IMPLEMENTACIÓN DE UN PROTOTIPO EN SOFTWARE LIBRE PARA LA GESTION DE PRESTAMO DE MATERIALES EN UNA INSTITUCIÓN EDUCATIVA, CASO DE ESTUDIO (UNIVERSIDAD DE CARTAGENA).

Autor

EVER JESÚS BLANCO MEZA

Trabajo de grado para optar el título de Magister en Software Libre

Director

Msc. FREDDY MENDEZ ORTIZ

Asesor:

Doctor. DIOFANOR ACEVEDO CORREA

UNIVERSIDAD AUTONOMA DE BUCARAMANGA
FACULTAD DE INGENIERÍA DE SISTEMAS
MAESTRIA EN SOFTWARE LIBRE
GRUPO DE INVESTIGACION PRISMA
2014

DEDICATORIA

A mis amigos, por confiar en mí, por el apoyo incondicional y por todos los momentos vividos.

A todas las personas que hicieron parte de mí formación en los estudios de maestría.

Al profesor FREDY MENDEZ por su apoyo, asesoría, dirección y por la confianza depositada en mí para la realización y culminación de este proyecto.

TABLA DE CONTENIDO.

RESUMEN

El presente trabajo describe el análisis, diseño e implementación de una herramienta de software que brinda asignaciones para la gestión de préstamos de materiales en los distintos laboratorios de una Institución de Educación Superior, para nuestro caso, la universidad de Cartagena. Dentro del desarrollo se determinan los tiempos utilizados bajo la metodología del prototipo evolutivo el cual parte de la implementación inicial donde el usuario interviene con sus comentarios generándose así un proceso de refinamiento a través de las diferentes versiones hasta llegar a un sistema adecuado. Se presentan los resultados del desarrollo del proyecto, los cuales comprenden la implementación del software: Validando la calidad del mismo basándonos en la Norma ISO/IEC 25010–Modelo de calidad (2005) en las características de funcionabilidad y usabilidad, el ultimo criterio soportado bajo la encuesta de una muestra de 8 usuarios extraída de una población de 450 estudiantes. Hubo cumplimiento con las necesidades establecidas en los requerimientos del producto.

El sistema **SRED** fue construido con las siguientes tecnologías usando el patrón arquitectónico de diseño MVC:

- HTML 5
- CSS 3
- JAVASCRIPT
- PHP
- POSTGRES SQL 9.3

Gracias a las anteriores características en la implementación de la solución SRED se garantiza la.

- **Portabilidad:** Para usar la solución solo necesitas conexión a internet y un navegador web, independientemente del sistema operativo incluyendo también S.O móviles.

- **Rentabilidad:** Las tecnologías usadas en SRED son de libre distribución y de desarrollo maduro en el mercado garantizando así la reducción en el costo de desarrollo e implementación a corto, mediano y largo plazo.

- **Escalabilidad:** Para que el software escale de manera eficiente solo se necesita agregar recursos de hardware a la maquina servidor que en este caso es un VPS (Virtual Prívate Server) sin necesidad de detener los servicios de la aplicación.

- **Mantenibilidad:** El desarrollo modular de la aplicación basado en programación orientada a objetos (POO) garantiza facilidad al momento de Detectar, Corregir y aplicar posibles correcciones y/o actualizaciones de funcionalidad al sistema.

- **Estabilidad:** La solución se encuentra albergada por un servidor VPS (Virtual Prívate Server) de buen rendimiento el cual se encuentra activo las 24/7 durante todo el año, evitando percances que podrían generar la intermitencia en el servicio.

INTRODUCCION

A lo largo de la historia, los laboratorios en las universidades siempre han necesitado herramientas que le permitan administrar su información de una manera eficiente. Algunos laboratorios en las universidades utilizan herramientas como sistemas de información de gran capacidad, pero estas no son de uso libre, también hay algunos que utilizan herramientas más sencillas, como hojas de cálculo, o incluso planillas o cuadernos.

