InicidenciaService.listAllIncidencia());
      return mav;
   }

   @PostMapping("/saveIncidencia")
   public ModelAndView save(@Valid IncidenciaEntity bien, BindingResult result) {
      if(result.hasErrors()) {
         return addIncidencia(bien);
      }
```

```
        InicidenciaService.save(bien);
        return listaEquipo();
    }

    @GetMapping("/addIncidencia")
    public ModelAndView addIncidencia(@ModelAttribute("oficina")IncidenciaEntity
     bien ){
        ModelAndView mv = new ModelAndView("IncidenciaAdd");
        mv.addObject("inicidenciaIntro", bien);
        mv.addObject("equipo", EquipoInformaticoService.listAllEquipo());
        mv.addObject("tecnico",TecnicoServiceImpl.listAllTecnico());
        return mv;
    }
    @GetMapping("/editIncidencia/{id}")
    public ModelAndView edit(@PathVariable("id") int id) {

        return addIncidencia(InicidenciaService.findOne(id));
    }

    @GetMapping("/deleteIncidencia/{id}")
    public ModelAndView delete(@PathVariable("id") int id) {
        InicidenciaService.removeBienInformatico(id);
        return listaEquipo();
    }
}
```

Nota: Controller de la clase incidencia.
Fuente: Elaboración propia (2020).

4.2.2. IMPLEMENTACIÓN DEL SOFTWARE BAJO EL MARCO ADAPTADO DE J2EE Y ICONIX

Según las fases para desarrollar el software, utilizando la metodología Iconix, descrito en el capítulo II, se obtienen los artefactos; análisis de requisitos, revisión de requisitos, diseño preliminar, revisión de diseño preliminar, diseño detallado, revisión crítica de diseño e implementación.

A continuación, mostramos los resultados de los artefactos de acuerdo a la metodología Iconix.

4.2.2.1. ANÁLISIS DE REQUISITOS

A. REQUISITOS FUNCIONALES

Tabla 17

Requisitos funcionales.

N°	REQUISITOS FUNCIONALES
REQF 001	El administrador debe ser capaz de crear una cuenta para un usuario (jefe de informática, operador PAD y asistente informático), previa solicitud de acceso.
REQF 002	El sistema debe ser accedido mediante una cuenta de usuario y contraseña.
REQF 003	El sistema debe permitir registrar, modificar y eliminar un bien informático.
REQF 004	El sistema debe ser capaz de mostrar la matriz de todos los equipos informáticos de la institución.
REQF 005	El sistema debe ser capaz de realizar la búsqueda de un bien informático de acuerdo al código patrimonial.
REQF 006	El sistema debe permitir visualizar los datos de un bien informático.
REQF 007	El sistema debe permitir registrar, modificar y eliminar una incidencia de un bien informático.
REQF 008	El sistema ser capaz de mostrar el registro histórico de incidencias de los bienes informáticos.

REQF 009	El sistema debe permitir registrar una oficina.
REQF 010	El sistema debe permitir buscar, modificar y eliminar una oficina.
REQF 011	El sistema debe ser capaz de emitir un reporte de bienes informáticos por oficina o categoría.
REQF 012	El sistema debe ser capaz de emitir un reporte histórico de incidencias de un bien informático.

Fuente: Elaboración propia (2020).

B. REQUISITOS NO FUNCIONALES

Tabla 18

Requisitos no funcionales.

N°	REQUISITOS NO FUNCIONALES
REQNF 001	El sistema debe poseer una interfaz amigable y fácil para la utilización del usuario.
REQNF 002	El sistema debe ser capaz de ejecutarse en cualquier sistema operativo garantizando su portabilidad en distintos navegadores web.
REQNF 003	El sistema debe ser capaz de rechazar el logueo incorrecto del usuario.

Fuente: Elaboración propia (2020).

4.2.2.2. GLOSARIO DE TÉRMINOS

Administrador	Bien informático	Institución	Informática
Jefe de informática	Incidencia	Reporte	Solicitud de acceso
Operador PAD	código patrimonial	Oficina	Cuenta de usuario
Asistente	informático		

4.2.2.3. GLOSARIO DE TÉRMINOS

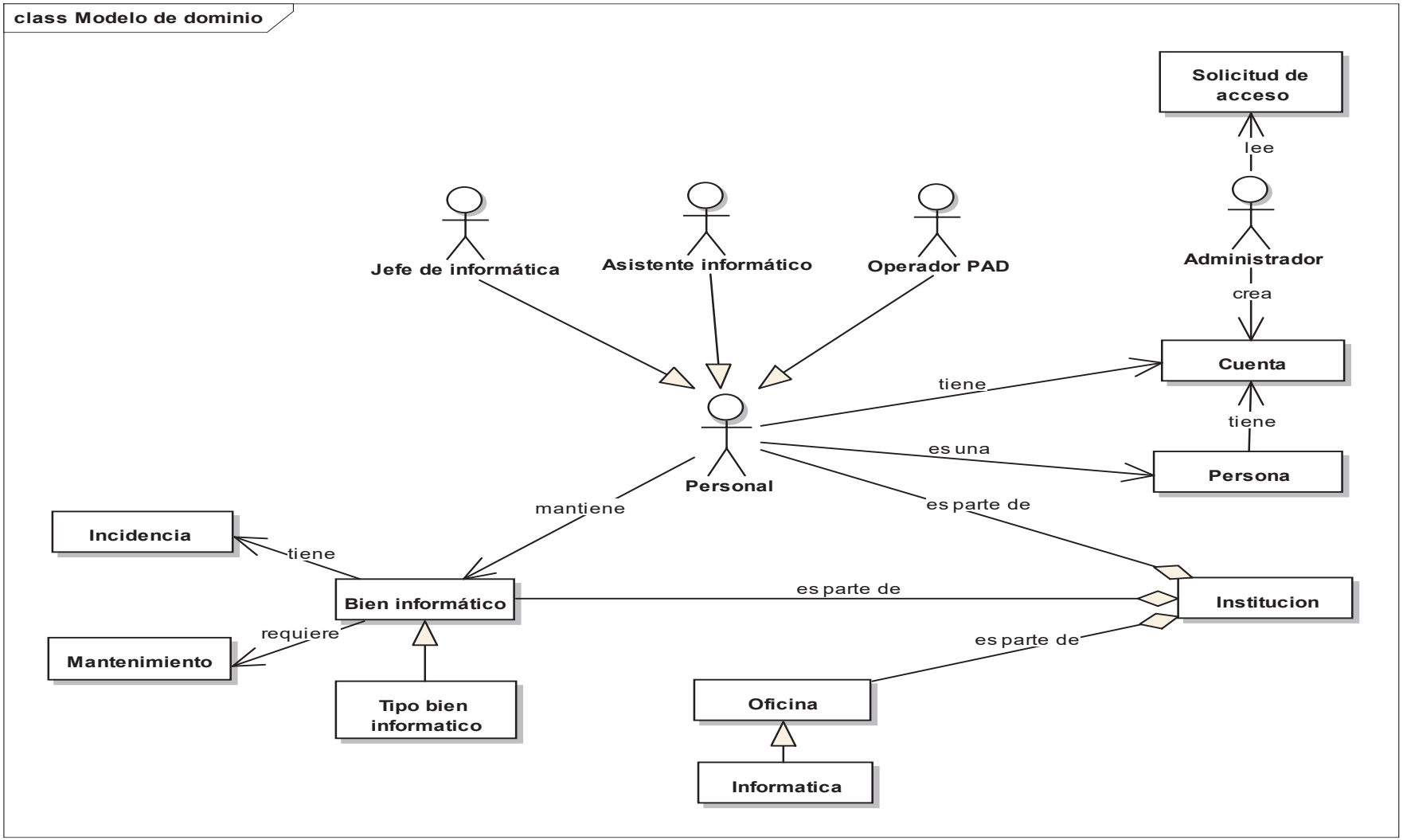


Figura 17. Modelo de dominio inicial.

4.2.2.4. RELACION ENTRE REQUISITOS FUNCIONALES Y CASOS DE USO

Tabla 19

Relación entre requisitos funcionales y casos de uso.

REQUISITOS FUNCIONALES	CASO DE USO
1. El administrador debe ser capaz de crear una cuenta para un usuario (jefe de informática, operador PAD y asistente informático), previa solicitud de acceso.	1. Crear cuenta de usuario
2. El sistema debe ser accedido mediante una cuenta de usuario y contraseña.	2. Autenticar usuario
3. El sistema debe permitir registrar, modificar y eliminar un bien informático. **4.** El sistema debe ser capaz de mostrar la matriz de todos los equipos informáticos de la institución. **5.** El sistema debe ser capaz de realizar la búsqueda de un bien informático de acuerdo al código patrimonial. **6.** El sistema debe permitir visualizar los datos de un bien informático.	3. Registrar bien informático 4. Mantener bien informático
7. El sistema debe permitir registrar, modificar y eliminar una incidencia de un bien informático. **8.** El sistema ser capaz de mostrar el registro histórico de incidencias de los bienes informáticos.	5. Registrar incidencia 6. Mantener incidencia