"*Durante las últimas décadas, las universidades colombianas han optado por dos opciones: cuando la universidad tiene suficientes recursos escogen generalmente la compra de un software comercial que tenga funcionalidades que se adapten a sus necesidades, de otro lado, están las universidades que gestionan el préstamo de materiales de laboratorio de forma rudimentaria con hojas de cálculo*.

Actualmente no existen aplicaciones de software libre, con funcionalidades específicas para la gestión de préstamos de materiales en los distintos laboratorios de una universidad. Se busca disponer de una herramienta que permita tener mayor eficiencia en las labores antes mencionadas. Atendiendo a esto último, es que se realizó la IMPLEMENTACIÓN DE UN PROTOTIPO EN SOFTWARE LIBRE PARA LA GESTION DE PRESTAMO DE MATERIALES en la Universidad de Cartagena.

1. PRESENTACIÓN DEL PROYECTO

1.1. ANTECEDENTES

El proceso de asignación de recursos como: Salones, materiales de laboratorios, recursos multimedios es una de las principales actividades que se lleva a cabo en las distintas universidades, en muchas de estas instituciones estos procesos se llevan de forma manual y en algunos casos se encuentran sistematizados a medias a pesar de que hoy día se cuenta con herramientas tecnológicas que podrían utilizarse para implementar una solución efectiva para tal situación.

1.2. PRODUCTOS DE SOFTWARE LIBRE PARA LA GESTIÓN DE PRÉSTAMOS

1.2.1. TABLIX

- Sitio web: http://www.tablix.org
- Ultima actualización: ultimo release estable julio 2007
- Interfaz de usuario: línea de comando
- Lenguaje de programación: C
- Almacenamiento de datos: archivo XML
- Licencia: GPL
- Sistema Operativo: Unix/Linux (u otros compilando source)
- Traducciones: Ingles

Descripción: Se trata de un programa de línea de comando, especializado en aplicar algoritmos de resolución de horarios. Utiliza archivos XML como entrada y salida. Menciona un frontend grafico (GTK) del cual no hay más información, de todos modos lo menciona como una aplicación simple de horarios de escuela, enfocada solamente en la creación de horarios.

1.2.2. OPENCTT

- Sitio web: http://www.openctt.org/
- Sitio web alternativo: http://sourceforge.net/projects/openctt/
- Última actualización: ultimo release marzo 2008
- Estado (según sourceforge): prealpha
- Lenguaje de programación: C#
- Almacenamiento de datos: archivo con formato propietario.
- Licencia: GPL

Sistema Operativo: MS Windows 32 bit

- Traducciones: Ingles, Croata

Es un sistema que brinda poca información sobre su desarrollo, como manual de usuario y manual técnico.

1.2.3. TIMEFINDER

- Sitio web: http://timefinder.sourceforge.net/
- Última actualización: noviembre 2008
- Interfaz de usuario: aplicación Desktop (swing)
- Lenguaje de programación: Java
- Almacenamiento de datos: Archivo XML
- Licencia: Apache Licence
- Sistema Operativo: varios (Java)
- Traducciones: Inglés, Alemán

No maneja una estructura de cursos compatible con asignaturas/grupos/cursos, solamente maneja "eventos" que se reparten en una semana.
Utiliza (rangos 8 a 10, 10 a 12, etc), no adecuado considerando la realidad de Facultad de Ingeniería, de distintas horas de comienzo de clase.

1.2.4. VISUAL TIMETABLER

- Sitio web: http://visual.timetabling.free.fr
- Última actualización: septiembre 2010
- Interfaz de usuario: aplicación Desktop
- Lenguaje de programación: sin datos
- Almacenamiento de datos: MySQL

- Licencia: Freeware
- Sistema Operativo: MS Windows
- Traducciones: Frances

Posee versión monousuario y versión en red (varias instancias de la aplicación de escritorio que acceden a base de datos central).

El sitio web, la documentación y el software están en francés (no encontramos información en inglés o español), esto dificulta la posibilidad de evaluación, y lo muestra como un producto muy local a su ámbito regional.

1.2.5. FET

- Sitio web: http://www.lalescu.ro/liviu/fet/
- Ultima actualización: noviembre 2010
- Interfaz de usuario: aplicación Desktop (gtk)
- Lenguaje de programación: C++
- Almacenamiento de datos: Archivo XML
- Licencia: GPL
- Sistema Operativo: GNU-Linux, MS Windows
- Traducciones: varios idiomas, incluyendo inglés y español.

FET realiza solo el horario de cursos, y se enfoca en el grupo de estudiantes, y en el horario para el docente. Por esto lo veo como poco adecuado para horario de Universidad (en el manual tiene ejemplos de escuela primaria y secundaria esta como un pendiente el ejemplo de universidad).

Modelo de datos: se basa en el concepto de ACTIVIDAD. Una actividad puede definirse por la tupla (grupo estudiantes, materia, docente).

Asignación de salones: no siempre lo hace, en el manual supone que pocas veces es necesario. Asume el concepto de “home room” (de grupo de estudiantes o de docente), donde un grupo está siempre en el mismo salón.

1.2.6. UNITIME

- Sitio web: http://www.unitime.org
- Ultima actualización: febrero 2011
- Interfaz de usuario: aplicación Web
- Lenguaje de programación: Java
- Almacenamiento de datos: Hibernate como ORM y admite al menos los DBMS Mysql y Oracle
- Licencia: GPL
- Sistema Operativo: Servidor GNU/Linux y otros, clientes solo necesitan un navegador.
- Traducciones: Ingles

Unitime es un sistema creado inicialmente en la universidad de Praga, como una tesis de PhD por Tomas Muller, luego continuó su desarrollo en la Universidad de Purdue (USA), donde se utiliza actualmente para la gestión de una importante cantidad de cursos y exámenes.

Gestión de cuadros horarios de cursos: Maneja la información de carreras, cursos, asignaturas, grupos dentro de la asignatura y clases. A cada grupo se puede asignar horario y salón (también profesor) en forma manual o automática (para la asignación automática se pueden ingresar previamente las preferencias o restricciones horarias).

Gestión de cuadros horarios de exámenes: Permite ingresar las instancias de exámenes de cada asignatura, y asignarles horario y salón.

Gestión de eventos (uno o varios días): Permite la gestión de eventos, de uno o varios días, consultando la disponibilidad de salas, y permitiendo que la reserva pueda ser hecha (vía web) por un usuario no administrador (por ejemplo un docente), quedando esta reserva pendiente de aprobación por el administrador.

1.3. PLANTEAMIENTO DEL PROBLEMA Y JUSTIFICACION

En el estudio de los sistemas de software libre desarrollados para la gestión de préstamos de recursos no encontramos un aplicativo propiamente desarrollado para la gestión de préstamos de materiales, algunos de los sistemas encontrados son simples aplicaciones para escuelas y no para universidades como lo habíamos planteado, otros proyectos aunque más avanzados son poco flexibles como por ejemplo manejan la asignación en rango de horarios, otros están orientados o reservas de salones para actividades casuales como eventos, en general se hace necesario la implementación de un sistema que gestione el préstamo de cualquier tipo de material en instituciones de educación superior u universidades.

1.4. OBJETIVOS

1.4.1. OBJETIVO GENERAL

Desarrollar un sistema de gestión de préstamo de materiales a nivel de prototipo haciendo uso de herramientas de software libre que mejoren los procesos internos en las Instituciones de Educación Superior (Universidades).

1.4.2. OBJETIVOS ESPECÍFICOS

- Caracterización de las principales herramientas de software libre que sirven de apoyo en los procesos de gestión de préstamo de materiales en instituciones de educación superior.