9. El sistema debe permitir registrar una oficina. **10.** El sistema debe permitir registrar, modificar y eliminar una oficina.	7. Registrar oficina 8. Mantener oficina
11. El sistema debe ser capaz de emitir un reporte del registro histórico de incidencias de un bien informático.	9. Emitir reporte incidencias
12. El sistema debe ser capaz de emitir un reporte de bienes informáticos por oficina o tipo de bien.	10. Emitir reporte bienes informáticos por oficina 11. Emitir reporte bienes informáticos por tipo de bien

Fuente: Elaboración propia (2020).

4.2.2.5. LISTA DE CASOS DE USO

Tabla 20

Lista de casos de uso.

Nº	CASOS DE USO
CU1	Crear cuenta de usuario
CU2	Autenticar usuario
CU3	Registrar bien informático
CU4	Mantener bien informático
CU5	Registrar incidencia
CU6	Mantener incidencia
CU7	Registrar oficina
CU8	Mantener oficina
CU9	Emitir reporte incidencias

CU10	Emitir reporte bienes informáticos por oficina
CU11	Emitir reporte bienes informáticos por tipo de bien

Fuente: Elaboración propia (2020).

4.2.2.6. CASOS DE USO POR PAQUETES

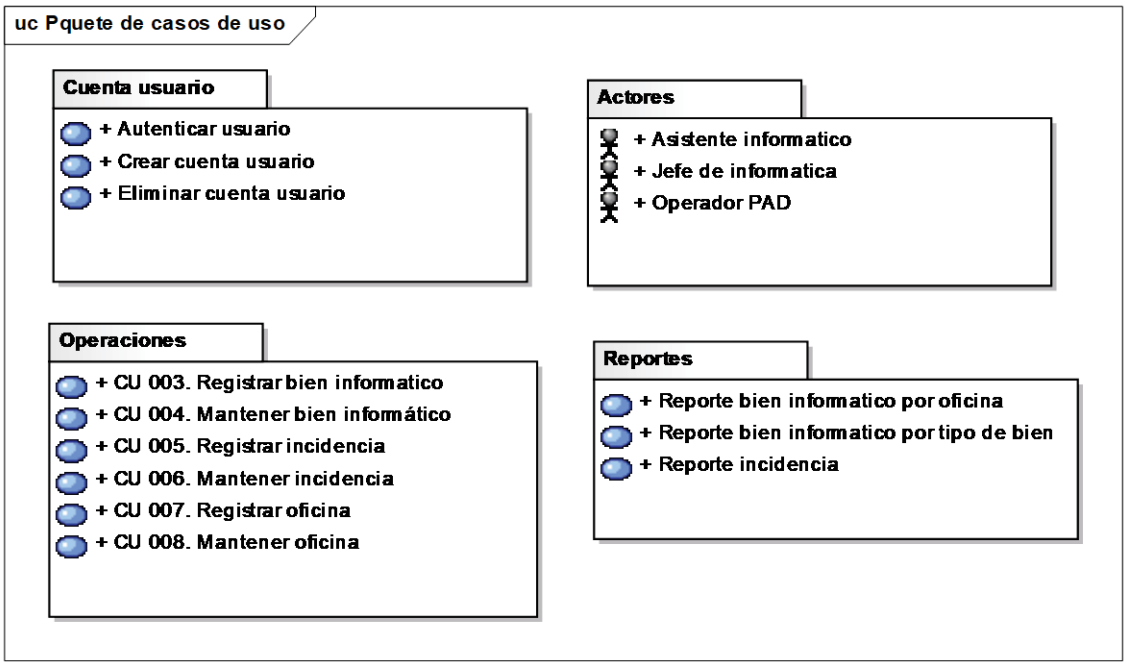


Figura 18. Empaquetado general de casos de uso.

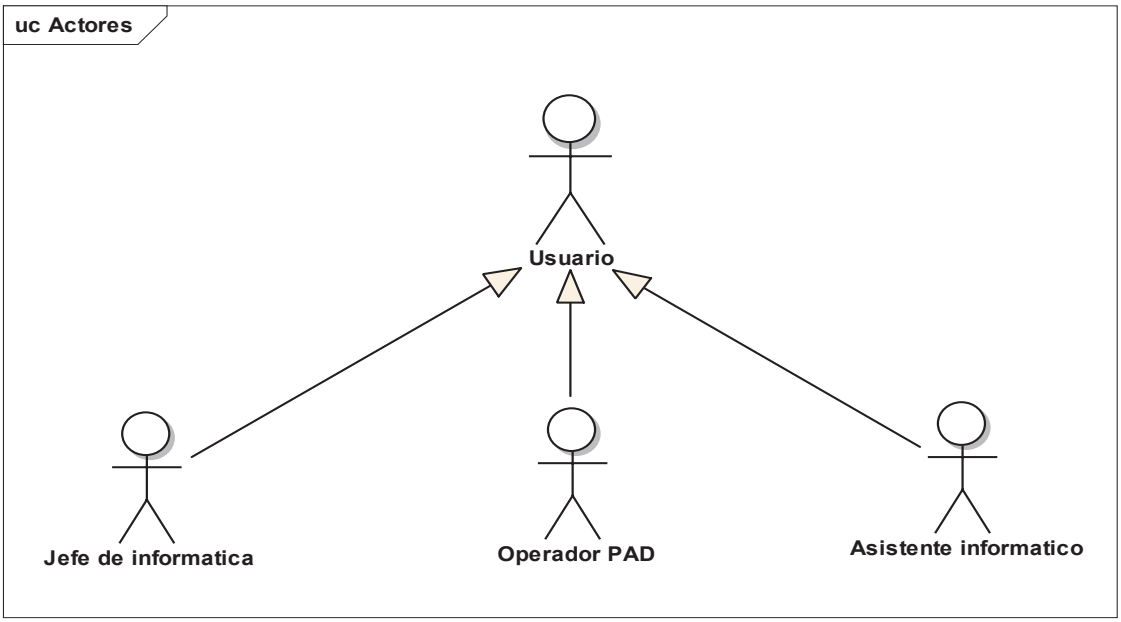


Figura 19. Paquete Actores.

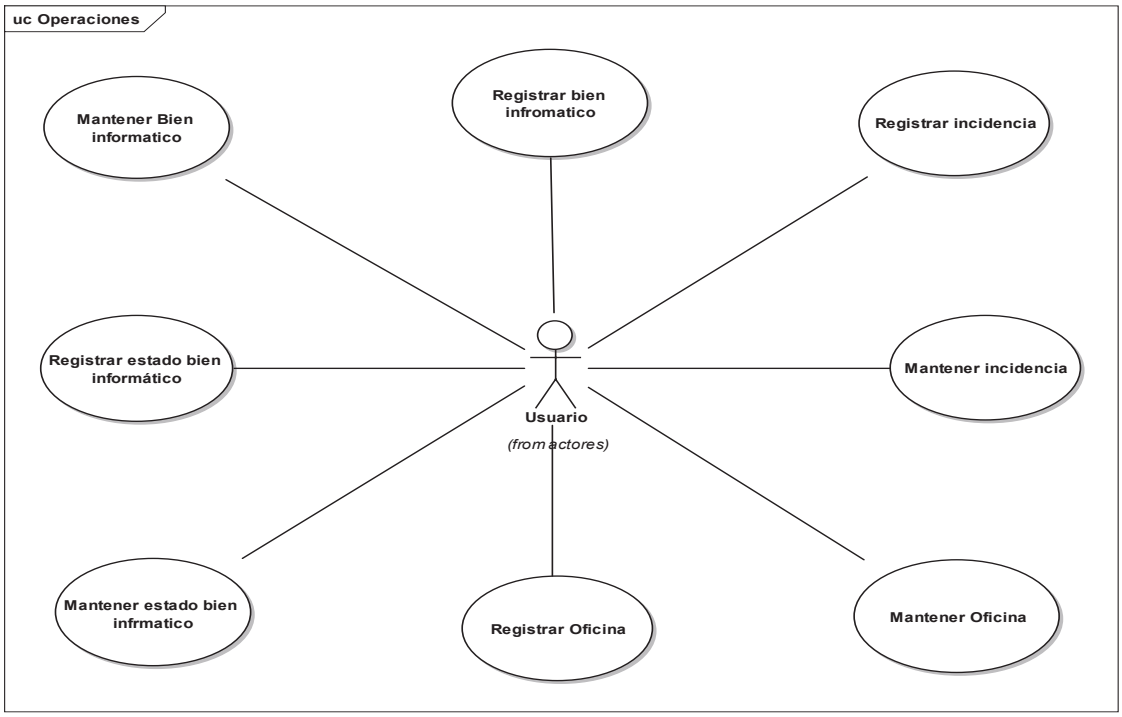


Figura 20. Diagrama de casos de uso.

4.2.2.7. PROTOTIPO DE LA INTERFAZ GRÁFICA

Figura 21. Prototipo acceso al sistema.

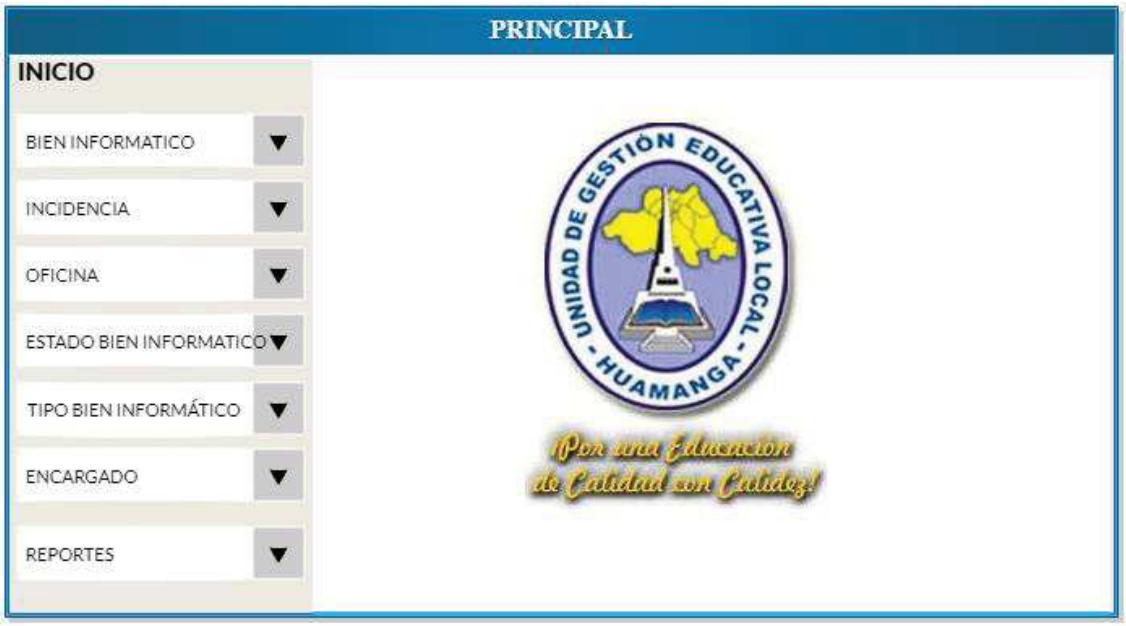


Figura 22. Prototipo menú principal.

Figura 23. Prototipo registrar bien informático.

Figura 24. Prototipo Mantener Bien Informático.

Figura 25. Prototipo registrar incidencia.

Figura 26. Prototipo mantener incidencia.

Figura 27. Prototipo registrar oficina.

Figura 28. Prototipo mantener oficina.

Figura 29. Prototipo registrar estado bien informático.

Figura 30. Prototipo mantener estado bien informático.

4.2.2.8 ETAPA 1: REVISIÓN DE REQUISITOS

A. REQUISITOS FUNCIONALES REVISADOS

Tabla 21

Requisitos Funcionales revisados.

N°	REQUISITOS FUNCIONALES
REQF 001	El **administrador** debe ser capaz de crear una cuenta para un **usuario (jefe de informática, operador PAD y asistente informático)**, previa **solicitud de acceso**.
REQF 002	El **sistema** debe ser accedido mediante una **cuenta de usuario** y contraseña.
REQF 003	El **sistema** debe permitir registrar un **bien informático**.
REQF 004	El **sistema** debe permitir buscar, visualizar, modificar y dar de baja un **bien informático.**
REQF 005	El **sistema** debe permitir registrar una **incidencia** de un **bien informático**.
REQF 006	El **sistema** debe permitir buscar, visualizar, modificar y eliminar una **incidencia** de un **bien informático**.
REQF 007	El **sistema** debe permitir registrar una **oficina**.
REQF 008	El **sistema** debe permitir buscar, visualizar, modificar y eliminar una **oficina**.

REQF 009	El **sistema** debe ser capaz de emitir un **reporte** de bienes informáticos por **oficina** o **tipo de bien**.
REQF 010	El **sistema** debe ser capaz de emitir un **reporte** del **registro histórico** de incidencias de un **bien informático**.

Fuente: Elaboración propia (2020).

B. REQUISITOS NO FUNCIONALES REVISADOS

Tabla 22

Requisitos Funcionales revisados.

N°	REQUISITOS NO FUNCIONALES
REQNF 001	El sistema debe poseer una interfaz amigable y fácil para la utilización del usuario.
REQNF 002	El sistema debe ser capaz de ejecutarse en cualquier sistema operativo garantizando su portabilidad en distintos navegadores web.
REQNF 003	El sistema debe ser capaz de rechazar el logueo incorrecto del usuario.

Fuente: Elaboración propia (2020).

4.2.2.9. REQUISITOS NO FUNCIONALES REVISADOS

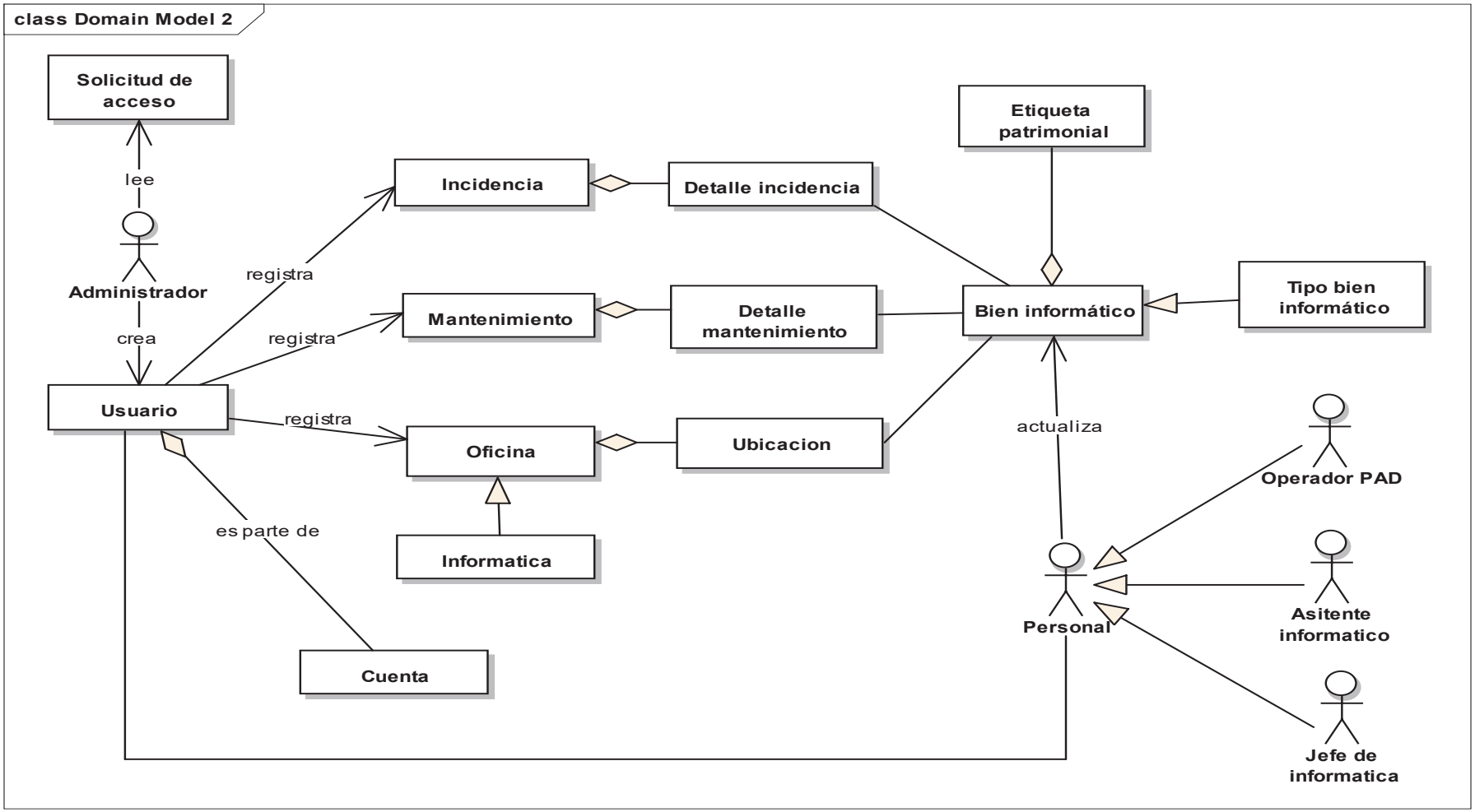


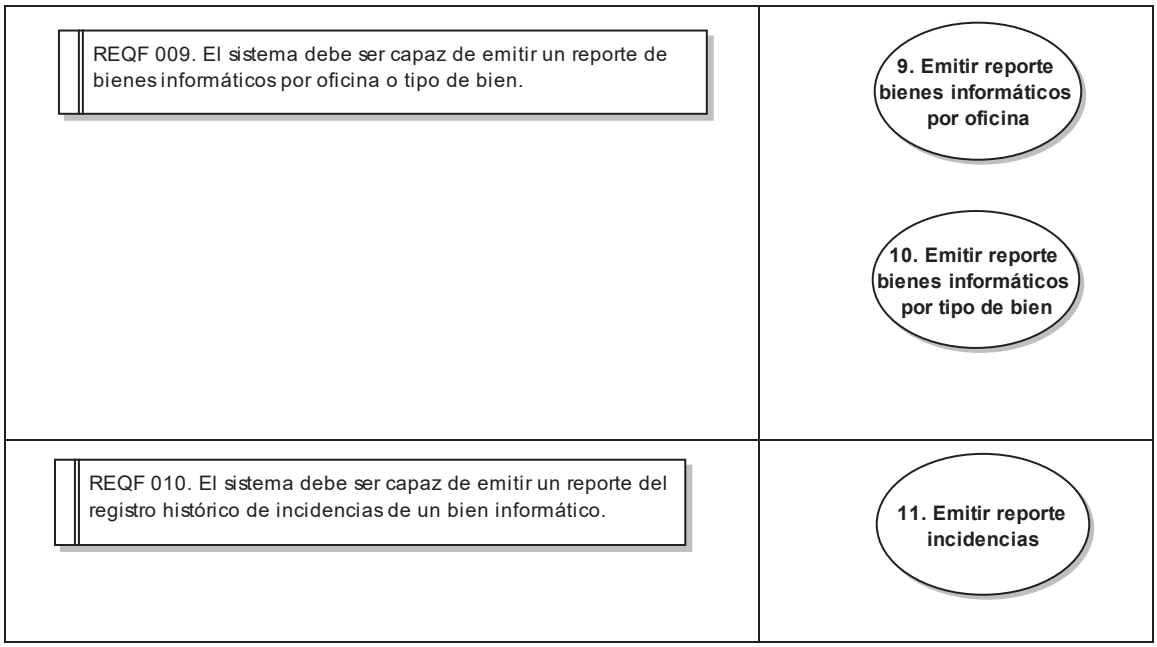
Figura 31. Modelo de dominio general revisado

4.2.2.10. REVISIÓN DE CASOS DE USO

Tabla 23

Revisión de casos de uso identificados según requisitos.

REQUISITOS FUNCIONALES	CASO DE USO
REQF 001. El administrador debe ser capaz de crear una cuenta para un usuario (jefe de informática, operador PAD y asistente informático), previa solicitud de acceso.informático), previa solicitud de acceso.	**1. Crear cuenta de usuario**
REQF 002. El sistema debe ser accedido mediante una cuenta de usuario y contraseña.	**2. Autenticar usuario**
REQF 003. El sistema debe permitir registrar un bien informático.	**3. Registrar bien informático**
REQF 004. El sistema debe permitir buscar, visualizar, modificar y eliminar un bien informático.patrimonial.	**4. Mantener bien informático**
REQF 005. El sistema debe permitir registrar una incidencia de un bien informático.	**5. Registrar incidencia**
REQF 006. El sistema debe permitir buscar, visualizar, modificar y eliminar una incidencia de un bien informático.	**6. Mantener incidencia**
REQF 007. El sistema debe permitir registrar una oficina.	**7. Registrar oficina**