- Definir las especificaciones mínimas del sistema de gestión de préstamo de materiales en instituciones de educación superior.
- Implementación del prototipo haciendo uso de tecnologías como Postgresql y PHP.
- Validar la funcionalidad del prototipo en al menos una dependencia de la institución.

2. MARCO TEÓRICO.

2.1. ARQUITECTURA MODELO VISTA CONTROLADOR.

MVC (por sus siglas en inglés) es un patrón de diseño de arquitectura de software usado principalmente en aplicaciones que manejan gran cantidad de datos y transacciones complejas donde se requiere una mejor separación de conceptos para que el desarrollo esté estructurado de una mejor manera, facilitando la programación en diferentes capas de manera paralela e independiente. MVC sugiere la separación del software en 3 estratos: Modelo, Vista y Controlador.

Modelo: Es la representación de la información que maneja la aplicación. El modelo en sí son los datos puros que puestos en contexto del sistema proveen de información al usuario o a la aplicación misma.

Vista: Es la representación del modelo en forma gráfica disponible para la interacción con el usuario. En el caso de una aplicación Web, la “Vista” es una página HTML con contenido dinámico sobre el cuál el usuario puede realizar operaciones.

Controlador: Es la capa encargada de manejar y responder las solicitudes del usuario, procesando la información necesaria y modificando el Modelo en caso de ser necesario.

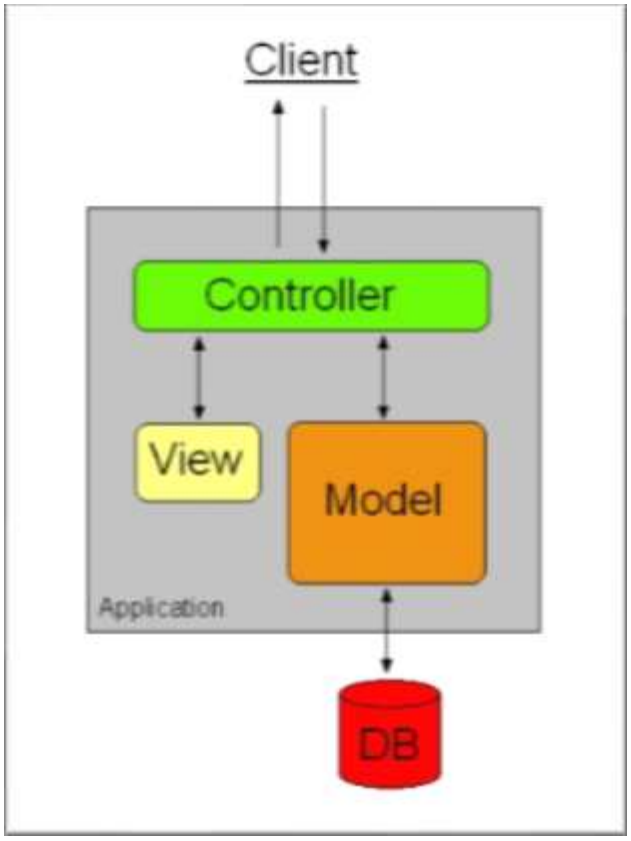

Figura 1. Arquitectura Modelo Vista Controlador.

Fuente. Bajada de internet.

2.1.1. CICLO DE VIDA DE LA ARQUITECTURA MODELO VISTA CONTROLADOR.

El ciclo de vida de MVC es normalmente representado por las 3 capas presentadas anteriormente y el cliente (también conocido como usuario). El siguiente diagrama representa el ciclo de vida de manera sencilla:

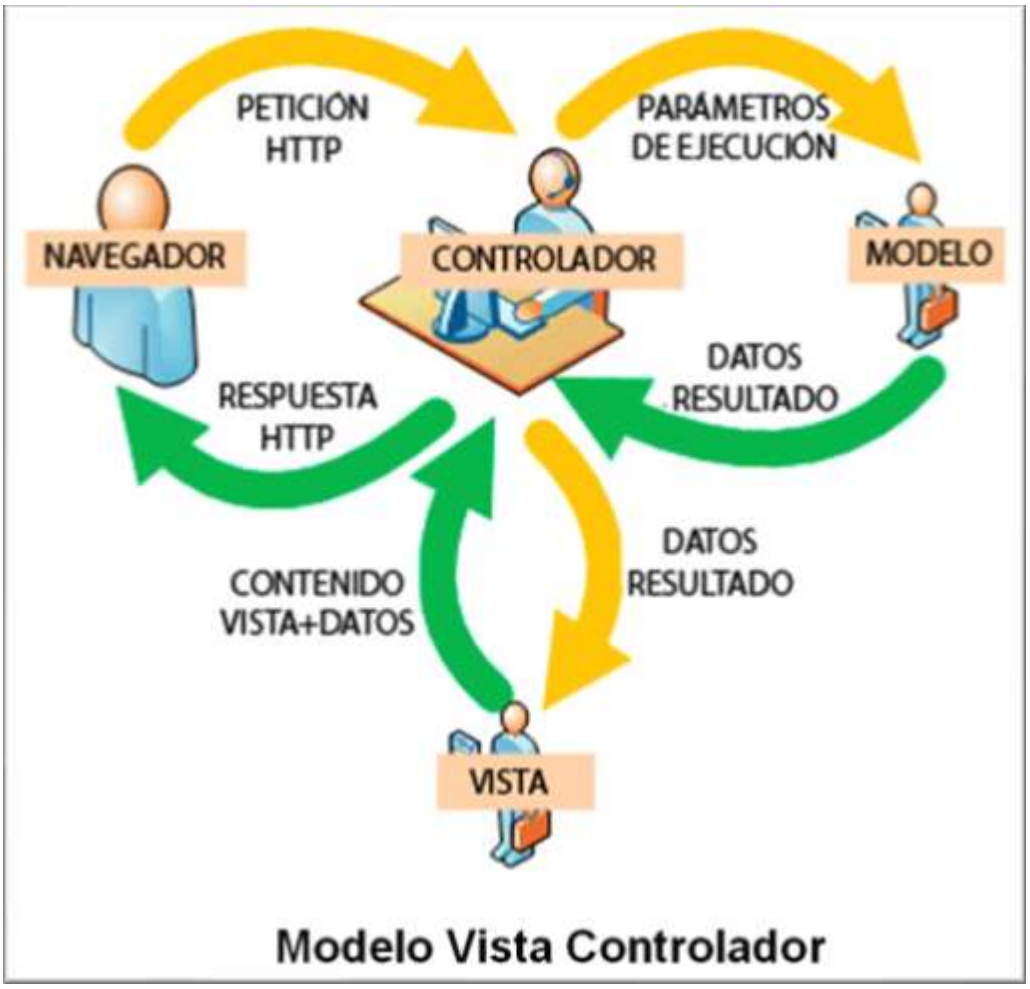

Figura 2. Ciclo de Vida Arquitectura Modelo Vista Controlador.

Fuente. Bajada de internet.

El primer paso en el ciclo de vida empieza cuando el usuario hace una solicitud al controlador con información sobre lo que el usuario desea realizar. Entonces el Controlador decide a quién debe delegar la tarea y es aquí donde el Modelo empieza su trabajo. En esta etapa, el Modelo se encarga de realizar operaciones sobre la información que maneja para cumplir con lo que le solicita el Controlador. Una vez que termina su labor, le regresa al Controlador la información resultante de sus

operaciones, el cual a su vez redirige a la Vista. La Vista se encarga de transformar los datos en información visualmente entendible para el usuario. Finalmente, la representación gráfica es transmitida de regreso al Controlador y éste se encarga de transmitírsela al usuario. El ciclo entero puede empezar nuevamente si el usuario así lo requiere.