REQF 008. El sistema debe permitir buscar, visualizar, modificar y eliminar una oficina.	**8. Mantener oficina**



Fuente: Elaboración propia (2020).

4.2.2.11. BORRADOR DE CASOS DE USO

Tabla 24

Revisión del caso de uso Registrar bien informático.

Caso de uso CU 03	Registrar bien informático
Actores	Jefe de informática, Operador PAD, Asistente Informático
Descripción	El jefe de informática, Operador PAD, Asistente Informático, registra un equipo informático
Curso básico	1. El usuario (jefe de informática, operador PAD, asistente informático), hace clic en el menú "BIEN INFORMÁTICO", el sistema desglosa las opciones: Registrar Bien Informático, y Mantener Bien Informático. 2. El usuario hace clic en la opción "Registrar Bien Informático", y el sistema le muestra la Pagina Registrar Bien Informático. 3. El usuario ingresa el código institucional, marca, modelo, serie, descripción, fecha ingreso. 4. El usuario selecciona la categoría, el sistema muestra el código de la categoría seleccionada. 5. El usuario selecciona la oficina, el sistema muestra la oficina seleccionada. 6. El usuario selecciona el estado, el sistema muestra estado seleccionado. 7. El usuario hace clic en el botón guardar, el sistema registra la información ingresada y muestra un mensaje "Registro satisfactorio".

Curso Alterno	**Datos aún sin ingresar:** El usuario hace clic en el botón guardar, el sistema muestra un mensaje de error "falta ingresar datos".

Fuente: Elaboración propia (2020).

Tabla 25

Revisión del caso de uso Mantener bien informático.

Caso de uso CU 04	Mantener bien informático
Actor	Jefe de informática, Operador PAD, Asistente Informático
Descripción	El jefe de informática, Operador PAD, Asistente Informático, puede buscar, visualizar, editar y dar de baja un bien informático.
Curso básico	1. El usuario (jefe de informática, asistente informático, Operador PAD), hace clic en el menú "BIEN INFORMÁTICO", el sistema desglosa las opciones**:** Registrar Bien Informático y Mantener Bien Informático. 2. El usuario hace clic en la opción "Mantener Bien Informático", y el sistema le muestra la Pagina Mantener Bien informático. 3. El usuario ingresa el código patrimonial del bien informático en la casilla en blanco, hace clic en el botón buscar y el sistema le muestra los detalles principales del bien informático. 4. Si el usuario desea actualizar, hace clic en el botón Editar y el sistema le muestra la ventana de edición cargando los datos del bien informático, luego hace clic en los campos (código patrimonial, marca, etc.) redefine los datos que desea actualizar y hace clic en el botón Guardar y el sistema actualiza y muestra un mensaje de "Actualización correcta". 5. Si el usuario desea dar de baja, hace clic en el botón Eliminar y el sistema elimina el bien informático y muestra el mensaje "Equipo informático dado de baja"
Curso Alterno	**Datos incorrectos:** El sistema muestra un mensaje de error cuando en la búsqueda se ingresa el código patrimonial incorrecto (Paso 3).

Fuente: Elaboración propia (2020).

Tabla 26

Revisión del caso de uso Registrar incidencia.

Caso de uso CU 05	Registrar incidencia
Actor	Jefe de informática, Operador PAD, Asistente Informático
Descripción	El Operador PAD, Asistente Informático registra una incidencia de un bien informático
Curso básico	1. El usuario (jefe de informática, operador PAD, asistente informático,), hace clic en el menú "INCIDENCIA", el sistema desglosa las opciones: Registrar Incidencia y Mantener Incidencia. 2. El usuario hace clic en la opción "Registrar Incidencia", y el sistema le muestra la Pagina Registrar Incidencia. 3. El usuario ingresa la descripción, denominación y mantenimiento de la incidencia. 4. El usuario ingresa el bien informático. 5. El usuario ingresa el técnico encargado del mantenimiento de la incidencia. 6. El usuario selecciona la fecha de incidencia, el sistema muestra fecha seleccionada. 7. El usuario hace clic en el botón guardar, el sistema registra la información ingresada y muestra un mensaje "Registro satisfactorio".
Curso Alterno	**Datos aún sin ingresar:** El usuario hace clic en el botón guardar, el sistema muestra un mensaje de error "falta ingresar datos".

Fuente: Elaboración propia (2020).

Tabla 27

Revisión del caso de uso Mantener incidencia.

Caso de uso CU 06	Mantener incidencia
Actor	Jefe de informática, Operador PAD, Asistente Informático
Descripción	El Operador PAD, Asistente Informático actualiza los datos registrados de un bien informático.
Curso básico	1. El usuario (jefe de informática, asistente informático, operador PAD), hace clic en el menú "INCIDENCIA", el sistema desglosa las opciones: Registrar Incidencia y Mantener Incidencia. 2. El usuario hace clic en la opción "Mantener Incidencia"