2.1.2. VENTAJAS DE LA ARQUITECTURA MODELO VISTA CONTROLADOR.

Las principales ventajas de hacer uso del patrón MVC son:

- La separación del Modelo de la Vista, es decir, separar los datos de la representación visual de los mismos.
- Es mucho más sencillo agregar múltiples representaciones de los mismos datos o información.
- Facilita agregar nuevos tipos de datos según sea requerido por la aplicación ya que son independientes del funcionamiento de las otras capas.
- Crea independencia de funcionamiento.
- Facilita el mantenimiento en caso de errores.
- Ofrece maneras más sencillas para probar el correcto funcionamiento del sistema.
- Permite el escalamiento de la aplicación en caso de ser requerido.

2.1.3. DESVENTAJAS DE LA ARQUITECTURA MODELO VISTA CONTROLADOR.

Las principales desventajas de hacer uso del patrón MVC son:

- La separación de conceptos en capas agrega complejidad al sistema.
- La cantidad de archivos a mantener y desarrollar se incrementa considerablemente.
- La curva de aprendizaje del patrón de diseño es más alta que usando otros modelos más sencillos.

2.1.4. ARQUITECTURA MODELO VISTA CONTROLADOR APLICADA.

Para dar comienzo al proceso de desarrollo e implementación se acordó utilizar un patrón arquitectónico que nos permitiera tener un proyecto más organizado y estandarizar el proceso de programación para que en el futuro sea más fácil su mantenimiento. Para la construcción del prototipo, se utilizó la arquitectura modelo vista controlador, debido a las ventajas que ofrece como: Portabilidad, escalabilidad y facilidad de mantenimiento.

2.2. PHP.

El lenguaje PHP es un lenguaje de programación de estilo clásico, es decir que es un lenguaje de programación con variables, sentencias condicionales, bucles, funciones, etc. No es un lenguaje de etiquetas como podría ser HTML, XML o WML. Está más cercano a JavaScript o a C++, para aquellos que conocen estos lenguajes.

Pero a diferencia de Java o JavaScript que se ejecutan en el navegador, PHP se ejecuta en el servidor, por eso nos permite acceder a los recursos que tenga el servidor como por ejemplo podría ser una base de datos. El programa PHP es ejecutado en el servidor y el resultado enviado al navegador. El resultado es normalmente una página HTML.

Al ser PHP un lenguaje que se ejecuta en el servidor no es necesario que su navegador lo soporte, es independiente del navegador, sin embargo para que las páginas PHP funcionen, el servidor donde están alojadas debe soportar PHP.

La ventaja que tiene PHP sobre otros lenguajes de programación que se ejecutan en el servidor (como podrían ser los script CGI Perl), es que nos permite intercalar las sentencias PHP en las páginas HTML, es un concepto algo complicado de entender si no se ha visto nunca como funciona unas páginas PHP o ASP.

2.2.1. MODELO DE ACCESO PHP.

Figura 3. Modelo de acceso PHP.

Fuente. Bajada de Internet.

2.2.2. ENTERPRISEDB-APACHEPHP.

El servidor web apache/PHP para postgreSQL se instala como complemento ejecutando la aplicación Stack Builder y seleccionando la opción respectiva, como podemos apreciar en la siguiente imagen.