	y el sistema le muestra la Pagina Mantener Incidencia. 3. El usuario ingresa el código de la incidencia en la casilla en blanco y hace clic en el botón buscar, el sistema le muestra los detalles principales de la incidencia. 4. Si el usuario desea actualizar hace clic en el botón Editar y el sistema le muestra la ventana de edición cargando los datos de la incidencia luego hace clic en los campos que quiere actualizar, redefine los datos y hace clic en el botón Guardar y el sistema actualiza y muestra un mensaje de "Actualización correcta". 5. Si el usuario desea eliminar hace clic en el botón Eliminar y el sistema elimina la incidencia y muestra el mensaje "Incidencia eliminada"
Curso Alterno	**Datos incorrectos:** El sistema muestra un mensaje de error cuando en la búsqueda se ingresa el código de incidencia incorrecto (Paso 3).

Fuente: Elaboración propia (2020).

Tabla 28

Revisión del caso de uso Reporte bien informático por oficina.

Caso de uso CU 09	Reporte bien informático por oficina
Actor	Jefe de informática, Operador PAD, Asistente Informático
Descripción	El Operador PAD, Asistente Informático emite reporte de los bienes informáticos por oficina.
Curso básico	1. El usuario (jefe de informática, asistente informático, operador PAD), hace clic en el menú "REPORTES", el sistema desglosa las opciones: Reporte bienes informáticos por oficina, Reporte bienes informáticos por tipo de bien y Reporte incidencias. 2. El usuario hace clic en la opción "Reporte bienes informáticos por oficina", y el sistema le muestra en formato PDF el Reporte de Equipos Informáticos por cada oficina.
Curso Alterno	**Error de sistema al actualizar datos:** El sistema muestra un mensaje de error generado por el sistema y no muestra el reporte.

Fuente: Elaboración propia (2020).

Tabla 29

Reporte bien informático por categoría.

Caso de uso CU 10	Reporte bien informático por categoría
Actor	Jefe de informática, Operador PAD, Asistente Informático
Descripción	El Operador PAD, Asistente Informático emite reporte de los bienes informáticos por oficina.
Curso básico	1. El usuario (jefe de informática, asistente informático, operador PAD), hace clic en el menú "REPORTES", el sistema desglosa las opciones: Reporte bienes informáticos por oficina, Reporte bienes informáticos por tipo de bien y Reporte incidencias. 2. El usuario hace clic en la opción "Reporte bienes informáticos por categoria", y el sistema le muestra en formato PDF el Reporte de Equipos Informáticos por tipo de bien.
Curso Alterno	**Error de sistema al actualizar datos:** El sistema muestra un mensaje de error generado por el sistema y no muestra el reporte.

Fuente: Elaboración propia (2020).

Tabla 30

Reporte incidencia ´por bien informático.

Caso de uso CU 11	Reporte de incidencia
Actor	Jefe de informática, Operador PAD, Asistente Informático
Descripción	El Operador PAD, Asistente Informático emite reporte de incidencias de un bien informático.
Curso básico	1. El usuario (jefe de informática, asistente informático, operador PAD), hace clic en el menú "REPORTES", el sistema desglosa las opciones: Reporte bienes informáticos por oficina, Reporte bienes informáticos por tipo de bien y Reporte incidencias. 2. El usuario hace clic en la opción "Reporte Incidencias", y el sistema le muestra en formato PDF el Reporte de incidencias de los cada uno de los Bienes Informáticos.
Curso Alterno	**Error Registro no encontrado:** El usuario ingresa el código patrimonial y hace clic en el botón VER

	REPORTE, el sistema muestra un mensaje "No existe código patrimonial". **Error de sistema:** El sistema muestra un mensaje de error "500, home" generado por el sistema y no muestra el reporte.

Fuente: Elaboración propia (2020).

4.2.2.12. ETAPA 2: ANÁLISIS Y DISEÑO PRELIMINAR

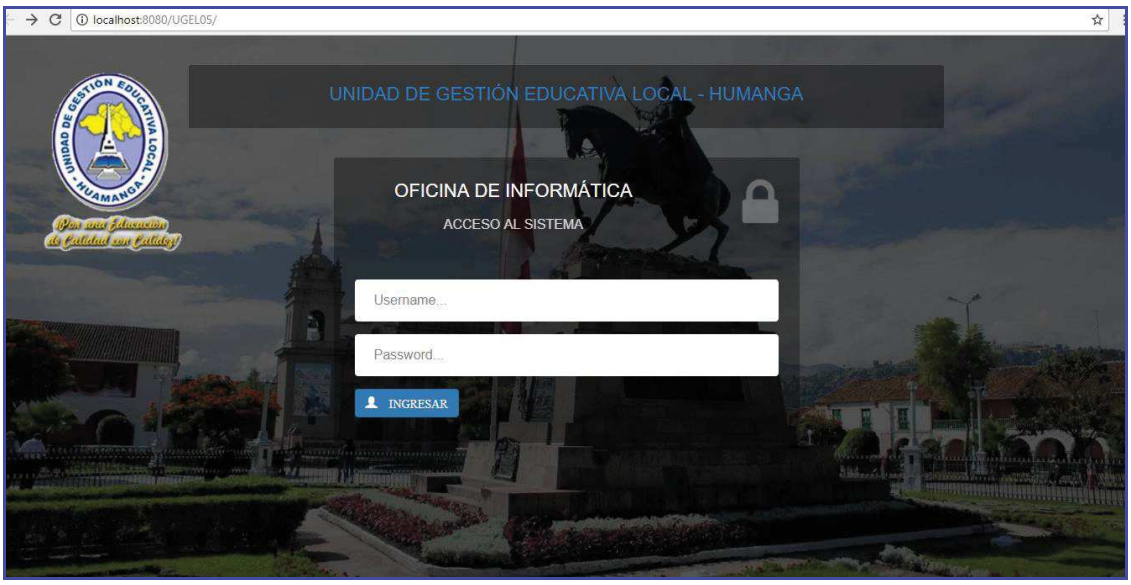


Figura 32. Interfaz de autenticación de usuario

Figura 33. Interfaz menú principal

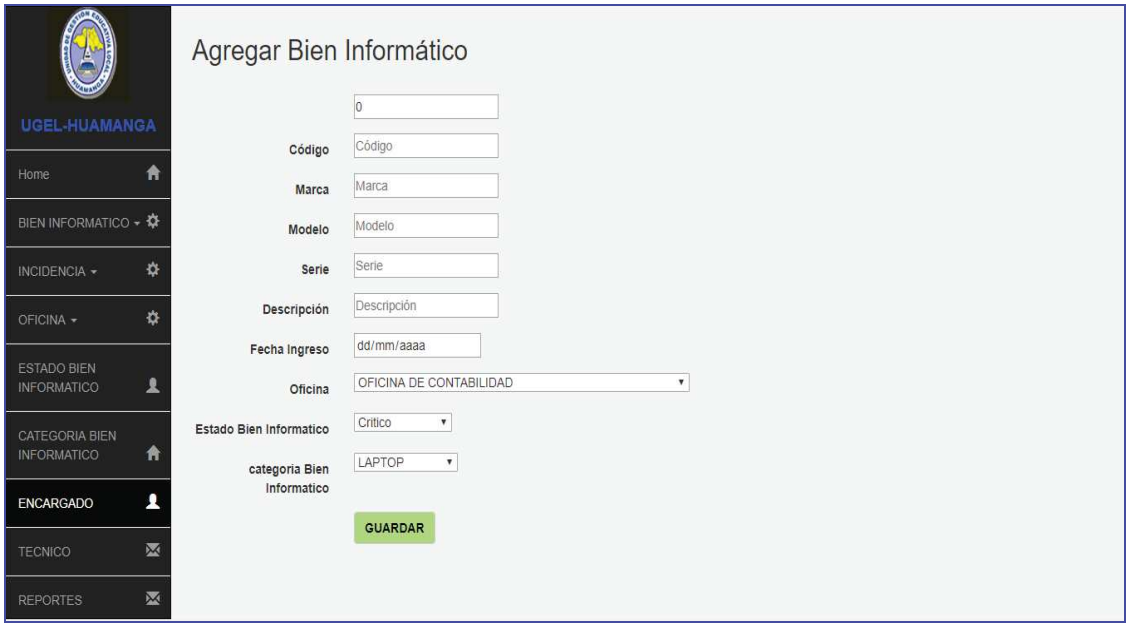


Figura 34. Interfaz agregar bien informático

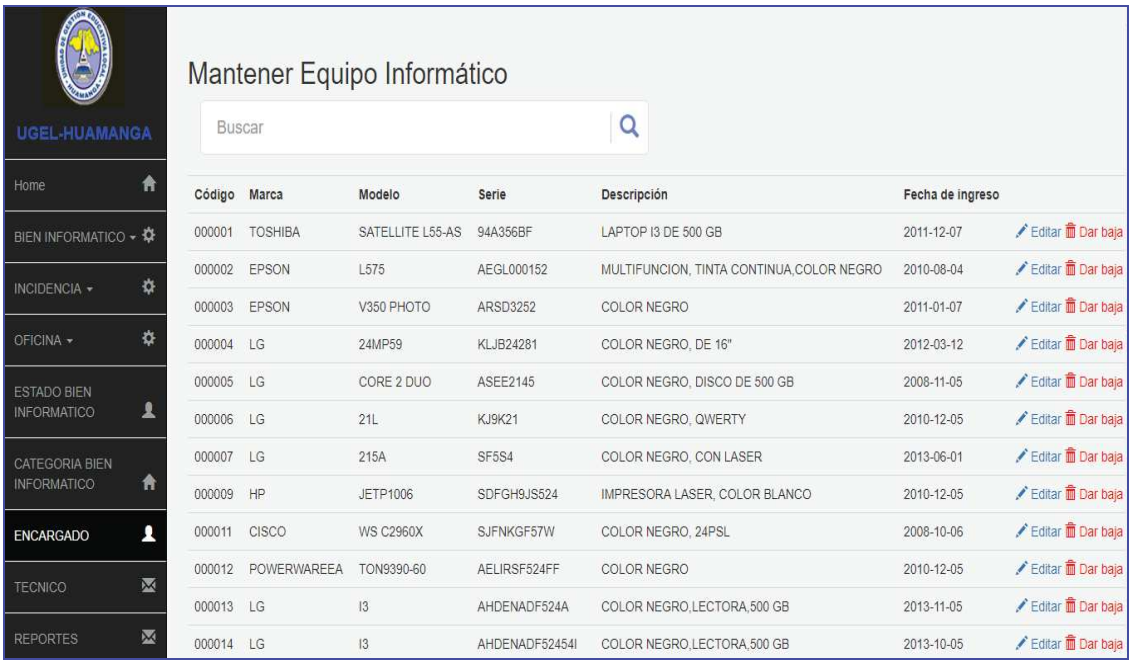


Figura 35. Interfaz mantener bien informático

Figura 36. Interfaz registrar incidencia

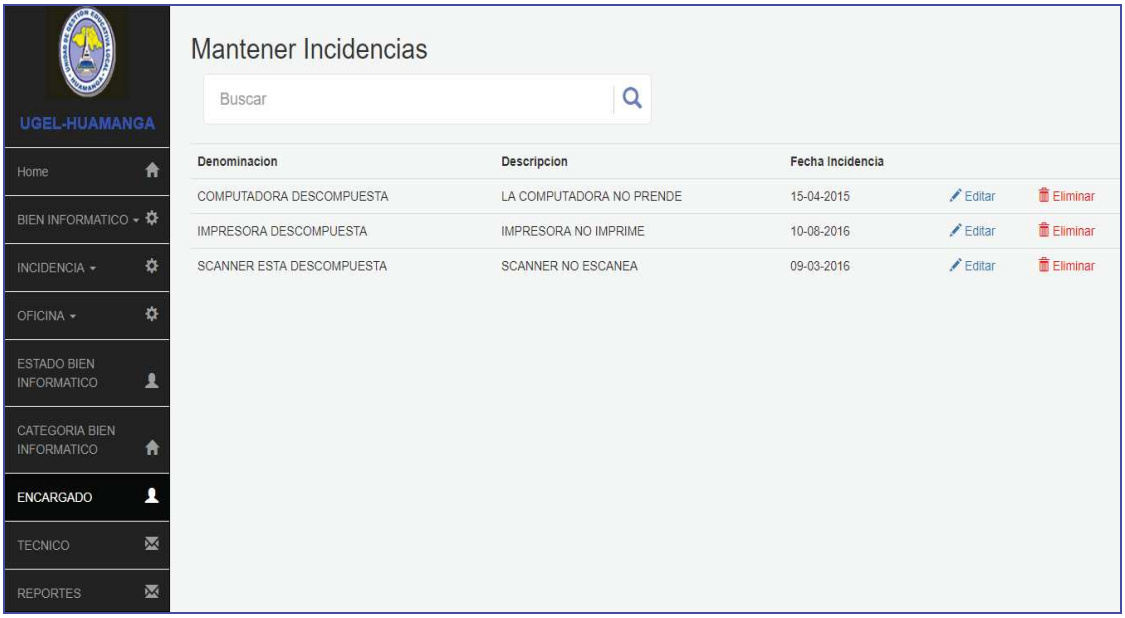


Figura 37. Interfaz mantener incidencia

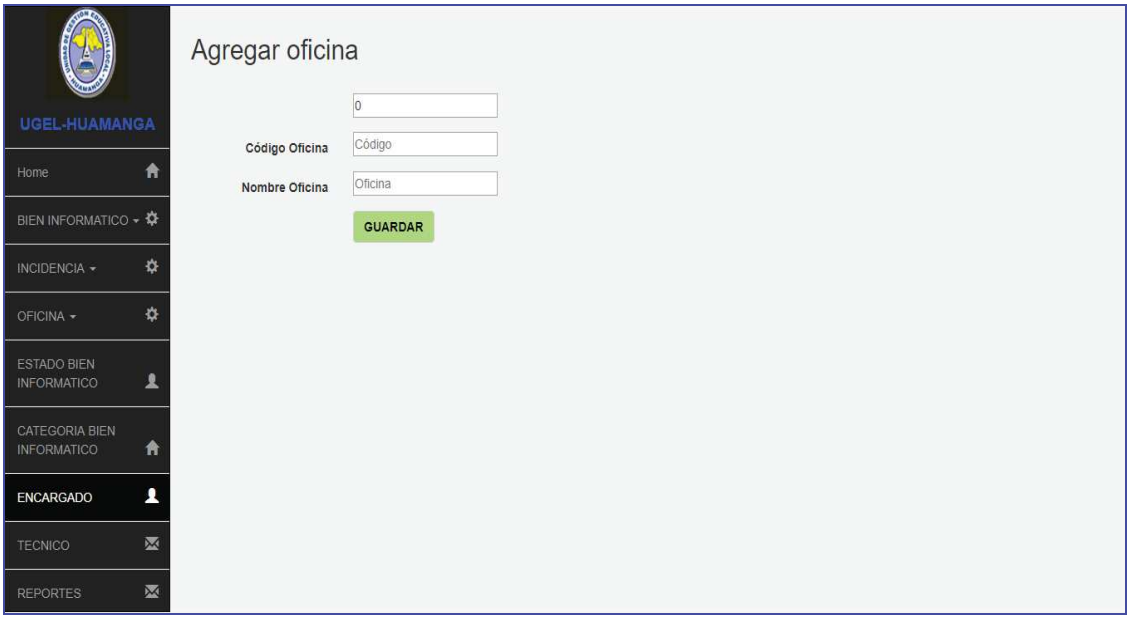


Figura 38. Interfaz agregar oficina

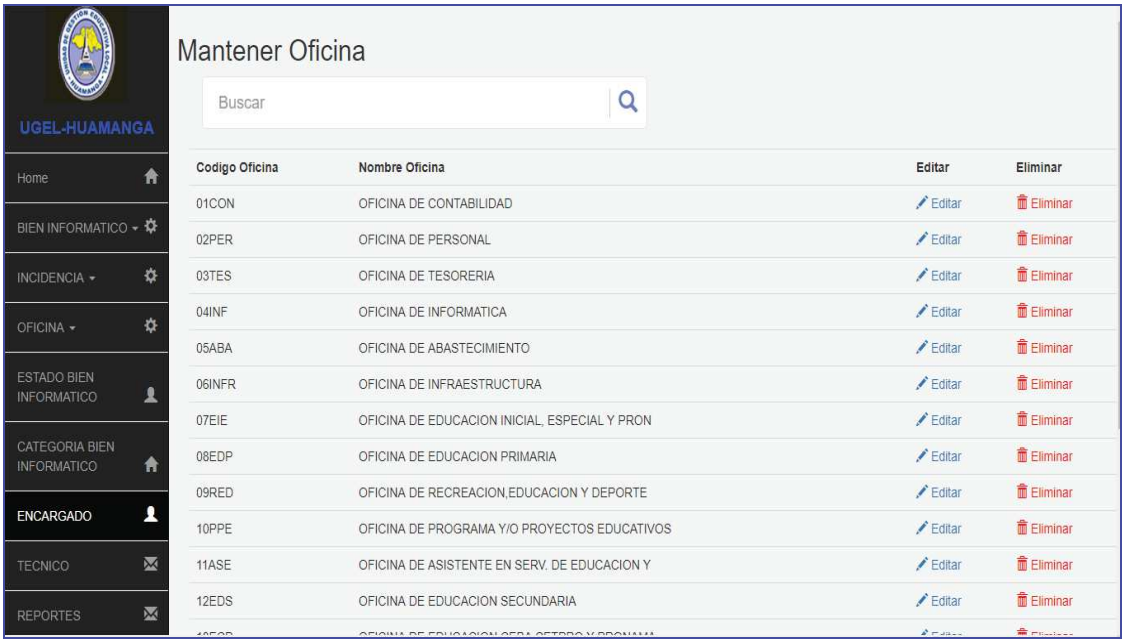


Figura 39. Interfaz Mantener oficina.

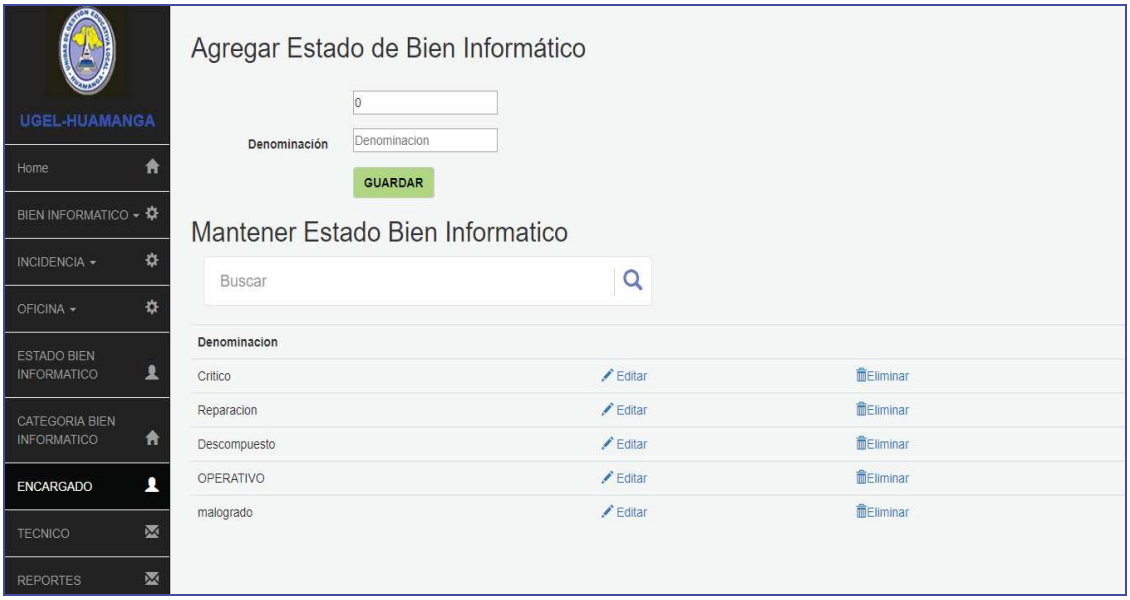


Figura 40. Interfaz Agregar y mantener estado bien informático.

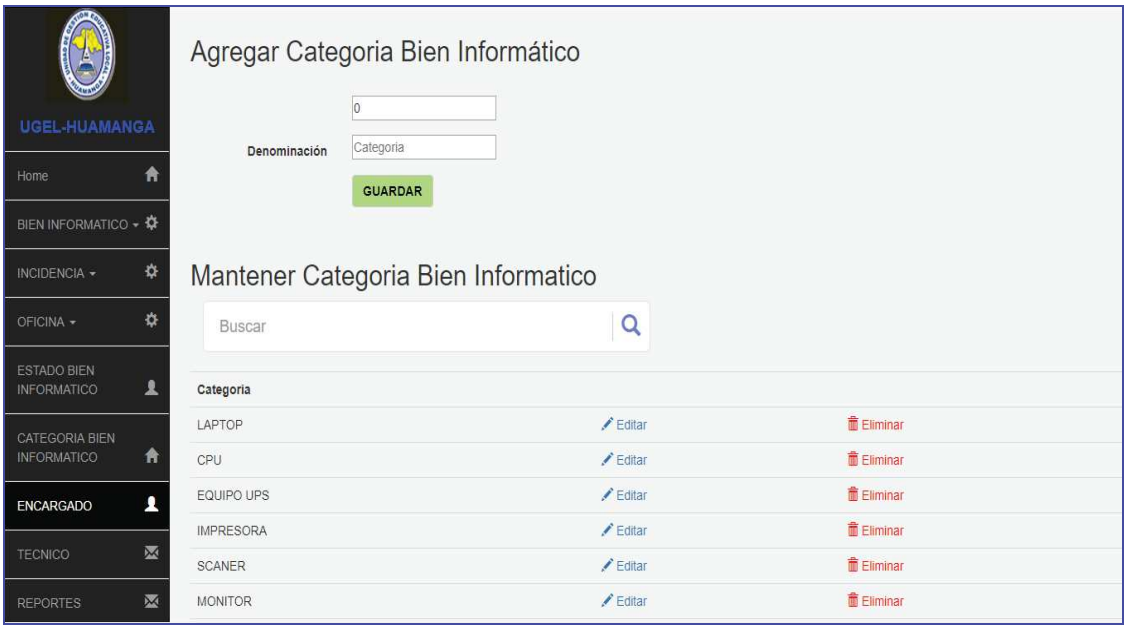


Figura 41. Interfaz de agregar y mantener categoría bien informática

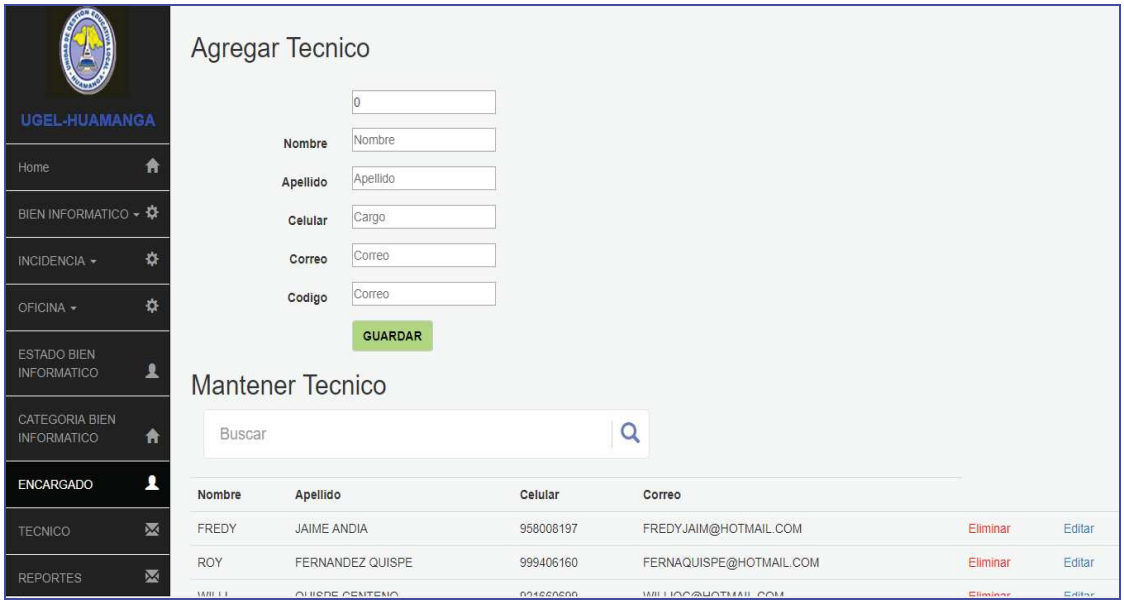


Figura 42. Interfaz de agregar y mantener encargado

Figura 42. Interfaz de Reportes

UGEL-HUAMANGA

Listado de Bienes Informaticos por oficina

CODIGO	MARCA	MODELO	SERIE	DESCRIPCIÓN	F. INGRESO
OFICINA DE INFORMATICA					
000001	TOSHIBA	SATELLITE L55-A	94A356BF	LAPTOP I3 DE 500 GB	7/12/11 0:00
000002	EPSON	L575	AEGL000152	MULTIFUNCION, TINTA CONTINUA,COLOR NEGRO	4/08/10 0:00