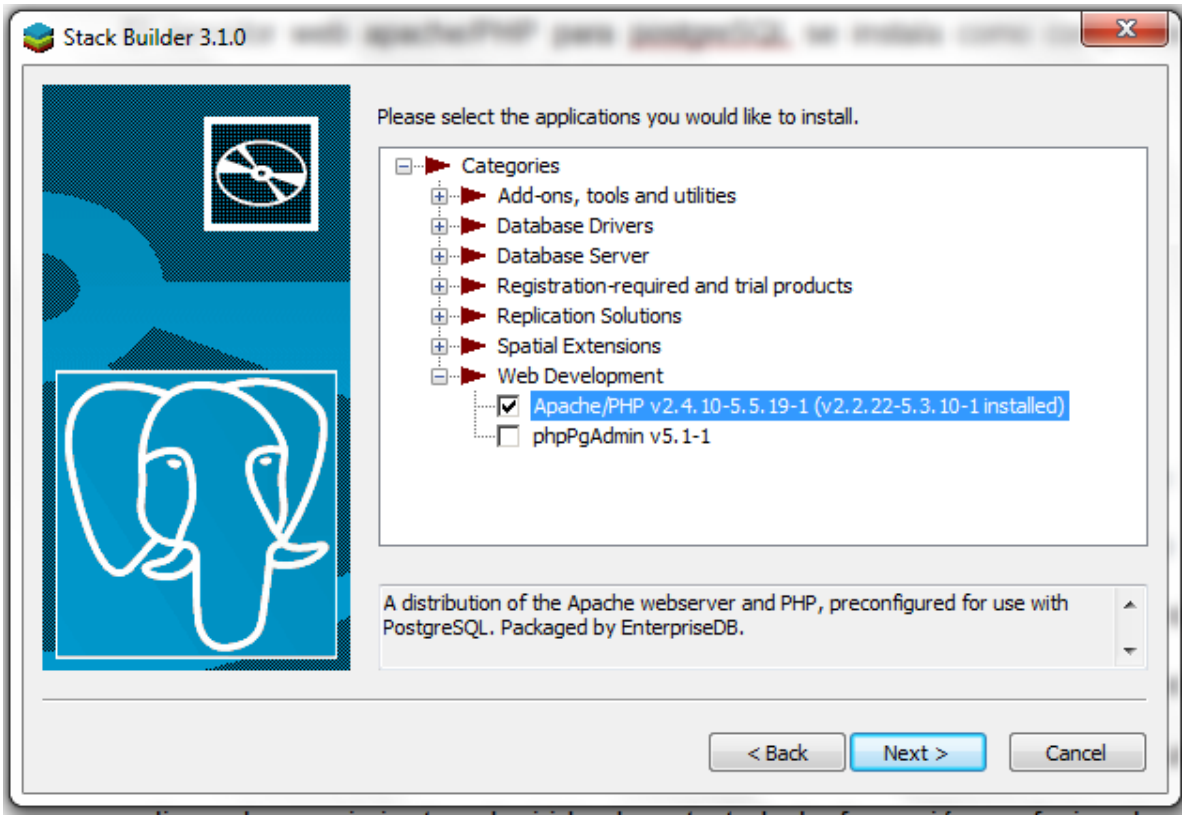

Figura 4. Instalación del servidor Apache PHP.

Fuente. Autor.

2.3. POSTGRESQL.

Características:

- Alta concurrencia. Mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos.

- **Tiene soporte para:**
  - Números de precisión arbitraria.
  - Texto de largo ilimitado.
  - Figuras geométricas.
  - Direcciones IP(IPv4 e IPv6).
  - Bloqueos de direcciones estilo CDR.
  - Direcciones MAC.
  - Arrays.
- Claves ajenas también denominadas Llaves ajenas o Claves Foráneas (foreign keys).

- Disparadores (triggers): Un disparador o trigger se define en una acción específica basada en algo ocurrente entro de la base de datos. En PostgreSQL esto significa la ejecución de un procedimiento almacenado basado en una determinada acción sobre una tabla específica. Ahora todos los disparadores se definen por seis características:

- El nombre del disparador o trigger
- El momento en que el disparador debe arrancar
- La tabla donde el disparador se activará
- La frecuencia de la ejecución
- La función que podría ser llamada.

2.4. PROGRAMACIÓN ORIENTADA A OBJETOS.

La programación estructurada tradicional se basa fundamentalmente en la ecuación de Wirth: Algoritmos + Estructuras de Datos = Programas

Esta ecuación significa que en la programación estructurada u orientada a procedimientos los datos y el código se trata por separado y lo único que se realiza son funciones o procedimientos que tratan esos datos y los van pasando de unos a otros hasta que se obtiene el resultado que se desea.