000003	EPSON	V350 PHOTO	ARSD3252	COLOR NEGRO	7/01/11 0:00
000004	LG	24MP59	KLJB24281	COLOR NEGRO, DE 16"	12/03/12 0:00
000005	LG	CORE 2 DUO	ASEE2145	COLOR NEGRO, DISCO DE 500 GB	5/11/08 0:00
000006	LG	21L	KJ9K21	COLOR NEGRO, QWERTY	5/12/10 0:00
000007	LG	215A	SF5S4	COLOR NEGRO, CON LASER	1/06/13 0:00
000008	APLUS	PLUSIII-1KL	SGH5E765T	COLOR NEGRO,DOBLE CONVERSION	11/12/15 0:00

Figura 43. Interfaz Reporte de bienes informáticos por oficina

UGEL-HUAMANGA

Lista de Incidencia por bien informatico

DENOMINACIÓN	FECHA INCIDENCIA	DESCRIPCIÓN
CODIGO BIEN : 000001		
COMPUTADORA POR ARREGLAR	15-04-2015	LA COMPUTADORA ESTA DESCOMPUESTA
CODIGO BIEN : 000002		
IMPRESORA SIN TINTA	10-08-2016	IMPRESORA QUE FALTA TINTA
CODIGO BIEN : 000003		
SCANNER ESTA SIENDO ARREGLADO	09-03-2016	SCANNER DESCOMPUESTA

Figura 44. Interfaz Reportes Lista de incidencias por bienes informáticos

4.2.2.13. DIAGRAMA DE ROBUSTEZ

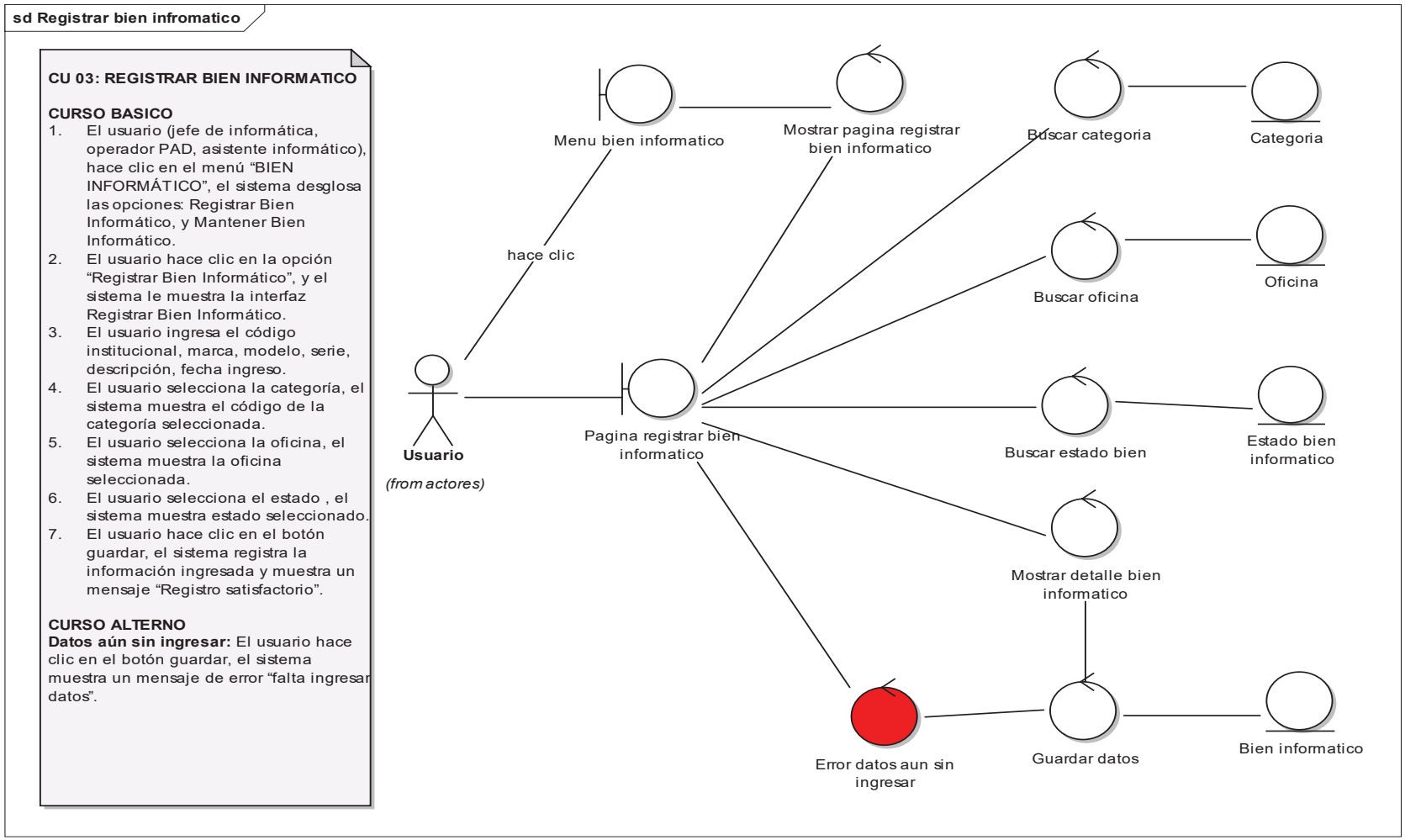


Figura 45. Diagrama de robustez: registrar bien informático

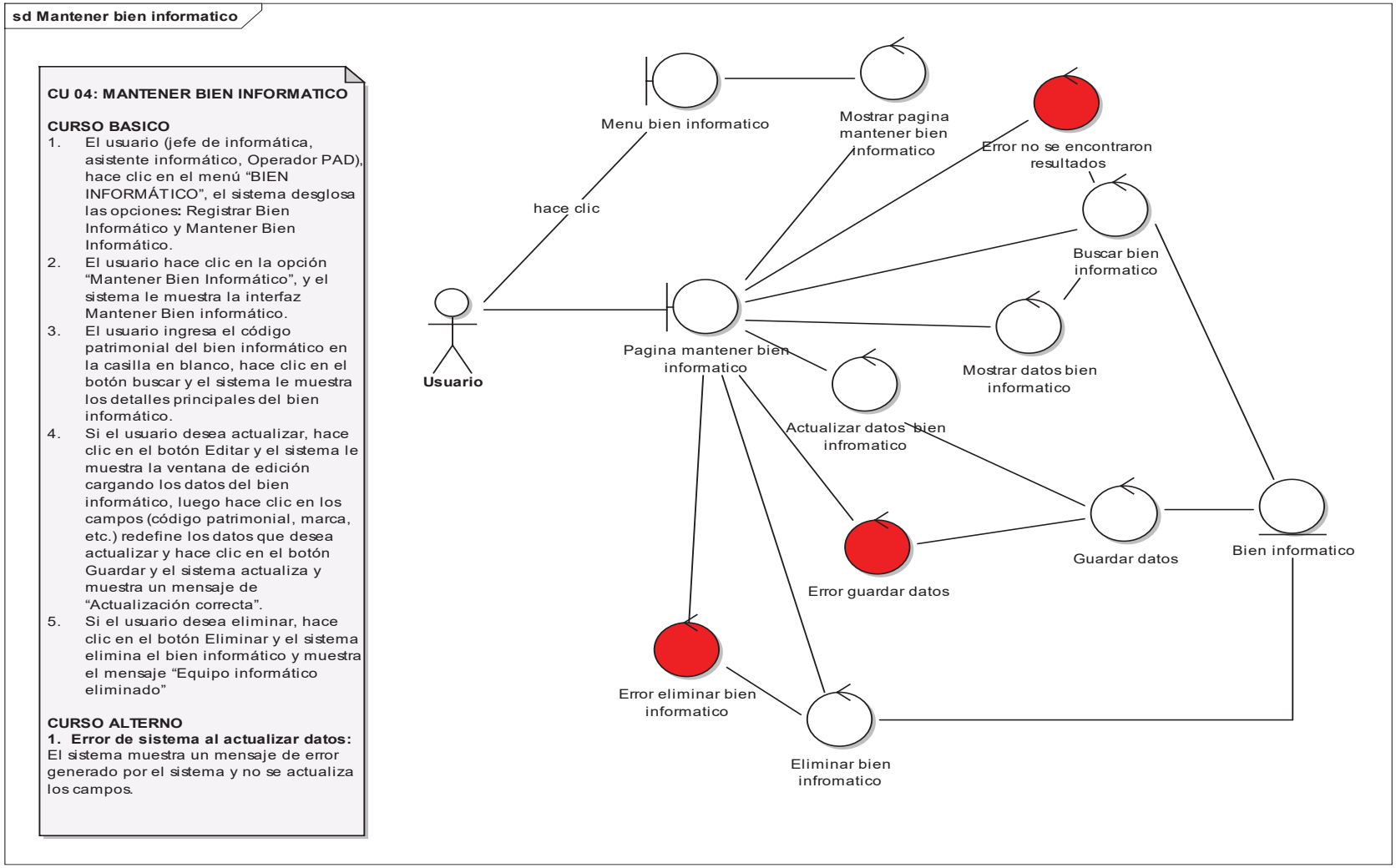


Figura 46. Diagrama de robustez: mantener bien informático

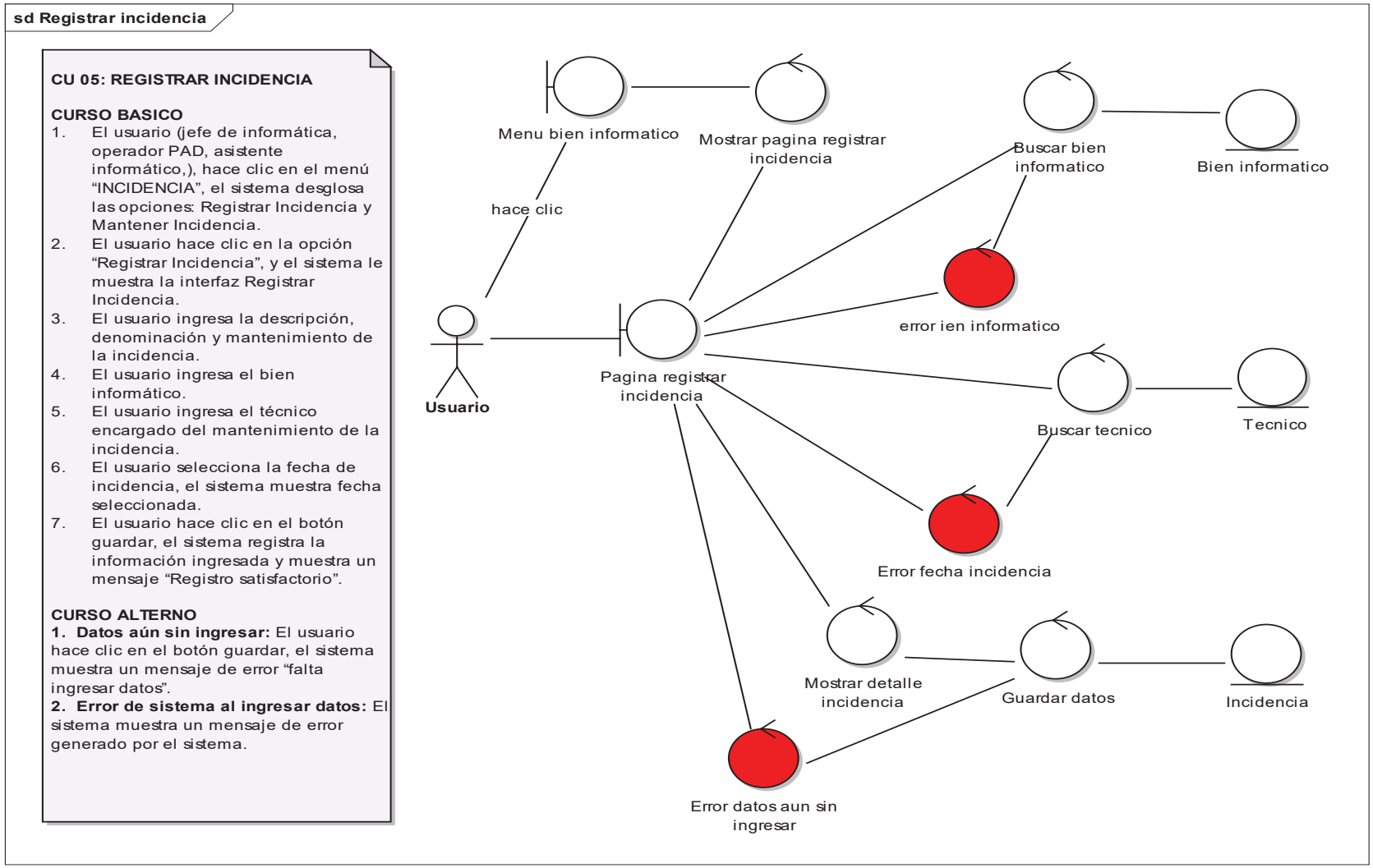


Figura 47. Diagrama de robustez: registrar incidencia

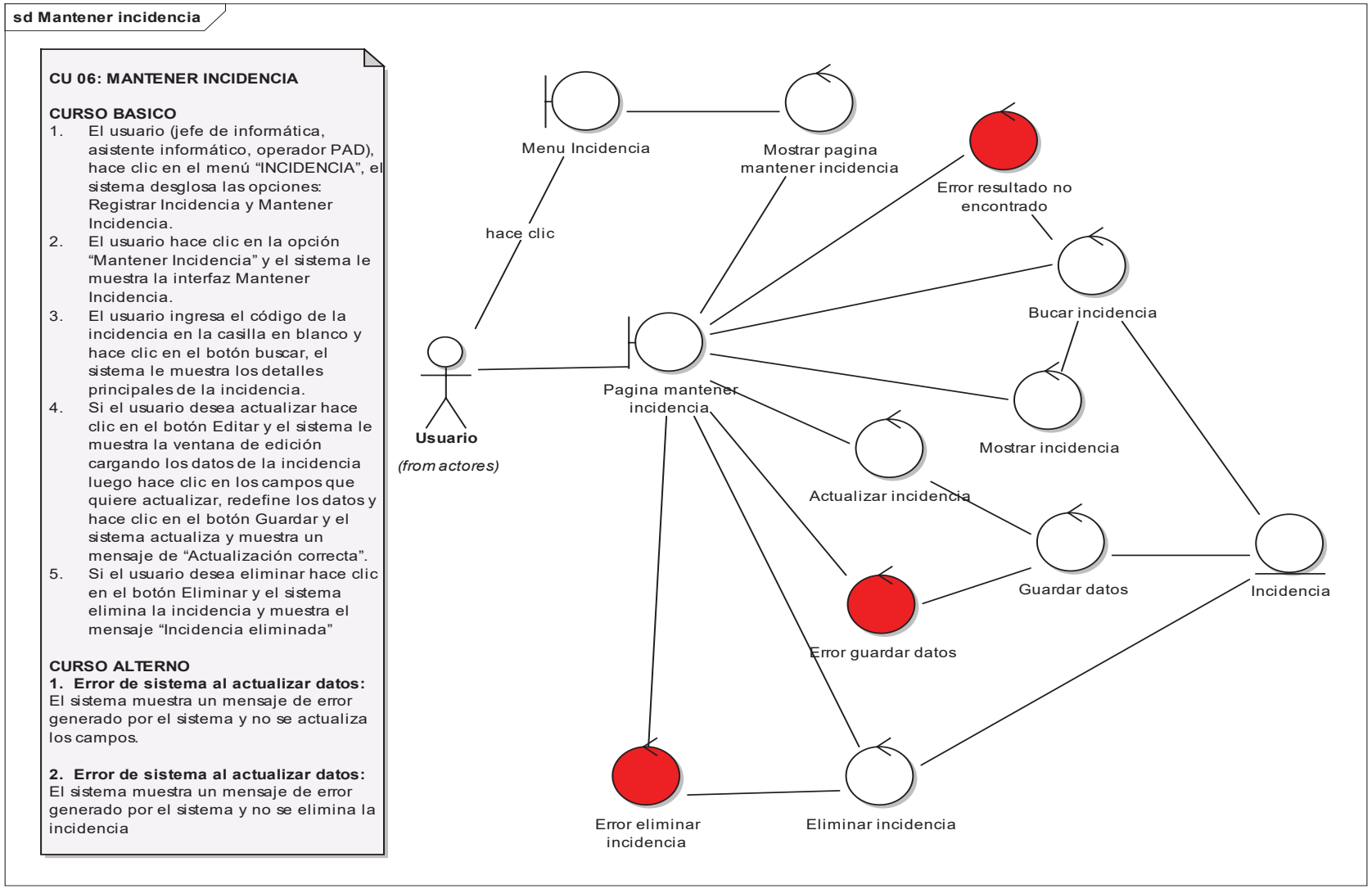


Figura 48. Diagrama de robustez: registrar bien informático

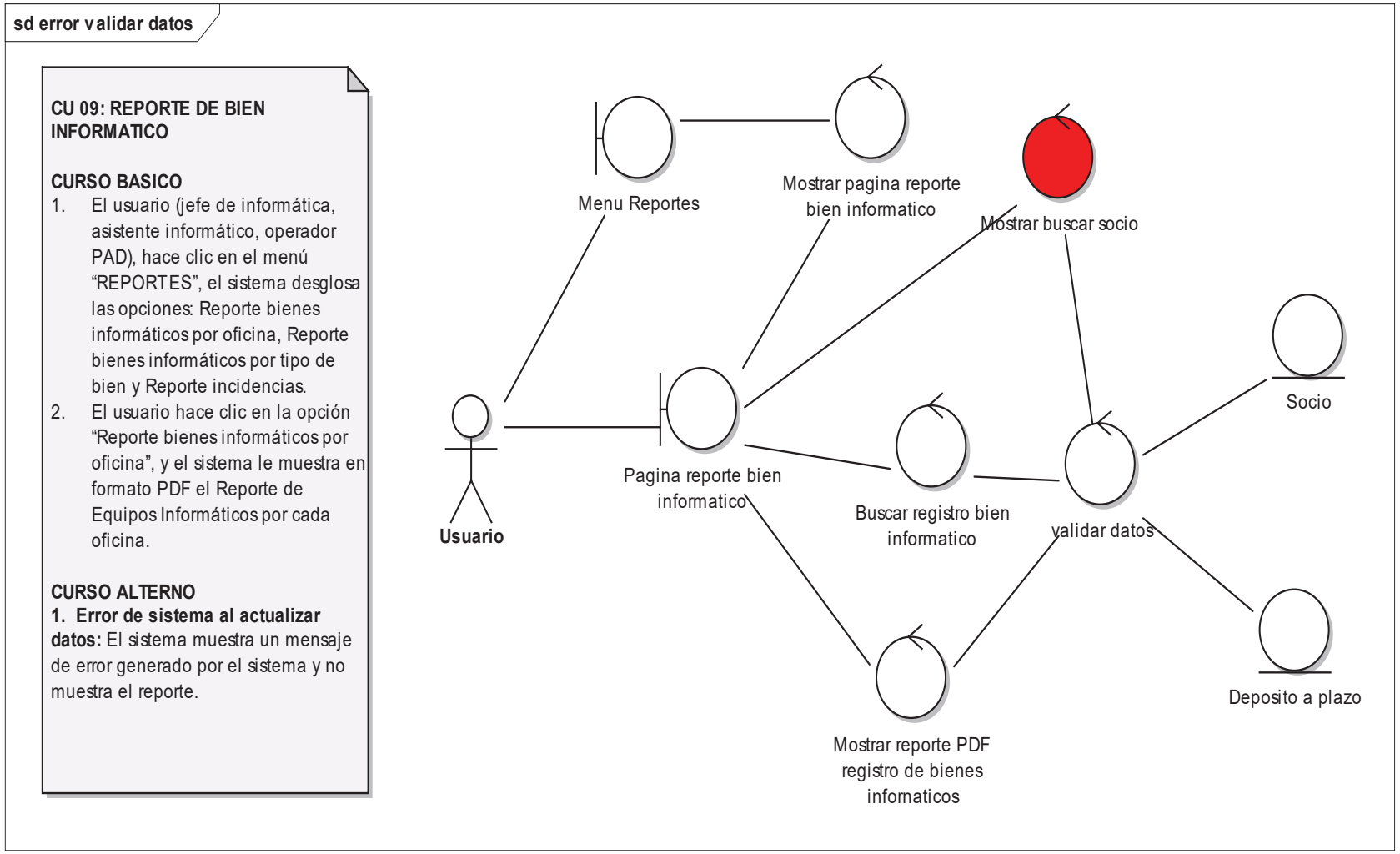


Figura 49. Diagrama de robustez: reporte bien informático

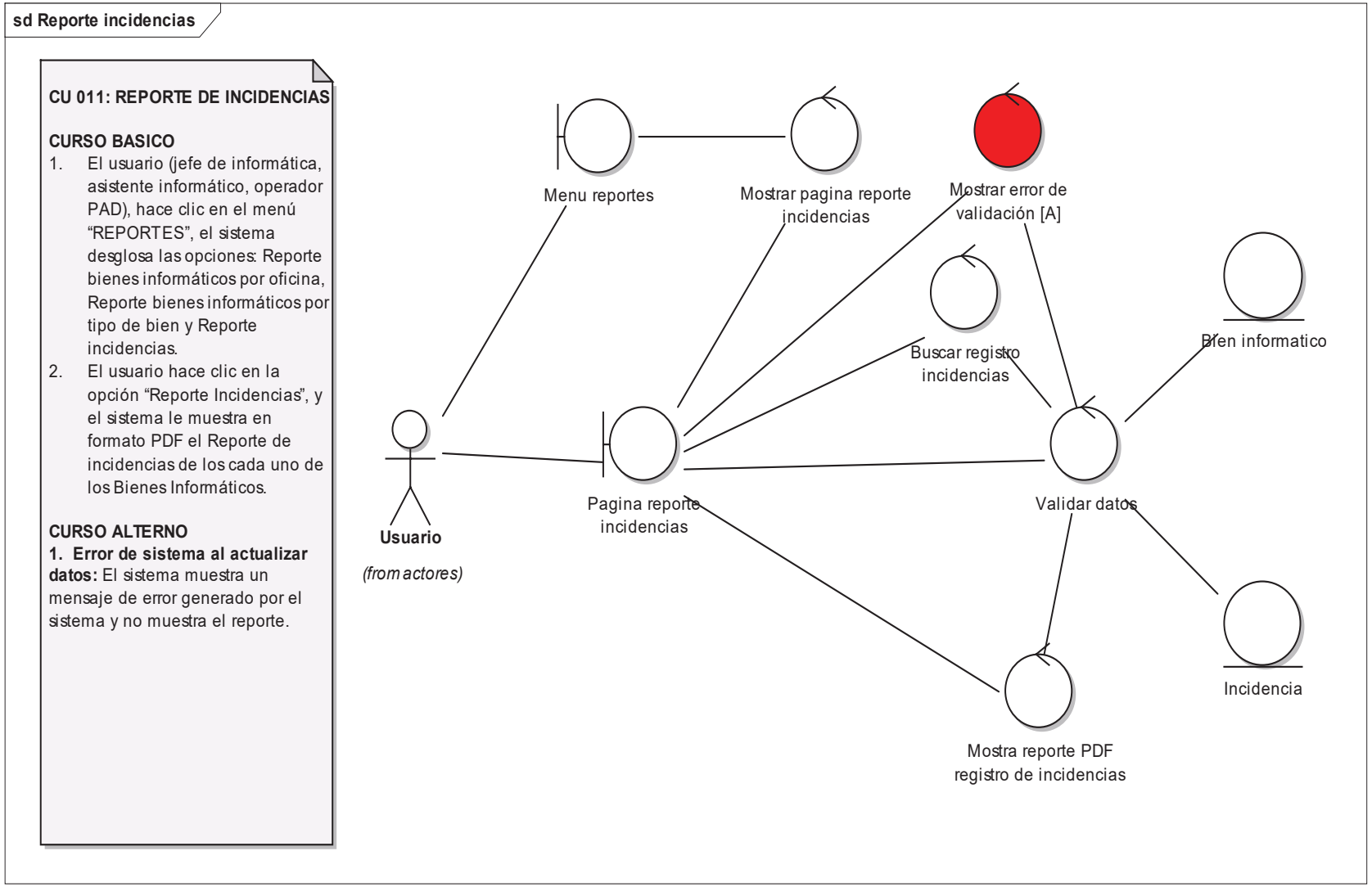


Figura 50. Diagrama de robustez: reporte incidencia

4.2.2.14. MODELO DE DOMINIO ACTUALIZADO

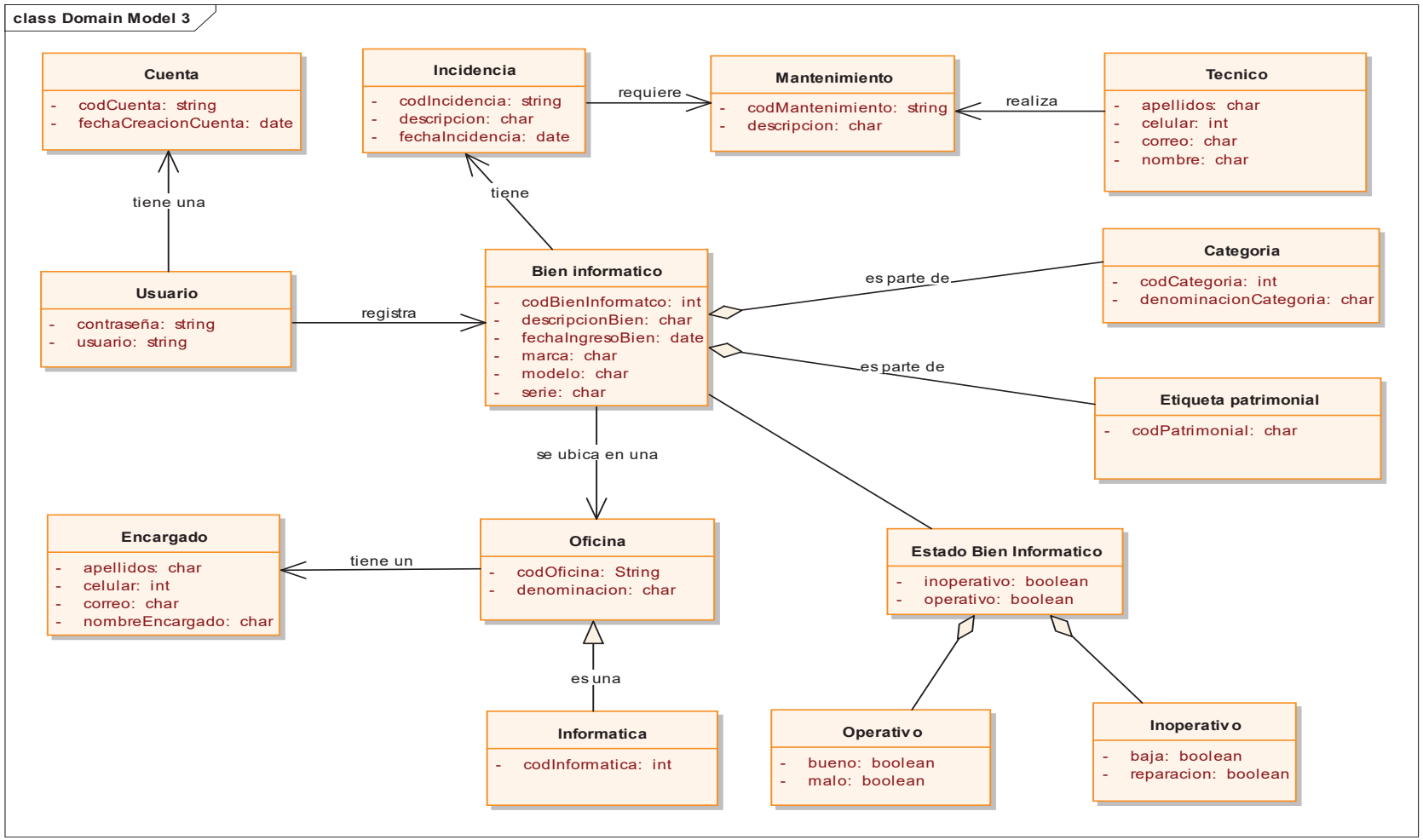


Figura 51. Modelo de dominio actualizado

4.2.2.15. DISEÑO DETALLADO

A. DIANGRAMA DE COMPONENTES

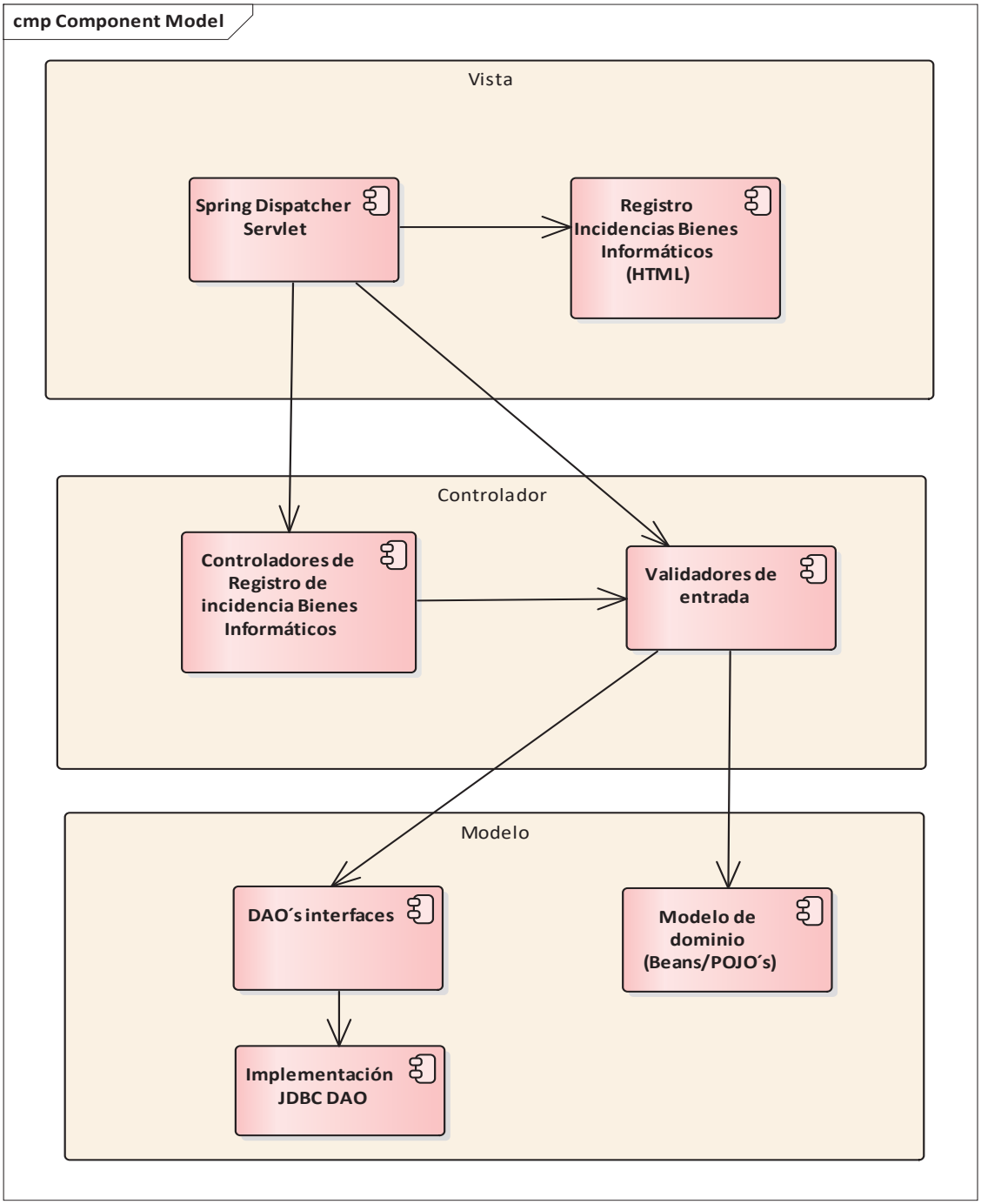


Figura 52. Diagrama de componentes

B. DIAGRAMA DE DESSPLEGUE

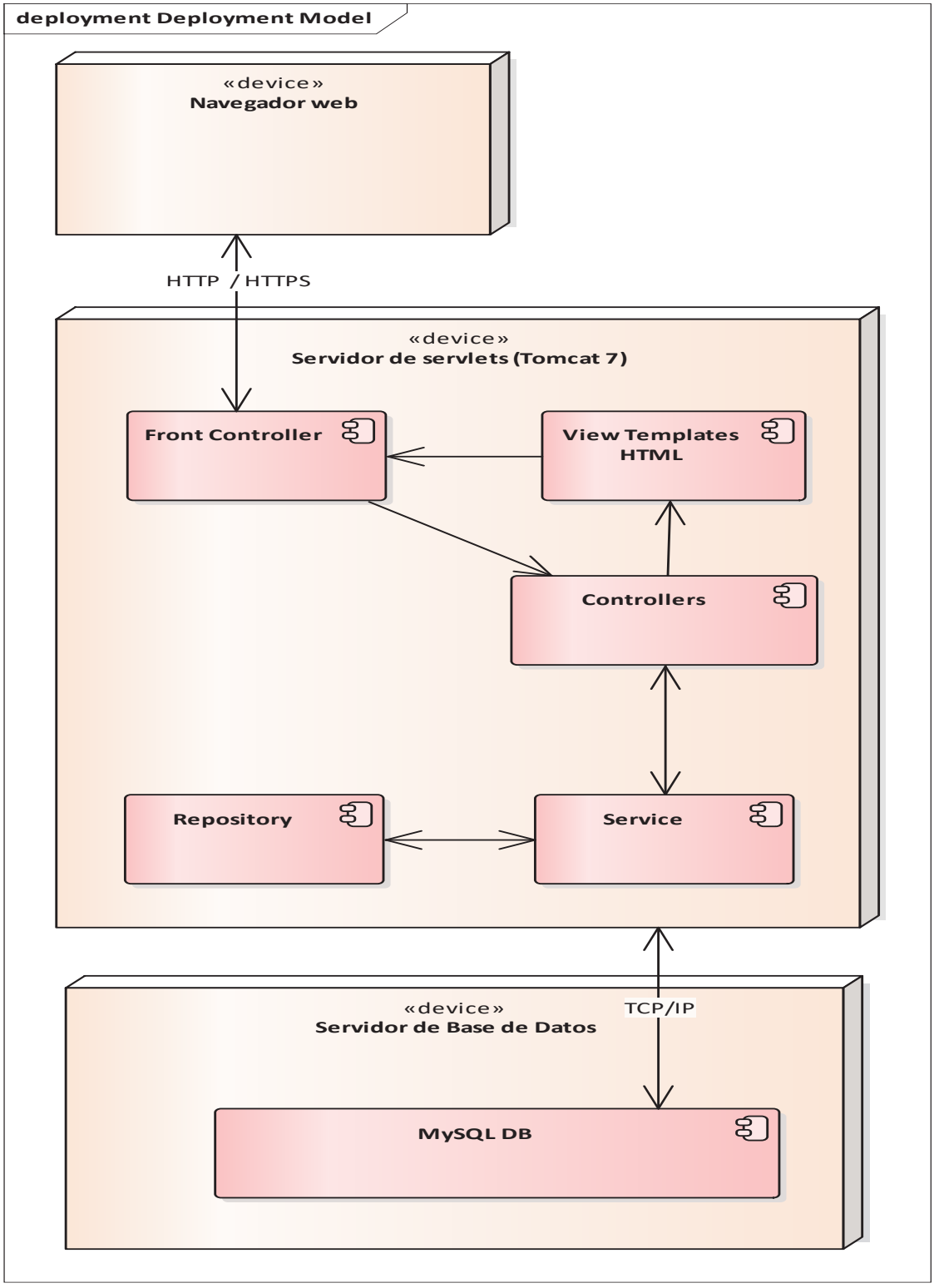


Figura 53. Diagrama de despliegue

4.2.2.16. DIAGRAMA DE SECUENCIA

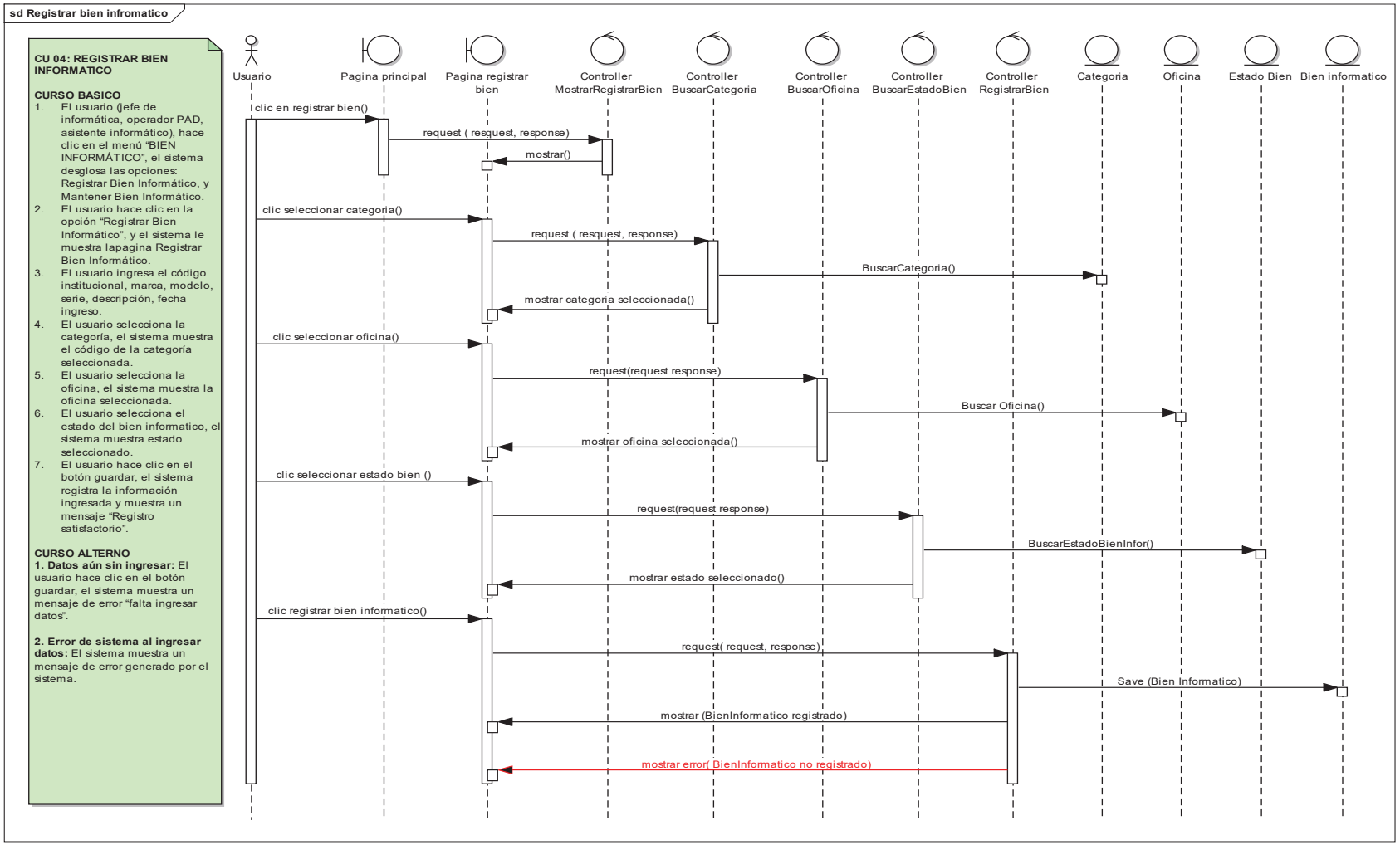


Figura 54. Diagrama de secuencia: registrar bien informático

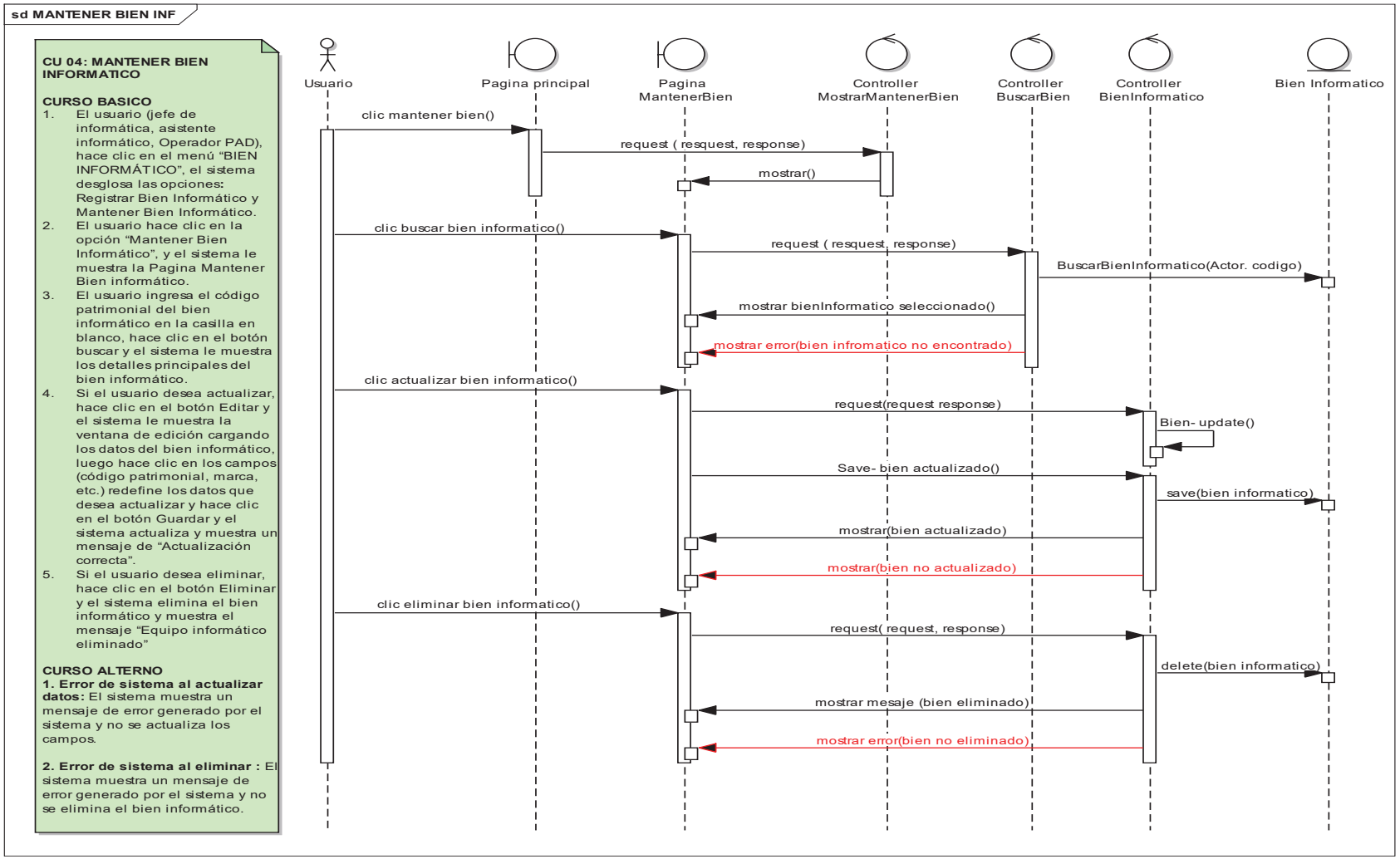


Figura 55. Diagrama de secuencia: mantener bien informático

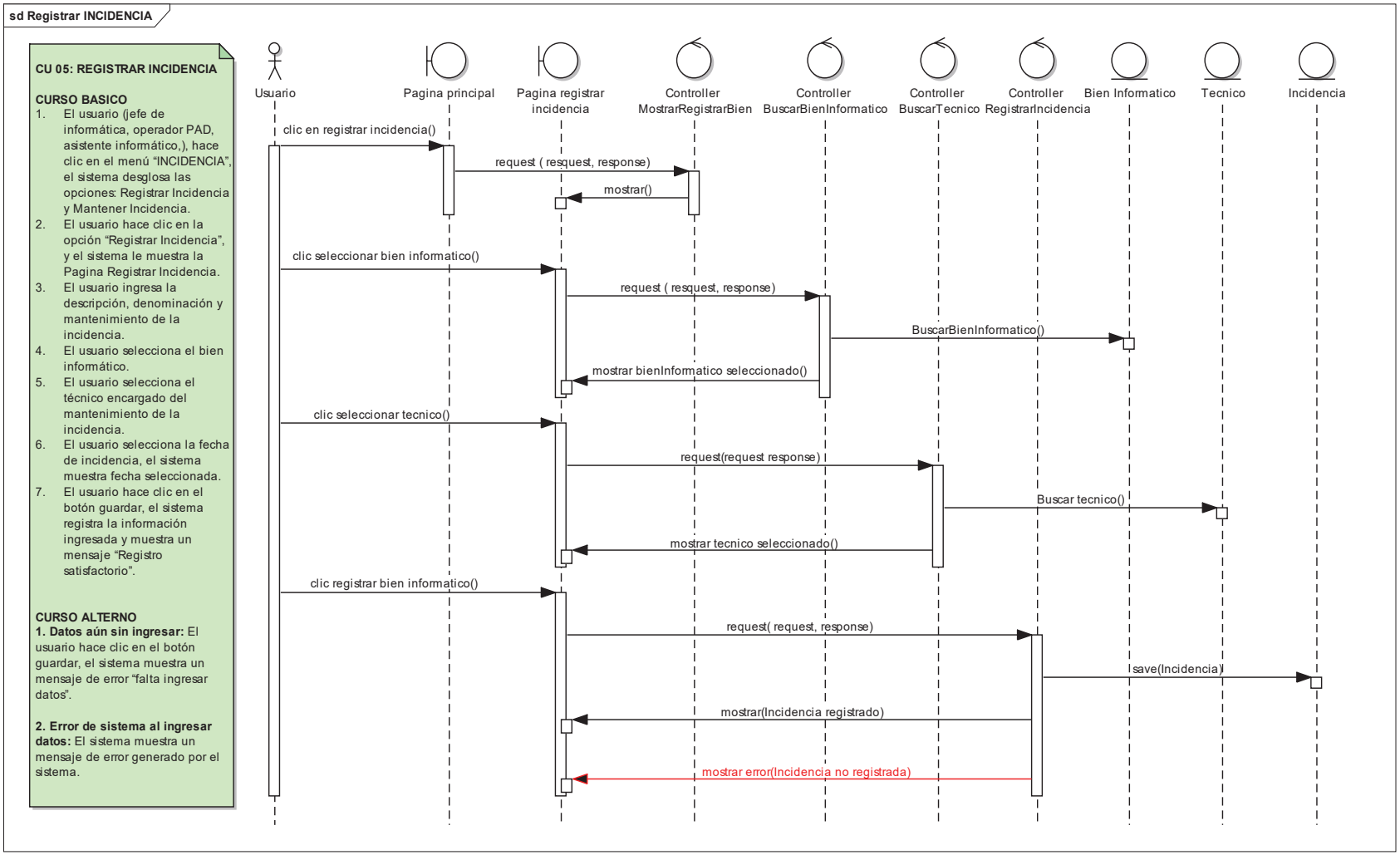


Figura 56. Diagrama de secuencia: registrar incidencia

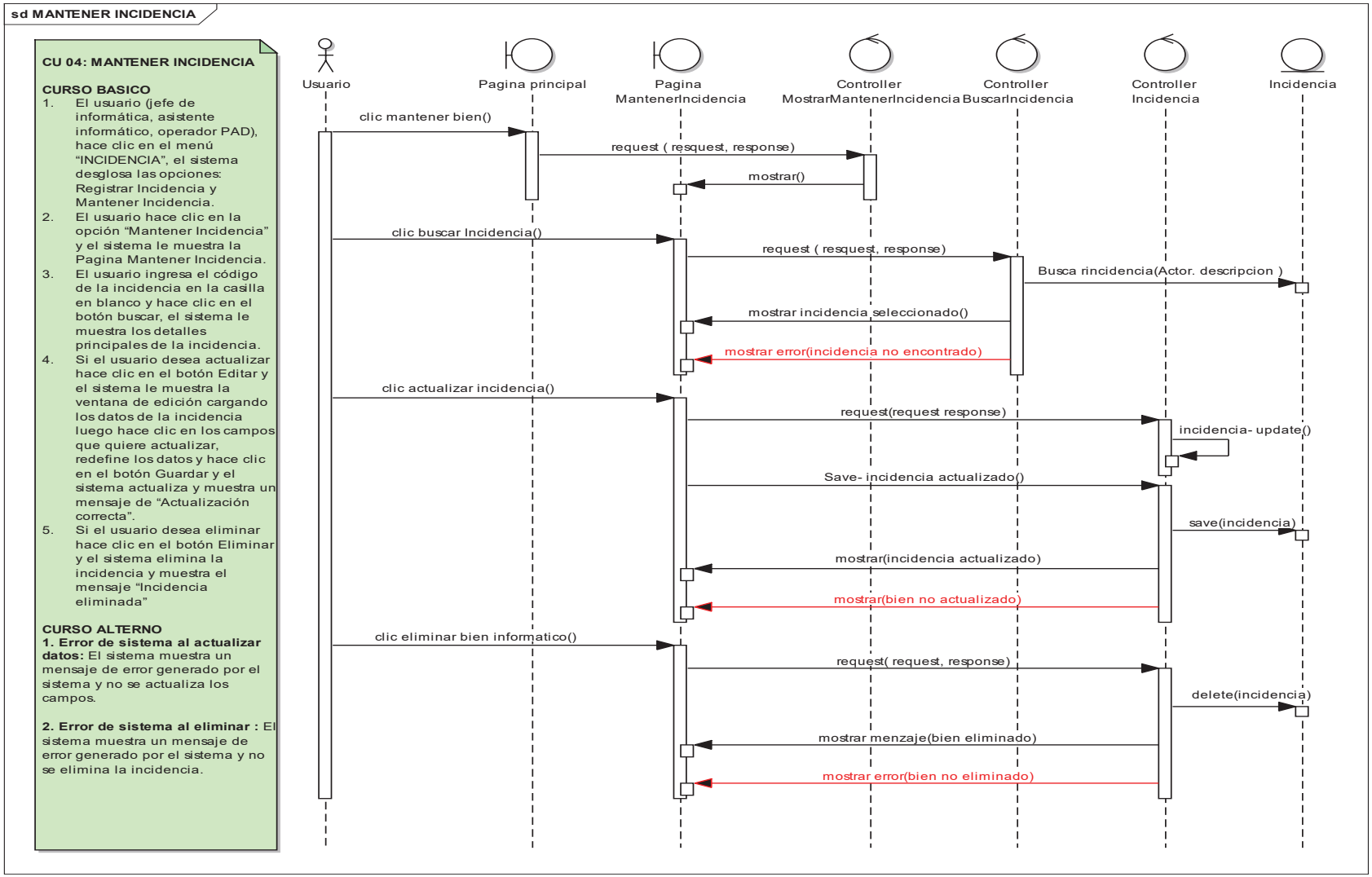


Figura 57. Diagrama de secuencia: Mantener incidencia

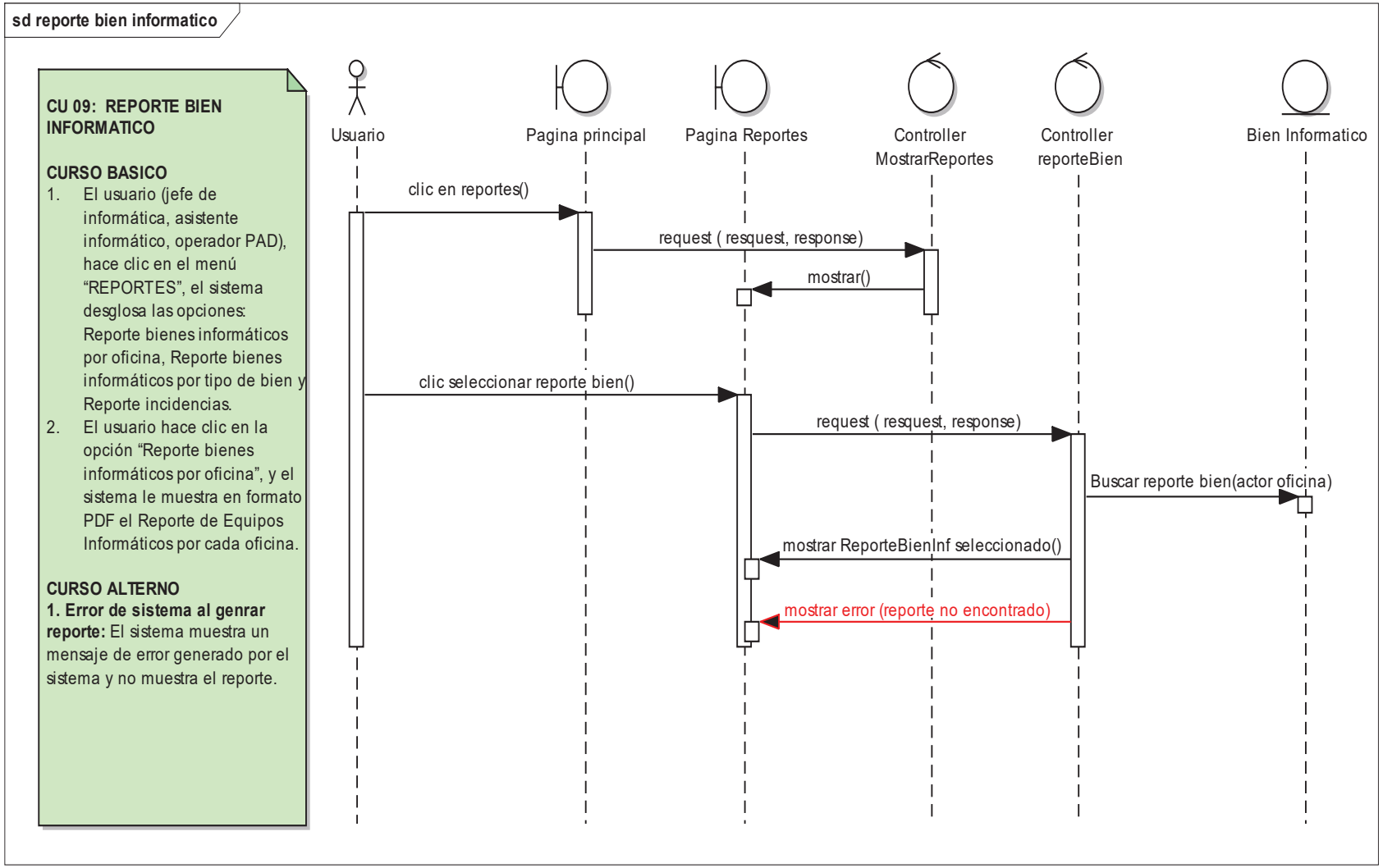


Figura 58. Diagrama de secuencia: Reporte bien informático

4.2.2.17. DIAGRAMA DE LA BASE DE DATOS

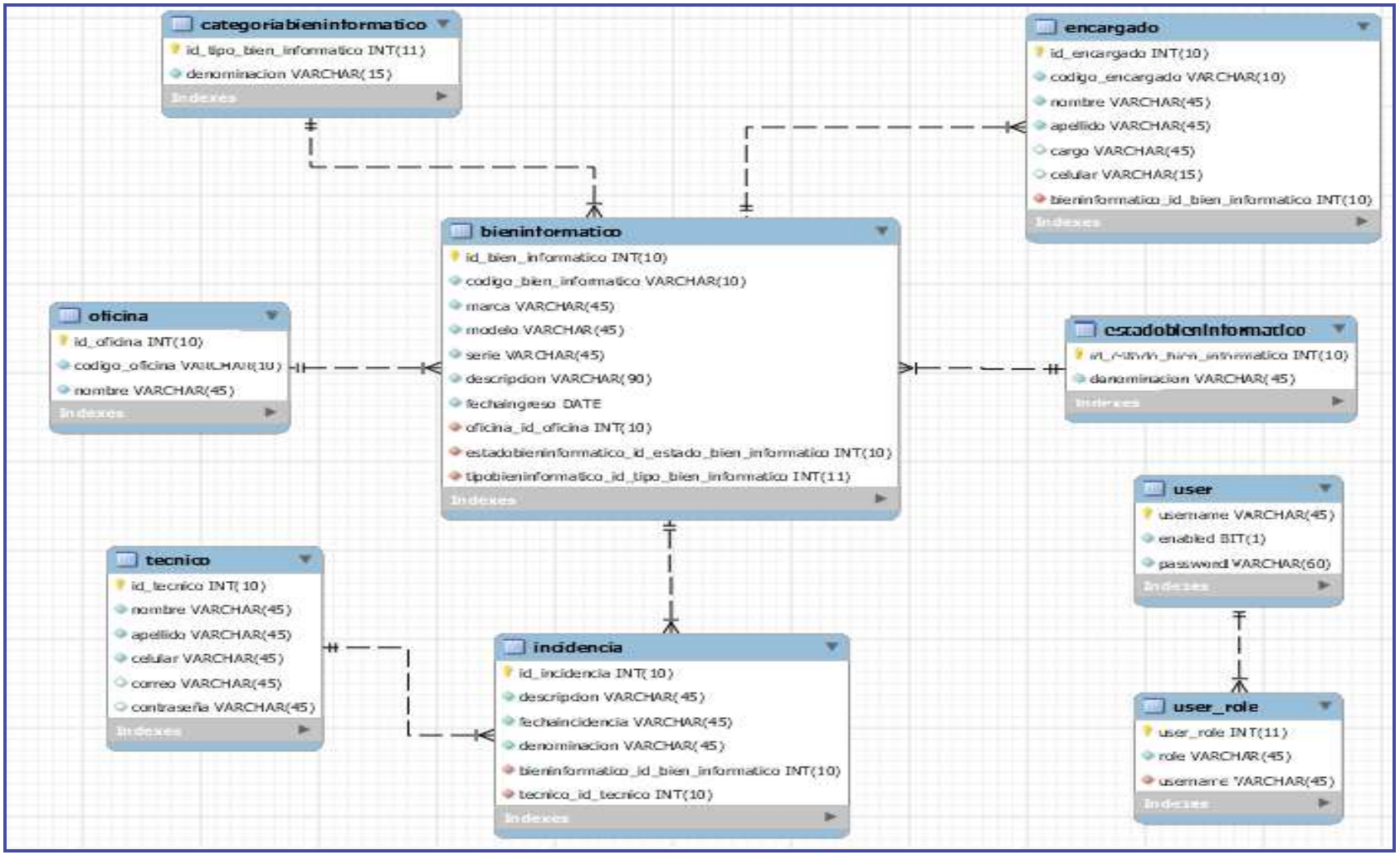


Figura 59. Diagrama de la Base de Datos

4.2.2.18. IMPLEMENTACIÓN

Para la implementación de los casos de uso se utilizó el IDE (entorno de desarrollo integrado) NetBeans, el lenguaje de programación JAVA junto al framework SPRING, servidor web de apache Tomcat, lenguaje HTML y el adminitsrador de base de datos MySQL.

A. DICCIONARIO DE DATOS

Tabla 31

NOMBRE DE LA TABLA: BienInformático

Nombre de la Columna	Tipo de dato	Clave primaria	Valores nulos	Auto Incremental
id_bien_informático	int	Si	No	Si
codigo_bien_informatico	varchar	No	No	No
marca	varchar	No	No	No
modelo	varchar	No	No	No
serie	varchar	No	No	No
descripcion	varchar	No	No	No
fechaingreso	date	No	No	No

Fuente: Elaboración propia (2020).

Tabla 32

NOMBRE DE LA TABLA: TipoBienInformático

Nombre de la Columna	Tipo de dato	Clave primaria	Valores nulos	Auto Incremental
id_tipo_bien_informatico	int	Si	No	Si
denominacion	varchar	No	No	No

Fuente: Elaboración propia (2020).

Tabla 33

NOMBRE DE LA TABLA: EstadoBienInformático

Nombre de la Columna	Tipo de dato	Clave primaria	Valores nulos	Auto Incremental
id_estado_bien_informatico	int	Si	No	Si
denominacion	varchar	No	No	No

Fuente: Elaboración propia (2020).

Tabla 34

NOMBRE DE LA TABLA: Incidencia

Nombre de la Columna	Tipo de dato	Clave primaria	Valores nulos	Auto Incremental
id_incidencia	int	Si	No	Si
descripcion	varchar	No	No	No
fechaincidencia	varchar	No	No	No
denominacion	varchar	No	No	No

Fuente: Elaboración propia (2020).

Tabla 35

NOMBRE DE LA TABLA: Encargado

Nombre de la Columna	Tipo de dato	Clave primaria	Valores nulos	Auto Incremental
id_encargado	int	Si	No	Si
codigo_encargado	varchar	No	No	No
nombre	varchar	No	No	No
apellido	varchar	No	No	No

cargo	varchar	No	No	No
celular	varchar	No	No	No

Fuente: Elaboración propia (2020).

Tabla 36

NOMBRE DE LA TABLA: Oficina

Nombre de la Columna	Tipo de dato	Clave primaria	Valores nulos	Auto Incremental
id_oficina	int	Si	No	Si
codigo_oficina	varchar	No	No	No
nombre	varchar	No	No	No

Fuente: Elaboración propia (2020).

Tabla 37

NOMBRE DE LA TABLA: Técnico

Nombre de la Columna	Tipo de dato	Clave primaria	Valores nulos	Auto Incremental
id_tecnico	int	Si	No	Si
nombre	varchar	No	No	No
apellido	varchar	No	No	No
celular	varchar	No	No	No
correo	varchar	No	No	No

Fuente: Elaboración propia (2020).

B. SCRIPT PARA LA CREACIÓN DE TABALAS DE LA BASE DE DATOS

```
CREATE TABLE `bieninformatico` (
  `id_bien_informatico` int(10) NOT NULL,
  `codigo_bien_informatico` varchar(10) NOT NULL,
  `marca` varchar(45) NOT NULL,
  `modelo` varchar(45) NOT NULL,
  `serie` varchar(45) NOT NULL,
  `descripcion` varchar(90) NOT NULL,
  `fechaingreso` date NOT NULL,
  `oficina_id_oficina` int(10) NOT NULL,
  `estadobieninformatico_id_estado_bien_informatico` int(10) NOT NULL,
  `tipobieninformatico_id_tipo_bien_informatico` int(11) NOT NULL
)
```

```
CREATE TABLE `estadobieninformatico` (
  `id_estado_bien_informatico` int(10) NOT NULL,
  `denominacion` varchar(45) NOT NULL
)
```

```
CREATE TABLE `tipobieninformatico` (
`id_tipo_bien_informatico` int(11) NOT NULL,
`denominacion` varchar(15) NOT NULL
)
```

```
CREATE TABLE `incidencia` (
  `id_incidencia` int(10) NOT NULL,
  `descripcion` varchar(45) NOT NULL,
  `fechaincidencia` varchar(45) NOT NULL,
  `denominacion` varchar(45) NOT NULL,
  `bieninformatico_id_bien_informatico` int(10) NOT NULL,
  `tecnico_id_tecnico` int(10) NOT NULL
)
```

```
CREATE TABLE `oficina` (
  `id_oficina` int(10) NOT NULL,
  `codigo_oficina` varchar(10) NOT NULL,
  `nombre` varchar(45) NOT NULL
)
```

```
CREATE TABLE `tecnico` (
  `id_tecnico` int(10) NOT NULL,
  `nombre` varchar(45) NOT NULL,
  `apellido` varchar(45) NOT NULL,
  `celular` varchar(45) NOT NULL,
  `correo` varchar(45) DEFAULT NULL,
  `contraseña` varchar(45) DEFAULT NULL
)
```

```
CREATE TABLE `encargado` (
  `id_encargado` int(10) NOT NULL,
  `codigo_encargado` varchar(10) NOT NULL,
  `nombre` varchar(45) NOT NULL,
  `apellido` varchar(45) NOT NULL,
  `cargo` varchar(45) DEFAULT NULL,
  `celular` varchar(15) DEFAULT NULL,
  `bieninformatico_id_bien_informatico` int(10) NOT NULL
)
```

C. CODIFCACIÓN DE LOS CASOS DE USO DE BIEN INFORMÁTICO

a. Código de la vista

REGISTRAR BIEN INFORMÁTICO

```
<div class="main ">
  <div class="container">
    <h3>Usuario : <span th:text="${user}" ></span></h3>
    <h2>Agregar equipo informático</h2> <br/>
<form class="form-horizontal" role="form"  action=""
  th:action="@{/saveEquipoInformatico}" th:object="${bienIntro}" method="post">
      <div class="form-group">
        <label class="col-lg-2 control-label"></label>
        <div class="col-lg-10">
         <input type="text" th:field="*{idBienInformatico}" readOnly="readonly" />
        </div>
      </div>
      <div class="form-group">
        <label class="col-lg-2 control-label">Código</label>
        <div class="col-lg-10">
          <input type="text" th:field="*{codigoBienInformatico}"
              placeholder="Código"/>
        </div>
      </div>

      <div class="form-group">
        <label class="col-lg-2 control-label">Marca</label>
        <div class="col-lg-10">
          <input type="text" th:field="*{marca}"
              placeholder="Marca"/>
        </div>
      </div>

      <div class="form-group">
```

```
    <label class="col-lg-2 control-label">Modelo</label>
    <div class="col-lg-10">
      <input  type="text" th:field="*{modelo}"
           placeholder="Modelo"/>
    </div>
  </div>
  <div class="form-group">
    <label class="col-lg-2 control-label">Serie</label>
    <div class="col-lg-10">
      <input type="text" th:field="*{serie}"
           placeholder="Serie"/>
    </div>
  </div>

  <div class="form-group">
    <label class="col-lg-2 control-label">Descripción</label>
    <div class="col-lg-10">
      <input type="text" th:field="*{descripcion}"
           placeholder="Descripción"/>
    </div>
  </div>

  <div class="form-group">
    <label class="col-lg-2 control-label">Fecha Ingreso</label>
    <div class="col-lg-10">
      <input type="date" th:field="*{fechaingreso}"
           placeholder="Fecha Ingreso"/>
    </div>
  </div>

  <div class="form-group">
    <label class="col-lg-2 control-label">Oficina</label>
    <div class="col-lg-10">
```

```
            <select th:field="*{oficinaIdOficina}">
                <option th:each="oficina : ${oficina}"
                        th:value="${oficina.idOficina}"
                        th:text="${oficina.nombre}"></option>
            </select>
        </div>
    </div>

    <div class="form-group">
        <label class="col-lg-2 control-label">Estado Bien informatico</label>
        <div class="col-lg-10">
            <select th:field="*{estadobieninformaticoIdEstadoBienInformatico}">
                <option th:each="estado : ${estado}"
                        th:value="${estado.idEstadoBienInformatico}"
                        th:text="${estado.denominacion}"></option>
            </select>
        </div>
    </div>

    <div class="form-group">
        <label class="col-lg-2 control-label">Tipo Bien informatico</label>
        <div class="col-lg-10">
            <select th:field="*{tipobieninformaticoIdTipoBienInformatico}">
                <option th:each="tipo : ${tipo}"
                        th:value="${tipo.idTipoBienInformatico}"
                        th:text="${tipo.denominacion}"></option>
            </select>
        </div>
    </div>

    <div class="form-group">
        <div class="col-lg-offset-2 col-lg-10">
            <button style="color: black; background-color: green" type="submit"
```

```
            class="btn btn-default"><b>GUARDAR</b></button>
          </div>
        </div>
      </form>
    </div>
```

MANTENER BIEN INFORMÁTICO

```
<div class="main ">
        <div class="container">
          <br/>
          <h2>Equipo Informático</h2>
          <div class="row">
            <div class="col-md-6">

              <div id="custom-search-input">
                <div class="input-group col-md-12">
                  <input  id="searchTerm"  type="text"  onkeyup="doSearch()"
                  class="form-control input-lg"
                      placeholder="Buscar"/>
                  <span class="input-group-btn">
                    <button class="btn btn-info btn-lg" type="button">
                      <i class="glyphicon glyphicon-search"></i>
                    </button>
                  </span>
                </div>
              </div>
              <br/>
            </div>
          </div>
          <table class="table" id="regTable">
            <tr>
```

```
                <th>Código</th>
                <th>Marca</th>
                <th>Modelo</th>
                <th>Serie</th>
                <th>Descripción</th>
                <th>Fecha de ingreso</th>
                <th></th>
            </tr>
            <tr th:each="equipoI : ${equipoI}">

                <td th:text="${equipoI.codigoBienInformatico}"></td>
                <td th:text="${equipoI.marca}"></td>
                <td th:text="${equipoI.modelo}"></td>
                <td th:text="${equipoI.serie}"></td>
                <td th:text="${equipoI.descripcion}"></td>
                <td th:text="${equipoI.fechaingreso}"></td>

                 <td><a
                th:href="@{/editBI/{id}(id=${equipoI.idBienInformatico})}"><span
                  class="glyphicon glyphicon-pencil"></span> Editar</a>
                 <a     th:href="@{/deleteBI/{id}(id=${equipoI.idBienInformatico})}"
                  style="color:red"><span
                 class="glyphicon    glyphicon-trash"    style="color:    red"></span>
                Eliminar</a></td>
            </tr>
        </table>
    </div>
</div>
```

VISTA ESTADO BIEN INFORMÁTICO

```
<div class="main ">
  <div class="container">
    <!-- Content Here -->
    <h2>Agregar Estado de Bien Informático</h2> <br/>
        <form class="form-horizontal" role="form" action=""
          th:action="@{/saveEstadoBI}" th:object="${estadoBIIngreso}"
            method="post">
          <div class="form-group">
        <label class="col-lg-2 control-label"></label>
        <div class="col-lg-10">
            <input type="text" th:field="*{idEstadoBienInformatico}"
             readOnly="readonly"/>
        </div>
      </div>
      <div class="form-group">
        <label class="col-lg-2 control-label">Denominación</label>
        <div class="col-lg-10">
          <input type="text" th:field="*{denominacion}"
              placeholder="Denominacion"/>
        </div>
      </div>
      <div class="form-group">
        <div class="col-lg-offset-2 col-lg-10">
          <button style="color: black; background-color: green" type="submit"
           class="btn btn-default"><b>GUARDAR</b></button>
        </div>
      </div>
    </form>

    <h2>Estado Bien Informatico</h2>
```

```
<div class="row">
  <div class="col-md-6">
    <div id="custom-search-input">
      <div class="input-group col-md-12">
        <input id="searchTerm" type="text" onkeyup="doSearch()"
         class="form-control input-lg"
        placeholder="Buscar"/>
        <span class="input-group-btn">
        <button class="btn btn-info btn-lg" type="button">
          <i class="glyphicon glyphicon-search" style="color: #328cd8"></i>
        </button>
      </span>
      </div>
    </div>
    <br/>
  </div>
</div>

<table class="table" id="regTable">
  <tr>
    <th>Denominacion</th>
    <th></th>
    <th></th>
  </tr>
  <tr th:each="estadoBI : ${estadoBI}">
    <td th:text="${estadoBI.denominacion}"></td>

    <td><a
   th:href="@{/editEstadoBI/{id}(id=${estadoBI.idEstadoBienInformatico})}"<
   ><span
          class="glyphicon glyphicon-pencil"></span> Editar</a></td>
            <td><a
           th:href="@{/deleteEstadoBI/{id}(id=${estadoBI.idEstadoBienInform
```

```
                    atico})}" style="color: red"><span
            class="glyphicon glyphicon-trash" style="color:red"></span>
          Eliminar</a></td>
        </tr>
      </table>
    </div>
</div>
```

REGISTRAR OFICINA

```
<div class="main">
  <!-- Content Here -->
  <div class="container">
    <h2>Agregar oficina</h2> <br/>
      <form class="form-horizontal" role="form" action="" th:action="@{/save}"
        th:object="${oficinaIntro}" method="post">
       <div class="form-group">
          <label class="col-lg-2 control-label"></label>
          <div class="col-lg-10">
            <input type="text" th:field="*{idOficina}" readOnly="readonly" />
          </div>
       </div>
       <div class="form-group">
          <label class="col-lg-2 control-label">Código Oficina</label>
          <div class="col-lg-10">
            <input type="text" th:field="*{codigoOficina}"
                placeholder="Código" />
          </div>
       </div>
       <div class="form-group">
          <label class="col-lg-2 control-label">Nombre Oficina</label>
          <div class="col-lg-10">
```

```
            <input type="text" th:field="*{nombre}"
                placeholder="Oficina" />
          </div>
        </div>

        <div class="form-group">
          <div class="col-lg-offset-2 col-lg-10">
            <button style="color: black; background-color: green" type="submit"
             class="btn btn-default"><b>GUARDAR</b></button>
          </div>
        </div>
      </form>
    </div>
</div>
```

MANTENER OFICINA

```
<div class="main">
  <!-- Content Here -->
  <h2>Lista de Oficina</h2>

  <div class="container">
    <div class="row">
      <div class="col-md-6">

        <div id="custom-search-input">
          <div class="input-group col-md-12">
            <input id="searchTerm" type="text" onkeyup="doSearch()"
                  class="form-control input-lg" placeholder="Buscar" />
            <span class="input-group-btn">
            <button class="btn btn-info btn-lg" type="button">
              <i class="glyphicon glyphicon-search" style="color: #328cd8"></i>
```

```
                </button>
              </span>
              </div>
            </div>
            <br/>
          </div>
        </div>
        <table class="table" id="regTable">
          <tr>

            <th>Codigo Oficina</th>
            <th>Nombre Oficina</th>
            <th>Editar</th>
            <th>Eliminar</th>
          </tr>
          <tr th:each="oficina : ${oficina}">

            <td th:text="${oficina.codigoOficina}"></td>
            <td th:text="${oficina.nombre}"></td>

             <td > <a th:href="@{/edit/{id}(id=${oficina.idOficina})}"><span
            class="glyphicon glyphicon-pencil"></span> Editar</a>  </td>
            <td > <a th:href="@{/delete/{id}(id=${oficina.idOficina})}"
            style="color:red"><span class="glyphicon glyphicon-trash"
            style="color:red"></span> Eliminar</a>  </td>
          </tr>
        </table>
    </div>
</div>
```

VISTA ENCARGADO

```
<div class="main ">
  <div class="container">
    <h2>Agregar encargado</h2> <br/>
     <form class="form-horizontal" role="form" th:action="@{/saveEncargado}"
      th:object="${encargadoIngreso}" method="post">
      <div class="form-group">
        <label class="col-lg-2 control-label"></label>
        <div class="col-lg-10">
          <input th:field="*{idEncargado}" readOnly="readonly" />
        </div>
      </div>
      <div class="form-group">
        <label class="col-lg-2 control-label">Codigo</label>
        <div class="col-lg-10">
          <input type="text" th:field="*{codigoEncargado}"
              placeholder="Codigo"/>
        </div>
      </div>

      <div class="form-group">
        <label class="col-lg-2 control-label">Nombre</label>
        <div class="col-lg-10">
          <input type="text" th:field="*{nombre}"
              placeholder="Nombre"/>
        </div>
      </div>
      <div class="form-group">
        <label class="col-lg-2 control-label">Apellido</label>
        <div class="col-lg-10">
          <input type="text" th:field="*{apellido}"
              placeholder="Apellido"/>
```

```
        </div>
    </div>
    <div class="form-group">
        <label class="col-lg-2 control-label">Cargo</label>
        <div class="col-lg-10">
            <input type="text" th:field="*{cargo}"
                placeholder="Cargo"/>
        </div>
    </div>
    <div class="form-group">
        <label class="col-lg-2 control-label">Celular</label>
        <div class="col-lg-10">
            <input type="text" th:field="*{celular}"
                placeholder="Celular"/>
        </div>
    </div>
    <div class="form-group">
        <label class="col-lg-2 control-label">Bien Informatico</label>
        <div class="col-lg-10">
            <select th:field="*{bieninformaticoIdBienInformatico}">
                <option th:each="bienInformatico : ${bienInformatico}"
                    th:value="${bienInformatico.idBienInformatico}"
                    th:text="${bienInformatico.codigoBienInformatico}"></option>
            </select>
        </div>
    </div>

    <div class="form-group">
        <div class="col-lg-offset-2 col-lg-10">
            <button style="color: black; background-color: green" type="submit"
             class="btn btn-default">GUARDAR</button>
        </div>
    </div>
```

```
</form>

<h2>Encargado</h2>
<div class="row">
  <div class="col-md-6">
    <div id="custom-search-input">
      <div class="input-group col-md-12">
        <input id="searchTerm" type="text" onkeyup="doSearch()"
         class="form-control input-lg"
         placeholder="Buscar"/>
        <span class="input-group-btn">
        <button class="btn btn-info btn-lg" type="button">
          <i class="glyphicon glyphicon-search" ></i>
        </button>
      </span>
      </div>
    </div>
    <br/>
  </div>
</div>

<table id="regTable" class="table">
  <tr>
    <th>Codigo</th>
    <th>Nombre</th>
    <th>Apellido</th>
    <th>Cargo</th>
    <th>Celular</th>
    <th></th>
    <th></th>
  </tr>
  <tr th:each="encargado : ${encargado}">
    <td th:text="${encargado.codigoEncargado}"></td>
```

```
            <td th:text="${encargado.nombre}"></td>
            <td th:text="${encargado.apellido}"></td>
            <td th:text="${encargado.cargo}"></td>
            <td th:text="${encargado.celular}"></td>
          <td > <a
            th:href="@{/editEncargado/{id}(id=${encargado.idEncargado})}"><span
             class="glyphicon glyphicon-pencil"></span> Editar</a>  </td>
             <td > <a th:href="@{/deleteEncargado/{id}(id=${encargado.idEncargado})}
              " ><span
                    class="glyphicon glyphicon-trash"  ></span> Eliminar</a>  </td>
          </tr>
        </table>
    </div>
</div>
```

VISTA TÉCNICO

```
<div class="main ">
   <div class="container">
      <h2>Agregar Tecnico</h2> <br/>
        <form    class="form-horizontal"    role="form"    th:action="@{/saveTecnico}"
         th:object="${tecnicoIngreso}" method="post">
         <div class="form-group">
            <label class="col-lg-2 control-label"></label>
            <div class="col-lg-10">
               <input type="text" th:field="*{idTecnico}" readOnly="readonly" />
            </div>
         </div>

         <div class="form-group">
            <label class="col-lg-2 control-label">Nombre</label>
            <div class="col-lg-10">
               <input type="text" th:field="*{nombre}"
```

```
                placeholder="Nombre"/>
        </div>
    </div>
    <div class="form-group">
        <label class="col-lg-2 control-label">Apellido</label>
        <div class="col-lg-10">
            <input type="text" th:field="*{apellido}"
                placeholder="Apellido"/>
        </div>
    </div>
    <div class="form-group">
        <label class="col-lg-2 control-label">Celular</label>
        <div class="col-lg-10">
            <input type="text" th:field="*{celular}"
                placeholder="Cargo"/>
        </div>
    </div>
    <div class="form-group">
        <label class="col-lg-2 control-label">Correo</label>
        <div class="col-lg-10">
            <input type="email" th:field="*{correo}"
                placeholder="Correo"/>
        </div>
    </div>

    <div class="form-group">
        <label class="col-lg-2 control-label">Codigo</label>
        <div class="col-lg-10">
            <input type="text" th:field="*{contraseña}"
                placeholder="Correo"/>
        </div>
    </div>
    <div class="form-group">
```

```
        <div class="col-lg-offset-2 col-lg-10">
          <button style="color: black; background-color: green" type="submit"
          class="btn btn-default">GUARDAR</button>
        </div>
      </div>
    </form>

    <h2>Encargado</h2>
    <div class="row">
      <div class="col-md-6">

        <div id="custom-search-input">
          <div class="input-group col-md-12">
            <input id="searchTerm" type="text" onkeyup="doSearch()"
             class="form-control input-lg"
                placeholder="Buscar"/>
            <span class="input-group-btn">
            <button class="btn btn-info btn-lg" type="button">
              <i class="glyphicon glyphicon-search"></i>
            </button>
          </span>
          </div>
        </div>
        <br/>
      </div>
    </div>
    <table class="table" id="regTable" >
      <tr>
        <th>Nombre</th>
        <th>Apellido</th>
        <th>Celular</th>
        <th>Correo</th>
```

```
        </tr>
        <tr th:each="tecnico : ${tecnico}">

          <td th:text="${tecnico.nombre}"></td>
          <td th:text="${tecnico.apellido}"></td>
          <td th:text="${tecnico.celular}"></td>
          <td th:text="${tecnico.correo}"></td>

          <td > <a th:href="@{/deleteTecnico/{id}(id=${tecnico.idTecnico})}"
           style="color:red" >Eliminar</a>  </td>
          <td > <a
         th:href="@{/editTecnico/{id}(id=${tecnico.idTecnico})}">Editar</a>  </td>
        </tr>
      </table>
    </div>
</div>
```

b. Código del modelo

MODELO BIEN INFORMÁTICO

```
package com.ugel.entidades;

import javax.persistence.*;
import java.sql.Date;

@Entity
@Table(name = "bieninformatico", schema = "bdincidenciapractica", catalog = "")
public class BieninformaticoEntity {
    private int idBienInformatico;
    private String codigoBienInformatico;
    private String marca;
    private String modelo;
    private String serie;
```

```
private String descripcion;
private Date fechaingreso;
private int oficinaIdOficina;
private int estadobieninformaticoIdEstadoBienInformatico;
private int tipobieninformaticoIdTipoBienInformatico;

@Id
@Column(name = "id_BienInformatico")
public int getIdBienInformatico() {
    return idBienInformatico;
}

public void setIdBienInformatico(int idBienInformatico) {
    this.idBienInformatico = idBienInformatico;
}

@Basic
@Column(name = "CodigoBienInformatico")
public String getCodigoBienInformatico() {
    return codigoBienInformatico;
}

public void setCodigoBienInformatico(String codigoBienInformatico) {
    this.codigoBienInformatico = codigoBienInformatico;
}

@Basic
@Column(name = "marca")
public String getMarca() {
    return marca;
}

public void setMarca(String marca) {
```

```
    this.marca = marca;
}

@Basic
@Column(name = "modelo")
public String getModelo() {
    return modelo;
}

public void setModelo(String modelo) {
    this.modelo = modelo;
}

@Basic
@Column(name = "serie")
public String getSerie() {
    return serie;
}

public void setSerie(String serie) {
    this.serie = serie;
}

@Basic
@Column(name = "descripcion")
public String getDescripcion() {
    return descripcion;
}

public void setDescripcion(String descripcion) {
    this.descripcion = descripcion;
}
```

```
    @Basic
    @Column(name = "fechaingreso")
    public Date getFechaingreso() {
        return fechaingreso;
    }

    public void setFechaingreso(Date fechaingreso) {
        this.fechaingreso = fechaingreso;
    }

    @Basic
    @Column(name = "oficina_id_Oficina")
    public int getOficinaIdOficina() {
        return oficinaIdOficina;
    }

    public void setOficinaIdOficina(int oficinaIdOficina) {
        this.oficinaIdOficina = oficinaIdOficina;
    }

    @Basic
    @Column(name = "estadobieninformatico_id_EstadoBienInformatico")
    public int getEstadobieninformaticoIdEstadoBienInformatico() {
        return estadobieninformaticoIdEstadoBienInformatico;
    }

    public void setEstadobieninformaticoIdEstadoBienInformatico(int
estadobieninformaticoIdEstadoBienInformatico) {
        this.estadobieninformaticoIdEstadoBienInformatico =
estadobieninformaticoIdEstadoBienInformatico;
    }

    @Basic
```

```
    @Column(name = "tipobieninformatico_id_TipoBienInformatico")
    public int getTipobieninformaticoIdTipoBienInformatico() {
        return tipobieninformaticoIdTipoBienInformatico;
    }

    public void setTipobieninformaticoIdTipoBienInformatico(int
tipobieninformaticoIdTipoBienInformatico) {
        this.tipobieninformaticoIdTipoBienInformatico =
tipobieninformaticoIdTipoBienInformatico;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        BieninformaticoEntity that = (BieninformaticoEntity) o;

        if (idBienInformatico != that.idBienInformatico) return false;
        if (oficinaIdOficina != that.oficinaIdOficina) return false;
        if (estadobieninformaticoIdEstadoBienInformatico !=
       that.estadobieninformaticoIdEstadoBienInformatico)
        return false;
        if (tipobieninformaticoIdTipoBienInformatico !=
         that.tipobieninformaticoIdTipoBienInformatico) return false;
        if (codigoBienInformatico != null ?
         !codigoBienInformatico.equals(that.codigoBienInformatico) :
         that.codigoBienInformatico != null)
         return false;
        if (marca != null ? !marca.equals(that.marca) : that.marca != null) return false;
        if (modelo != null ? !modelo.equals(that.modelo) : that.modelo != null) return
       false;
        if (serie != null ? !serie.equals(that.serie) : that.serie != null) return false;
```

```
        if (descripcion != null ? !descripcion.equals(that.descripcion) : that.descripcion !=
        null) return false;
        if (fechaingreso != null ? !fechaingreso.equals(that.fechaingreso) :
        that.fechaingreso != null) return false;

        return true;
    }

    @Override
    public int hashCode() {
        int result = idBienInformatico;
        result = 31 * result + (codigoBienInformatico != null ?
        codigoBienInformatico.hashCode() : 0);
        result = 31 * result + (marca != null ? marca.hashCode() : 0);
        result = 31 * result + (modelo != null ? modelo.hashCode() : 0);
        result = 31 * result + (serie != null ? serie.hashCode() : 0);
        result = 31 * result + (descripcion != null ? descripcion.hashCode() : 0);
        result = 31 * result + (fechaingreso != null ? fechaingreso.hashCode() : 0);
        result = 31 * result + oficinaIdOficina;
        result = 31 * result + estadobieninformaticoIdEstadoBienInformatico;
        result = 31 * result + tipobieninformaticoIdTipoBienInformatico;
        return result;
    }
}
```

MODELO ESTADO BIEN INFORMÁTICO

```
package com.ugel.entidades;

import javax.persistence.*;
@Entity
@Table(name = "estadobieninformatico", schema = "bdincidenciapractica", catalog =
"")
```

```
public class EstadobieninformaticoEntity {
    private int idEstadoBienInformatico;
    private String denominacion;

    @Id
    @Column(name = "id_EstadoBienInformatico")
    public int getIdEstadoBienInformatico() {
        return idEstadoBienInformatico;
    }

    public void setIdEstadoBienInformatico(int idEstadoBienInformatico) {
        this.idEstadoBienInformatico = idEstadoBienInformatico;
    }

    @Basic
    @Column(name = "denominacion")
    public String getDenominacion() {
        return denominacion;
    }

    public void setDenominacion(String denominacion) {
        this.denominacion = denominacion;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        EstadobieninformaticoEntity that = (EstadobieninformaticoEntity) o;

        if (idEstadoBienInformatico != that.idEstadoBienInformatico) return false;
        if (denominacion != null ? !denominacion.equals(that.denominacion) :
```

```
        that.denominacion != null) return false;
        return true;
    }
    @Override
    public int hashCode() {
        int result = idEstadoBienInformatico;
        result = 31 * result + (denominacion != null ? denominacion.hashCode() : 0);
        return result;
    }
}
```

TIPO BIEN INFORMÁTICO

```
package com.ugel.entidades;

import javax.persistence.*;

@Entity
@Table(name = "tipobieninformatico", schema = "bdincidenciapractica", catalog = "")
public class TipobieninformaticoEntity {
    private int idTipoBienInformatico;
    private String denominacion;

    @Id
    @Column(name = "id_TipoBienInformatico")
    public int getIdTipoBienInformatico() {
        return idTipoBienInformatico;
    }

    public void setIdTipoBienInformatico(int idTipoBienInformatico) {
        this.idTipoBienInformatico = idTipoBienInformatico;
    }

    @Basic
```

```
    @Column(name = "denominacion")
    public String getDenominacion() {
        return denominacion;
    }
    public void setDenominacion(String denominacion) {
        this.denominacion = denominacion;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        TipobieninformaticoEntity that = (TipobieninformaticoEntity) o;

        if (idTipoBienInformatico != that.idTipoBienInformatico) return false;
        if (denominacion != null ? !denominacion.equals(that.denominacion) :
       that.denominacion != null) return false;

        return true;
    }

    @Override
    public int hashCode() {
        int result = idTipoBienInformatico;
        result = 31 * result + (denominacion != null ? denominacion.hashCode() : 0);
        return result;
    }
}
```

OFICINA

```
package com.ugel.entidades;

import javax.persistence.*;
@Entity
@Table(name = "oficina", schema = "bdincidenciapractica", catalog = "")
public class OficinaEntity {
    private int idOficina;
    private String codigoOficina;
    private String nombre;

    @Id
    @Column(name = "id_Oficina")
    public int getIdOficina() {
        return idOficina;
    }

    public void setIdOficina(int idOficina) {
        this.idOficina = idOficina;
    }

    @Basic
    @Column(name = "codigoOficina")
    public String getCodigoOficina() {
        return codigoOficina;
    }

    public void setCodigoOficina(String codigoOficina) {
        this.codigoOficina = codigoOficina;
    }

    @Basic
```

```
    @Column(name = "nombre")
    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        OficinaEntity that = (OficinaEntity) o;

        if (idOficina != that.idOficina) return false;
        if (codigoOficina != null ? !codigoOficina.equals(that.codigoOficina) : that.codigoOficina != null)
            return false;
        if (nombre != null ? !nombre.equals(that.nombre) : that.nombre != null) return false;
        return true;
    }

    @Override
    public int hashCode() {
        int result = idOficina;
        result = 31 * result + (codigoOficina != null ? codigoOficina.hashCode() : 0);
        result = 31 * result + (nombre != null ? nombre.hashCode() : 0);
        return result;
    }
}
```

TÉCNICO

```
package com.ugel.entidades;
import javax.persistence.*;

@Entity
@Table(name = "tecnico", schema = "bdincidenciapractica", catalog = "")
public class TecnicoEntity {
    private int idTecnico;
    private String nombre;
    private String apellido;
    private String celular;
    private String correo;
    private String contraseña;

    @Id
    @Column(name = "id_Tecnico")
    public int getIdTecnico() {
        return idTecnico;
    }

    public void setIdTecnico(int idTecnico) {
        this.idTecnico = idTecnico;
    }

    @Basic
    @Column(name = "nombre")
    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
```

```
@Basic
@Column(name = "apellido")
public String getApellido() {
    return apellido;
}

public void setApellido(String apellido) {
    this.apellido = apellido;
}

@Basic
@Column(name = "celular")
public String getCelular() {
    return celular;
}

public void setCelular(String celular) {
    this.celular = celular;
}

@Basic
@Column(name = "correo")
public String getCorreo() {
    return correo;
}

public void setCorreo(String correo) {
    this.correo = correo;
}

@Basic
@Column(name = "contraseña")
```

```
public String getContraseña() {
    return contraseña;
}

public void setContraseña(String contraseña) {
    this.contraseña = contraseña;
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;

    TecnicoEntity that = (TecnicoEntity) o;

    if (idTecnico != that.idTecnico) return false;
    if (nombre != null ? !nombre.equals(that.nombre) : that.nombre != null) return
   false;
    if (apellido != null ? !apellido.equals(that.apellido) : that.apellido != null) return
   false;
    if (celular != null ? !celular.equals(that.celular) : that.celular != null) return false;
    if (correo != null ? !correo.equals(that.correo) : that.correo != null) return false;
    if (contraseña != null ? !contraseña.equals(that.contraseña) : that.contraseña !=
     null) return false;

    return true;
}

@Override
public int hashCode() {
    int result = idTecnico;
    result = 31 * result + (nombre != null ? nombre.hashCode() : 0);
    result = 31 * result + (apellido != null ? apellido.hashCode() : 0);
```

```
        result = 31 * result + (celular != null ? celular.hashCode() : 0);
        result = 31 * result + (correo != null ? correo.hashCode() : 0);
        result = 31 * result + (contraseña != null ? contraseña.hashCode() : 0);
        return result;
    }
}
```

ENCARGADO

```
package com.ugel.entidades;

import javax.persistence.*;
@Entity
@Table(name = "encargado", schema = "bdincidenciapractica", catalog = "")
public class EncargadoEntity {
    private int idEncargado;
    private String codigoEncargado;
    private String nombre;
    private String apellido;
    private String cargo;
    private String celular;
    private int bieninformaticoIdBienInformatico;

    @Id
    @Column(name = "id_Encargado")
    public int getIdEncargado() {
        return idEncargado;
    }

    public void setIdEncargado(int idEncargado) {
        this.idEncargado = idEncargado;
```

```
}

@Basic
@Column(name = "codigoEncargado")
public String getCodigoEncargado() {
    return codigoEncargado;
}

public void setCodigoEncargado(String codigoEncargado) {
    this.codigoEncargado = codigoEncargado;
}

@Basic
@Column(name = "nombre")
public String getNombre() {
    return nombre;
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}

@Basic
@Column(name = "apellido")
public String getApellido() {
    return apellido;
}

public void setApellido(String apellido) {
    this.apellido = apellido;
}

@Basic
```

```
    @Column(name = "cargo")
    public String getCargo() {
        return cargo;
    }

    public void setCargo(String cargo) {
        this.cargo = cargo;
    }

    @Basic
    @Column(name = "celular")
    public String getCelular() {
        return celular;
    }

    public void setCelular(String celular) {
        this.celular = celular;
    }

    @Basic
    @Column(name = "bieninformatico_id_BienInformatico")
    public int getBieninformaticoIdBienInformatico() {
        return bieninformaticoIdBienInformatico;
    }

    public void setBieninformaticoIdBienInformatico(int
bieninformaticoIdBienInformatico) {
        this.bieninformaticoIdBienInformatico = bieninformaticoIdBienInformatico;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
```

```
        if (o == null || getClass() != o.getClass()) return false;
        EncargadoEntity that = (EncargadoEntity) o;
        if (idEncargado != that.idEncargado) return false;
        if (bieninformaticoIdBienInformatico != that.bieninformaticoIdBienInformatico)
       return false;
        if (codigoEncargado != null ? !codigoEncargado.equals(that.codigoEncargado) :
       that.codigoEncargado != null)
        return false;
        if (nombre != null ? !nombre.equals(that.nombre) : that.nombre != null) return
       false;
        if (apellido != null ? !apellido.equals(that.apellido) : that.apellido != null) return
       false;
        if (cargo != null ? !cargo.equals(that.cargo) : that.cargo != null) return false;
        if (celular != null ? !celular.equals(that.celular) : that.celular != null) return false;
        return true;
    }

    @Override
    public int hashCode() {
        int result = idEncargado;
        result = 31 * result + (codigoEncargado != null ? codigoEncargado.hashCode() : 0);
        result = 31 * result + (nombre != null ? nombre.hashCode() : 0);
        result = 31 * result + (apellido != null ? apellido.hashCode() : 0);
        result = 31 * result + (cargo != null ? cargo.hashCode() : 0);
        result = 31 * result + (celular != null ? celular.hashCode() : 0);
        result = 31 * result + bieninformaticoIdBienInformatico;
        return result;
    }
}
```

c. Código del controlador

EQUIPO INFORMÁTICO

```
@Controller
public class EquipoInformaticoController {

   @Autowired
   @Qualifier("OficinaServiceImpl")
   private OficinaService oficinaService;
   @Autowired
   @Qualifier("EstadoEquipoInformaticoImpl")
   private EstadoBienInformaticoService estadoBienInformaticoService;
   @Autowired
   @Qualifier("tipoBienServiceImpl")
   private com.ugel.services.TipoBienInformaticoService TipoBienInformaticoService;

   @Autowired
   @Qualifier("EquipoInformaticoServiceImpl")
   private EquipoInformaticoService EquipoInformaticoService;

   @GetMapping("/listaEquipo")
   public ModelAndView listaEquipo() {
      ModelAndView mav = new ModelAndView("Equipo");
      //mav.addObject("oficinaIntro2",new OficinaEntity());
      mav.addObject("equipoI",EquipoInformaticoService.listAllEquipo());
      return mav;
   }

   @PostMapping("/saveEquipoInformatico")
   public ModelAndView save(@Valid BieninformaticoEntity bien, BindingResult
    result) {
      if(result.hasErrors()) {
         return addEquipo(bien);
      }
```

```
        EquipoInformaticoService.save(bien);
        return listaEquipo();
    }

    @PreAuthorize("hasRole('ROLE_USER')")
    @GetMapping("/addEquipoInformatico")
     public ModelAndView
     addEquipo(@ModelAttribute("oficina")BieninformaticoEntity bien ){
        ModelAndView mv = new ModelAndView("EquipoAdd");
        mv.addObject("bienIntro", bien);
        mv.addObject("oficina",oficinaService.listAllOficce());
       mv.addObject("tipo",TipoBienInformaticoService.listAllTipoBienInformaticoo());
        mv.addObject("estado",estadoBienInformaticoService.listAllEstadoBienInformaticoo());

        User user = (User)
       SecurityContextHolder.getContext().getAuthentication().getPrincipal();
        mv.addObject("user",user.getUsername());
        return mv;
    }

    @GetMapping("/editBI/{id}")
    public ModelAndView edit(@PathVariable("id") int id) {

        return addEquipo(EquipoInformaticoService.findOne(id));
    }

    @GetMapping("/deleteBI/{id}")
    public ModelAndView delete(@PathVariable("id") int id) {
        EquipoInformaticoService.removeBienInformatico(id);
        return listaEquipo();
    }
}
```

ESTADO BIEN INFORMÁTICO

```
@Controller
public class EstadoBienInformaticoController {
    @Autowired
    @Qualifier("EstadoEquipoInformaticoImpl")
    private EstadoBienInformaticoService estadoBienInformaticoService;

    @GetMapping("/listaEstadoBI")
    public ModelAndView listaEquipo() {
        ModelAndView mav = new ModelAndView("EstadoBienInformatico");

mav.addObject("estadoBI",estadoBienInformaticoService.listAllEstadoBienInformaticoo());
        mav.addObject("estadoBIIngreso", new EstadobieninformaticoEntity());
        return mav;
    }

    @PostMapping("/saveEstadoBI")
    public ModelAndView save(@Valid EstadobieninformaticoEntity estado,
     BindingResult result) {
        if(result.hasErrors()) {
            return addEstado(estado);
        }
        estadoBienInformaticoService.save(estado);
        return listaEquipo();
    }

    @GetMapping("/addEstadoBI")
    public ModelAndView addEstado(@ModelAttribute("oficina")EstadobieninformaticoEntity
     estado ){
        ModelAndView mv = new ModelAndView("EstadoBienInformatico");
        mv.addObject("estadoBIIngreso", estado);
        return mv;
    }
```

```
    @GetMapping("/editEstadoBI/{id}")
    public ModelAndView edit(@PathVariable("id") int id) {

        return addEstado(estadoBienInformaticoService.findOne(id));
    }
    @GetMapping("/deleteEstadoBI/{id}")
    public ModelAndView delete(@PathVariable("id") int id) {
        estadoBienInformaticoService.removeEstadoBienInformatico(id);
        return listaEquipo();
    }
}
```

TIPO BIEN INFORMÁTICO

```
@Controller
public class TipoBienInformaticoController {
    @Autowired
    @Qualifier("tipoBienServiceImpl")
    private TipoBienInformaticoService TipoBienInformaticoService;



    @GetMapping("/listaTipoBI")
    public ModelAndView listaEquipo() {
        ModelAndView mav = new ModelAndView("TipoBienInformatico");
        //mav.addObject("oficinaIntro2",new OficinaEntity());
        mav.addObject("tipoBI",TipoBienInformaticoService.listAllTipoBienInformaticoo());
        mav.addObject("tipoBIIngreso", new TipobieninformaticoEntity());
        return mav;
    }

    @PostMapping("/saveTipoBI")
     public ModelAndView save(@Valid TipobieninformaticoEntity tipo, BindingResult result)
     {
        if(result.hasErrors()) {
```

```
            return addEstado(tipo);
        }
        TipoBienInformaticoService.save(tipo);
        return listaEquipo();
    }

    @GetMapping("/addTipoBI")
    public ModelAndView addEstado(@ModelAttribute("oficina")TipobieninformaticoEntity
     tipo ){
        ModelAndView mv = new ModelAndView("TipoBienInformatico");
        mv.addObject("tipoBIIngreso", tipo);
        return mv;
    }
    @GetMapping("/editTipoBI/{id}")
    public ModelAndView edit(@PathVariable("id") int id) {

        return addEstado(TipoBienInformaticoService.findOne(id));
    }
    @GetMapping("/deleteTipoBI/{id}")
    public ModelAndView delete(@PathVariable("id") int id) {
        TipoBienInformaticoService.removeTipoBienInformatico(id);
        return listaEquipo();
    }
}
```

OFICINA

```
@Controller
public class OficinaController {
    @Autowired
    @Qualifier("OficinaServiceImpl")
    private OficinaService oficinaService;

    @GetMapping("/listaOficina")
    public ModelAndView listaOficina() {
```

```
        ModelAndView mav = new ModelAndView("Oficina");
        //mav.addObject("oficinaIntro2",new OficinaEntity());
        mav.addObject("oficina",oficinaService.listAllOficce());
        return mav;
    }

    @PostMapping("/save")
    public ModelAndView save(@Valid OficinaEntity post, BindingResult result) {
        if(result.hasErrors()) {
            return addOffice(post);
        }
        oficinaService.save(post);
        return listaOficina();
    }

    @GetMapping("/addOffice")
    public ModelAndView addOffice(@ModelAttribute("oficina")OficinaEntity oficina ){
       ModelAndView mv = new ModelAndView("OficinaAdd");
       mv.addObject("oficinaIntro", oficina);
        return mv;
    }

    @GetMapping("/edit/{id}")
    public ModelAndView edit(@PathVariable("id") int id) {
        return addOffice(oficinaService.findOne(id));
    }

    @GetMapping("/delete/{id}")
    public ModelAndView delete(@PathVariable("id") int id) {
        oficinaService.removeOficina(id);
        return listaOficina();
    }
}
```

TÉCNICO

```
@Controller
public class TecnicoController {
    @Autowired
    @Qualifier("TecnicoServiceImpl")
    private TecnicoService TecnicoServiceImpl;

    @GetMapping("/listaTecnico")
    public ModelAndView listaTecnico() {
        ModelAndView mav = new ModelAndView("Tecnico");
        mav.addObject("tecnico",TecnicoServiceImpl.listAllTecnico());
        mav.addObject("tecnicoIngreso", new TecnicoEntity());

        return mav;
    }

    @PostMapping("/saveTecnico")
    public ModelAndView save(@Valid TecnicoEntity tecnico, BindingResult result) {
        if(result.hasErrors()) {
            return addTecnico(tecnico);
        }
        TecnicoServiceImpl.save(tecnico);
        return listaTecnico();
    }

    @GetMapping("/addTecnico")
    public ModelAndView addTecnico(@ModelAttribute("oficina")TecnicoEntity tecnico ){
       ModelAndView mv = new ModelAndView("Tecnico");
       mv.addObject("tecnicoIngreso", tecnico);
        return mv;
    }

    @GetMapping("/editTecnico/{id}")
    public ModelAndView edit(@PathVariable("id") int id) {
```

```
        return addTecnico(TecnicoServiceImpl.findOne(id));
    }

    @GetMapping("/deleteTecnico/{id}")
    public ModelAndView delete(@PathVariable("id") int id) {
        TecnicoServiceImpl.removeTecnico(id);
        return listaTecnico();
    }
}
```

ENCARGADO

```
@Controller
public class EncargadoController {

    @Autowired
    @Qualifier("EncargadoImpl")
    private EncargadoService EncargadoService;

    @Autowired
    @Qualifier("EquipoInformaticoServiceImpl")
    private com.ugel.services.EquipoInformaticoService EquipoInformaticoService;

    @GetMapping("/listaEncargado")
    public ModelAndView listaEncargado() {
        ModelAndView mav = new ModelAndView("Encargado");
        mav.addObject("encargado",EncargadoService.listAllEncargado());
        mav.addObject("encargadoIngreso", new EncargadoEntity());
        mav.addObject("bienInformatico", EquipoInformaticoService.listAllEquipo());
        return mav;
    }

    @PostMapping("/saveEncargado")
    public ModelAndView save(@Valid EncargadoEntity encargado, BindingResult
```

```
    result) {
        if(result.hasErrors()) {
            return addEncargado(encargado);
        }
        EncargadoService.save(encargado);
        return listaEncargado();
    }

    @GetMapping("/addEncargado")
    public ModelAndView addEncargado(@ModelAttribute("oficina")EncargadoEntity
     encargado ){
        ModelAndView mv = new ModelAndView("Encargado");
        mv.addObject("encargadoIngreso", encargado);
        mv.addObject("bienInformatico", EquipoInformaticoService.listAllEquipo());
        return mv;
    }
    @GetMapping("/editEncargado/{id}")
    public ModelAndView edit(@PathVariable("id") int id) {

        return addEncargado(EncargadoService.findOne(id));
    }

    @GetMapping("/deleteEncargado/{id}")
    public ModelAndView delete(@PathVariable("id") int id) {
        EncargadoService.removeEncargado(id);
        return listaEncargado();
    }
}
```

4.3. RESULTADOS

Se propone un modelo referencial, para el desarrollo ágil de aplicaciones en plataforma Java EE, como una alternativa que mejora el tiempo de desarrollo, basado en la adaptabilidad de la infraestructura Java Platform Enterprise Edition, a la arquitectura técnica de la metodología ágil y formal ICONIX aplicado en el caso de estudio: registro de incidencia de bienes informáticos de la UGEL huamanga.

a) La utilización del Mapeo Objeto Relacional Hibernate, permite el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación. Adaptándose de esta manera en la capa de modelo del patron MVC, debido a que crea las entidades orientadas a objetos (las clases) de manera automática, y las actualizara en caso de que haya cambios en la base de datos.

Con la adopción de esta propuesta, ya creadas las entidades, se crea también el tipo de clase JavaBean o también conocido como Bean, el cual viene a ser una clase simple en Java, cuenta con los nombres de sus propiedades y métodos, un constructor, contiene sus atributos como privados y un método setter y getter para cada uno. Mediante estos JavaBeans, se desarrolló el modelo de objetos para la aplicación.

Al trabajar con el Mapeo Objeto Relacional Hibernate (tecnología utilizada en proyectos Java EE), se logra la adpatabilidad, mediante la implementación de componentes JavaBeans que se encuentran en la capa modelo de la arquitectura técnica de la metodología Iconix y la Java EE.

b) La utilización del Framework basado en HTTP y Servlets en la capa Vista, permite proveer la extensión y personalización de las aplicaciones web, primero porque en esta capa se recibe las solicitudes de peticiones HTTP por parte de los clientes, invocando la lógica de negocio adecuada que posteriormente se genera la respuesta que se devuelve al cliente y por otra parte se personaliza cualquier tipo de contenido, que por lo general es HTML, con el cual se crea grillas, combos, menus, etc.

Con la adopción de esta propuesta, se crea los servlets Java, que son clases Java diseñadas para responder a solicitudes HTTP en el contexto de una aplicación web y las JSP, que se puede ver como una extensión de HTML, que le ofrece la capacidad de

incluir fragmentos de código Java dentro de las páginas HTML. Una JSP se convierte a un servlet Java y se ejecuta en el servidor.

Al trabajar con el Framework basado en HTTP y Servlets, se logra la adaptabilidad, mediante la implementación de los componentes Servlet y JSP, que se encuentran en la capa vista de la arquitectura técnica de la metodología Iconix y la Java EE.

c) La utilización de las Clases Controladoras del Spring MVC, en la capa de Controlador de la aplicación , permite interpretar la entrada del usuario y transforma el resultado en un modelo que se mostrará al usuario en la Vista, mediante la anotación @Controller.

Con la adopción de esta propuesta, se crea los @Controller para una petición de un fichero .htm, luego para decidir exactamente a qué método de nuestro controlador queremos que se llame en función de la página htm pedida. Para ello utilizamos un método cualquiera en la clase con la anotacion @RequestMapping.

Al trabajar con las clases controladoras del Spring MVC, se logra la adaptabilidad, mediante la implementación de los componentes @conroller y @RequestMapping, que se encuentran en la capa controlador de la arquitectura técnica de la metodología Iconix y la Java EE.

4.4. ANALISIS DE RESULTADOS

En base a la adecuada aplicación de la tectnologías utilizadas en proyectos Java EE, se logra la adaptabilidad de la infraestructura Java Platform Enterprise Edition a la arquitectura técnica de la metodología ágil y formal Iconix, caso de estudio; registro de incidencias de bienes informáticos de la UGEL Huamanga, mediante la igualdad y similitud de componestes adoptados en la raquitectura técnica de la metodología y la Java EE. En todo el Capítulo IV análisis y resultados, se concluye que el Mapeo Objeto Relacional Hibernate, Framework basado en HTTP y Servlets y las clases Controladoras de Spring MVC, son mecanismos de adaptabilidad a la Arquitectura Técnica de la Metodología Ágil y Formal ICONIX.

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

5.1. CONCLUSIONES

a. De lo establecido en el numeral 4.3), del Capítulo IV - Resultados, se afirma, que la infraestructura Java Platform Enterprise Edition es un mecanismo de adaptabilidad a la Arquitectura Técnica de la Metodología Ágil y Formal ICONIX, mediante la implementación de componentes en cada capa de aplicación, facilitando el desarrollo ágil del software.

b. De lo establecido en el numeral 4.3), del Capítulo IV - Resultados, se afirma, que el Mapeo Objeto Relacional Hibernate es un mecanismo de adaptabilidad a la Arquitectura Técnica de la Metodología Ágil y Formal ICONIX, mediante la implementación de componentes en la capa modelo, facilitando el desarrollo ágil del software.

c. De lo establecido en el numeral 4.3), del Capítulo IV - Resultados, se afirma, que el Framework basado en HTTP y Servlets es un mecanismo de adaptabilidad a la Arquitectura Técnica de la Metodología Ágil y Formal ICONIX, mediante la implementación de componentes en lada capa vista, facilitando el desarrollo ágil del software.

d. De lo establecido en el numeral 4.3), del Capítulo IV - Resultados, se afirma, que las clases Controladoras de Spring MVC es un mecanismo de adaptabilidad a la Arquitectura Técnica de la Metodología Ágil y Formal ICONIX, mediante la implementación de componentes en la capa controlador, facilitando el desarrollo ágil del software.

5.2. RECOMENDACIONES

a. Es recomendable utilizar la Plataforma Java Enterprise Edition, porque proporcionando una infraestructura basada en estándares para el desarrollo ágil de aplicaciones Java distribuidas, de múltiples capas y basadas en componentes, además se puede implantar en diversos sistemas operativos, haciendo que las aplicaciones lleguen al mercado lo más pronto posible.

b. En el presente trabajo de investigación, respecto a la arquitectura Java EE, describe las tecnologías más esenciales de esta plataforma, una recomendación fundamental para la utilización de este trabajo de investigación como guía, es que el lector tenga una base de conocimiento acerca del lenguaje programación Java, porque que dicho lenguaje, viene hacer el núcleo de la arquitectura Java EE.

c. Se recomienda utilizar el presente modelo referencial, sobre el desarrollo de softwrare, dado en el presente trabajo de investigación, porque permite el desarrollo ágil de software, en proyectos de cualquier tamaño, mientras que el desarrollo de nuevos proyectos sigan inmersos en metodologías ágiles.

d. Se recomienda a la comunidad de software libre y programadores java, aplicar el marco de trabajo de código abierto java, para el desarrollo de aplicaciones web, mientras todo en lo que concierne a Java, siga en auge a nivel mundial.

e. Se recomienda contruir una herramienta automatizada de generación de código para la arquitectura técnica de la metodología agil y formal ICONIX, por ser una metodología ligera de desarrollo de software.

f. Se recomienda realizar la investigación de adaptabilidad de un entorno libre de código abierto, a la metodología agil y formal ICONIX, por ser una metodología pesado ligera de desarrollo de software.

BIBLIOGRAFÍA

Acosta, M. (2013). *"Estudio de patrones de diseño en Plataforma Java Enterprise Edition versión 6 para el desarrollo de aplicaciones web",* Tesis para optar el título de Ingeniero en Sistemas Computacionales. Universidad Técnica Del Norte. Ibarra, Ecuador.

Antonio, J. (2001). *El gran libro del protocolo.* Madrid, España: Autor.

Aumaille B. (2002), *J2EE Desarrollo de aplicaciones Web*, INFORMÁTICA TÉCNICA, España, edicion Francesc GARCIA.

Arias, F. (2006). Mitos y errores en la elaboración de tesis y proyectos de investigación. (3ra Ed.). Caracas: Episteme.

Ávila J. (2016). UF2406. El ciclo de vida del desarrollo de aplicaciones. 5ta edición. España. ELEARNING S.L.

Bavaresco, A. (2006). Proceso Metodológico en la Investigación. *Cómo hacer un diseño de investigación*. Maracaibo: La Universidad de Zulia.

Balestrini M. (2006). Como se elabora el proyecto de investigación. Consultores asociados BL. Caracas, República Bolivariana de Venezuela. Séptima edición.

Bauer, C. King, G. y Gregory, G. (2015). *Java Persistence with Hibernate*. (2da Ed.). MANNIG.

Bernal C. (2010). Metodología de la investigación. Administración, economía, humanidades y ciencias sociales. (3ra Ed.).

Carrasco, S. (2005). Metodología de la Investigación Científica. (1ra Ed.). Lima: San Marcos.

Ceballos J. (2015). Java TM. Interfaces gráficas y aplicaciones para Internet (4ta Ed.). España. Ra-Ma.

Chávez, N. (2007). Introducción a la Investigación Educativa. Maracaibo, Venesuela.

Conociendo una herramienta ORM (2013). Hibernate Recuperado el día 16 de agosto de 2019 desde: http://www.analyticaweb.com/desarrollo-web/conociendo-una-herramienta-orm hibernate.

Colección Esencial (2011). *Esencial Internet Explorer 9.* Cataluña, España: Editions ENI.

Deitel, H. y Deitel, P. (2003). *Cómo programar en C++ (4ª Ed.).* Juarez, Mexico: Pearson.

Del Busto, H. y Enriquez, O. (2012). *Mapeo objeto/relacional (orm)*. Revista Telem@ tica, 10(3):1–7.

ECURED. (2015). ECURED. Obtenido de https://www.ecured.cu/Framework.

Fadatare, R (2018). Guías de Java: mapeo de hibernate orm: Recuperado el día 18 de agosto de 2019 desde https://www.javaguides.net/2018/11/hibernate-hello-world-tutorial.html:

Flores, J. y Acuña, C. (2014). *Método de las 6´D Modelamiento – Algoritmo - Programación*: Lima Perú: MACRO.

García A. (2007). Fiscalidad Internacional De Los Cánones: Derechos De Autor, Propiedad Industrial Y Know- How. (1ra Ed.). España. LEX NOVA.

Galindo J. y Camps J. (2008). Diseño e implementación de un marco de trabajo (framework) de presentación para aplicaciones JEE.

Gutiérrez, J. (2008). *Laboratorio de Redes Neuronales.* Armenia, Colombia: Conceptos Gráficos.

Gutiérrez, J. y Tena, J. (2003). *Protocolos criptográficos y seguridad en redes.* Cantabria, España: Gráficas Calima

Gupta, A. (2013). *Java EE 7 Essentials*: *Enterprise Developer Handbook* (1ra Ed.). Estados Unidos de América: O'Reilly.

Hemrajani, A. (2006). *Agile Java Development: with Spring, Hibernate and Eclipse:* (1ra Ed.). *USA: Sams Publishing.*

Hernandez S. et al. (2010), Metodología de la Investigación Científica (5ta Ed.). *McGRAW-HILL* / INTERAMERICANA EDITORES, S.A. DE C.V. México D.F.

Hernández S. (2013). Maestria en tecnología Educativa,

Hernández R., Baptista P. et al. (2014). Metodología de la Investigación. *Alcance de la investigación.* (6 ed., págs. 88-101). México: McGraw-Hill.

Jendrock, E. et al. (2010). *The Java EE 5 Tutorial*: oracle.

Jendrock, E. et al. (2013). *The Java EE 6 Tutorial*: *Advanced Topics (*4ta Ed.). oracle.

Johnson, R. et al. (2010). *Professional Java Development with the Spring Framework*. (1ra Ed.). Wrox Press.

Joyanes, L. (2008). *Fundamentos de programación.* Madrid: España. McGRAWHill/Interamericana De España.

Luján, S. (2001). Programación en internet: Clientes web. Alicante. España: Club Universitario

Medina L. y López W. (2015). Revista Escoger Una Metodología Para Desarrollar Software, Difícil Decisión, Fundación Universitaria Los Libertadores, Bogotá.

Ordax J. y Ocaña P. (2012). Programación web en Java*. AULA MENTOR.* Ministerio de Educación de España. España.

Perrone, P. et al. (2003). *J2EE Developer's Handbook: Developer's Library.* Estados Unidos: Sams Publishing.

Porras, E. (2011). La Metodología Ágil y Formal ICONIX para el Desarrollo de Software: Teoría y Práctica. Ayacucho, Perú: Ami Ayacucho.

Que es Java EE (2018). *Java Empresarial*. Recuperado el día 13 de agosto de 2019 desde https://javaempresarial.com/que-es-java-ee/.

Rashmi, D. (2014). J2EE Made Easy. (1ra Ed.). India: VIKAS.

Rios, S. (2015). Java Revolutions: JSF 2 + Hibernate 4 + Spring 4: PrimeFaces 5 with JAX-WS y EJB'S.

Rodríguez D. (2012). Diseño e implementación de un Framework de Presentación. *Estudios de Informática y Multimedia*.

Romero, L. (1997). *Publicar en Internet: guía práctica para la creación de documentos HTML.* Cantabria, España: Universidad de Cantabria.

Rosenberg, D., Stephens, M. (2007). Use Case Driven Object Modeling with UML: Theory and Practice. New York, USA: Apress.

Rosenberg, D., Stephens, M., Collins-Cope, M. (2005). *Agile development with ICONIX Process: People, process, and pragmatism.* New York, USA: Apress.

Supo, J. (2012). *Seminario de Investigación Científica*. Arequipa.

Tamayo, M. y Tamayo, A. (1997). El Proceso de la Investigación Científica. México: LIMUSA. (3 ra edición).

Talledo, J. (2015). Implantación de aplicaciones web en entorno internet, intranet y extranet: MF0493_3: España: Paraninfo.

Thibaud, C. (2006). Mysql 5, Instalacion, Implementacion, Administracion, Programacion. Barcelona: Editions ENI.

Virvou, M. y Matsuura, S. (2012). *Knowledge-based software engineering: Proceedings of the Tenth Joint Conference on Knowledge-based software engineering.* Amsterdam, Netherlands: IOS Press.

Yarcuri, S. (2018). "*Adaptabilidad del Patrón MVC del Framework .Net a la Arquitectura Técnica de la Metodología Ágil y Formal ICONIX. Caso de estudio: Registro de Estudios Ambientales del DREMA".* Tesis para optar el título de Iingeniero en Sistemas.Universidad Nacinal de San Cristobal de Huamanga. Ayacucho, Perú.

ACTA DE SUSTENTACIÓN DE TESIS N° 040-2020-FIMGC

En la ciudad de Ayacucho, en cumplimiento a la **Resolución Decanal N° 273-2020-FIMGC-D,** siendo treinta del mes de octubre del 2020, a horas 11.00 a.m.; se reunieron los jurados del acto de sustentación, en el Auditórium virtual google meet del Campus Universitario de la Universidad Nacional de San Cristóbal de Huamanga.

Siendo el Jurado de la sustentación de tesis compuesto por el Presidente el, **Mg. Ing. Manuel Avelino LAGOS BARZOLA,** Jurado el **Ing. José Antonio GUERRERO HINOSTROZA,** Jurado – Asesor el **Mg. Ing. Hubner JANAMPA PATILLA,** y Secretario del proceso **Ing. Christian LEZAMA CUELLAR,** con el objetivo de recepcionar la sustentación de la tesis denominada **"ADAPTABILIDAD DE LA INFRAESTRUCTURA JAVA PLATFORM ENTERPRISE EDITION A LA ARQUITECTURA TÉCNICA DE LA METODOLOGÍA ÁGIL Y FORMAL ICONIX 2019. CASO DE ESTUDIO: REGISTRO DE INCIDENCIA DE BIENES INFORMÁTICOS DE LA UGEL HUAMANGA, 2019",** sustentado por la Bach. **Thalia Betty ATAUJE PUMALLIHUA,** bachiller **en Ingeniería de Sistemas.**

El Jurado luego de haber recepcionado la sustentación de la tesis y realizado las preguntas, el sustentante al haber dado respuesta a las preguntas, y el Jurado haber deliberado; califica con la nota aprobatoria de **16 (Dieciséis).**

En fe de lo cual, se firma la presente acta, por los miembros integrantes del proceso de sustentación.

Mg. Ing. Manuel Avelino LAGOS BARZOLA
Presidente

Mg. Ing. Hubner JANAMPA PATILLA
Jurado - Asesor

Ing. José Antonio GUERRERO HINOSTROZA
Jurado

Ing. Christian LEZAMA CUELLAR
Secretario del Proceso

c.c.:
Bach. Thalia Betty ATAUJE PUMALLIHUA
Jurados (4)
Archivo

UNSCH FACULTAD DE INGENIERÍA DE MINAS, GEOLOGÍA Y CIVIL

CONSTANCIA DE ORIGINALIDAD DE TRABAJO DE INVESTIGACIÓN

El que suscribe; responsable verificador de originalidad de trabajos de tesis de pregrado en segunda instancia para las Escuelas Profesionales de la Facultad de Ingeniería de Minas, Geología y Civil; en cumplimiento a la Resolución de Consejo Universitario N° 039-2021-UNSCH-CU, Reglamento de Originalidad de Trabajos de Investigación de la UNSCH y Resolución Decanal N° 158-2021-FIMGC-UNSCH-D, deja constancia que:

- Apellidos y Nombres del Bach. : Atauje Pumallihua Thalia Betty
- Escuela Profesional : Ingeniería De Sistemas
- Título de la Tesis : Adaptabilidad de la infraestructura Java Platform Enterprise Edition a la arquitectura técnica de la metodología ágil y formal Iconix 2019. Caso de estudio: registro de incidencia de bienes informáticos de la Ugel Huamanga, 2019.
- Evaluación de la originalidad : 25 % de similitud

Por tanto, según los artículos 12, 13 y 17 del Reglamento de Originalidad de Trabajos de Investigación, ***es procedente otorgar la constancia de originalidad*** para los fines que crea conveniente.

Ayacucho, 28 de junio del 2021

Firmado digitalmente por Mg. Ing. Johnny Henrry Ccatamayo Barrios
Fecha: 2021.06.28 19:38:18 -05'00'

Mg. Ing. Ccatamayo Barrios Johnny Henrry
Verificador de originalidad de trabajos de tesis de pregrado de la FIMGC

Numero de constancia: 072-2021-FIMGC.

FACULTAD DE INGENIERÍA
DE MINAS, GEOLOGÍA Y CIVIL
Av. Independencia S/N
Ciudad Universitaria
Central Tel 066 312510
Anexo 151

ADAPTABILIDAD DE LA INFRAESTRUCTURA JAVA PLATFORM ENTERPRISE EDITION A LA ARQUITECTURA TÉCNICA DE LA METODOLOGÍA ÁGIL Y FORMAL ICONIX 2019. CASO DE ESTUDIO: REGISTRO DE INCIDENCIA DE BIENES INFORMÁTIC

Fecha de entrega: 28-jun-2021 06:03p.m. (UTC-0500)
Identificador de la entrega: 1613499127
Nombre del archivo: o_de_Investigacion_Susteantado_THALIA_ATAUJE_PUMALLIHUA_EPIS.doc (5.21M)
Total de palabras: 19633
Total de caracteres: 127438

por Thalia Betty Atauje Pumallihua

ADAPTABILIDAD DE LA INFRAESTRUCTURA JAVA PLATFORM ENTERPRISE EDITION A LA ARQUITECTURA TÉCNICA DE LA METODOLOGÍA ÁGIL Y FORMAL ICONIX 2019. CASO DE ESTUDIO: REGISTRO DE INCIDENCIA DE BIENES INFORMÁTIC

INFORME DE ORIGINALIDAD

25%	24%	4%	13%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	Submitted to Universidad Nacional de San Cristóbal de Huamanga Trabajo del estudiante	5%
2	repositorio.unsch.edu.pe Fuente de Internet	2%
3	cybertesis.uni.edu.pe Fuente de Internet	1%
4	docplayer.es Fuente de Internet	1%
5	www.effectivetrainings.de Fuente de Internet	1%
6	www.scribd.com Fuente de Internet	1%
7	www.matera.com Fuente de Internet	1%

docslide.us

8	Fuente de Internet	1 %
9	iskudarnov.ru Fuente de Internet	1 %
10	repositorio.unfv.edu.pe Fuente de Internet	1 %
11	www.lisenme.com Fuente de Internet	1 %
12	expertojava.ua.es Fuente de Internet	1 %
13	dspace.unach.edu.ec Fuente de Internet	1 %
14	Submitted to Universidad Cesar Vallejo Trabajo del estudiante	1 %
15	git.cs.uni-paderborn.de Fuente de Internet	1 %
16	Submitted to University of Johannsburg Trabajo del estudiante	<1 %
17	Submitted to tsi Trabajo del estudiante	<1 %
18	bibing.us.es Fuente de Internet	<1 %
19	phpcondw.wordpress.com Fuente de Internet	<1 %

20	repositorio.utn.edu.ec Fuente de Internet	<1 %
21	Submitted to tec Trabajo del estudiante	<1 %
22	Submitted to Prince Sultan University Trabajo del estudiante	<1 %
23	www.dspace.uce.edu.ec Fuente de Internet	<1 %
24	www.djcxy.com Fuente de Internet	<1 %
25	www.gusdya.id Fuente de Internet	<1 %
26	Submitted to Politehnica University of Timisoara Trabajo del estudiante	<1 %
27	idoc.pub Fuente de Internet	<1 %
28	carlospesquera.com Fuente de Internet	<1 %
29	dispositivosmivilesuni5tap.blogspot.com Fuente de Internet	<1 %
30	1library.co Fuente de Internet	<1 %
31	repositorio.upao.edu.pe	

	Fuente de Internet	<1 %
32	Submitted to University of Wales Swansea Trabajo del estudiante	<1 %
33	repositoriotec.tec.ac.cr Fuente de Internet	<1 %
34	dspace.utb.edu.ec Fuente de Internet	<1 %
35	gitlab.fdmci.hva.nl Fuente de Internet	<1 %
36	id.123dok.com Fuente de Internet	<1 %
37	repositorio.ucv.edu.pe Fuente de Internet	<1 %
38	proxyinverso.blogspot.com Fuente de Internet	<1 %
39	Submitted to University of Greenwich Trabajo del estudiante	<1 %
40	hdl.handle.net Fuente de Internet	<1 %
41	repositorio.uns.edu.pe Fuente de Internet	<1 %

Excluir citas	Activo	Excluir coincidencias	< 30 words
Excluir bibliografía	Activo		