La *Programación Orientada a Objetos, POO* (OOP, Object Oriented Programming, en inglés), es una técnica de programación cuyo soporte fundamental es el ***objeto***. Un objeto es una extensión de un *Tipo Abstracto de Datos (TAD),* concepto ampliamente utilizado desde la década de los setenta. Un TAD es un tipo definido por el usuario, que encapsula un conjunto de datos y las operaciones sobre estos datos.

2.4.1. CLASES.

Son colecciones de objetos de características idénticas. Cuando se programa un objeto y se definen sus características y funcionalidades, realmente se programa una clase. Por lo tanto para realizar la abstracción de sistemas naturales, observamos y analizamos un grupo de cosas que tengan características comunes, el resultado de esta abstracción será válido para todas y cada una de estas cosas, y al conjunto de todas ellas lo llamamos clase.

2.4.2. OBJETOS.

Un objeto es cualquier cosa real o abstracta, que posee atributos y un conjunto de operaciones que manipulan esos atributos; atributos y métodos que le dan al objeto un comportamiento particular. Un objeto es una instancia de una clase, el estado del objeto se determina por el estado (valor) de sus propiedades o características (atributos). Por ejemplo, si observamos el estado de un vehículo en movimiento, uno de sus atributos es la velocidad actual de desplazamiento.

2.4.3. ATRIBUTOS.

Los atributos son las características de un objeto. Son un conjunto de datos (valores) y calificadores para aquellos datos. Estos atributos pueden ser desde tipos de datos simples (enteros, caracteres, cadenas de texto) hasta otros objetos.

2.4.4. MÉTODOS.

Son funciones o procedimientos propios de la clase que pueden tener acceso a los atributos de la misma para realizar las operaciones para los que son programados.

2.4.5. HERENCIA.

Se fundamenta en usar una clase ya creada para tomar sus características en clases más especializadas o derivadas de ésta para reutilizar el código que sea común con la clase base, y solamente definir nuevos métodos o redefinir algunos de los existentes para ajustarse al comportamiento particular de esta subclase.

2.4.6. BENEFICIOS DE LA PROGRAMACIÓN ORIENTADA A OBJETOS.

Mantenibilidad (facilidad de mantenimiento). Los programas que se diseñan utilizando el concepto de orientación a objetos son más fáciles de leer y comprender y el control de la complejidad del programa se consigue gracias a la ocultación de la información que permite dejar visibles sólo los detalles más relevantes.

Modificabilidad (facilidad para modificar los programas). Se pueden realizar añadidos o supresiones a programas simplemente añadiendo, suprimiendo o modificando objetos.

Resusabilidad. Los objetos, si han sido correctamente diseñados, se pueden usar numerosas veces y en distintos proyectos.

Fiabilidad. Los programas orientados a objetos suelen ser más fiables ya que se basan en el uso de objetos ya definidos que están ampliamente testados.

2.5. NORMA ISO/IEC 25010–MODELO DE CALIDAD.

La calidad en uso se refiere a productos en uso, se refiere a productos que probablemente sí hayan sido lanzados de forma comercial, y en los cuales se desea medir el grado de satisfacción del usuario con el programa, específicamente mide la capacidad del producto de software de ser entendido, aprendido, usado y atractivo al usuario, cuando es utilizado bajo condiciones específicas.

.

El modelo de calidad para la calidad del software en uso, posee 4 componentes en la Norma ISO/IEC 25010–MODELO DE CALIDAD, estos componentes son:

- **Apropiabilidad (Comprensibilidad):** Capacidad del producto de software para permitir al usuario entender si el software es adecuado, y cómo puede ser utilizado para las tareas y las condiciones particulares de la aplicación.

- **Facilidad de aprendizaje:** Capacidad del producto de software para permitir al usuario aprender su aplicación. Un aspecto importante a considerar aquí es la documentación del software.

- **Operabilidad:** Capacidad del producto de software para permitir al usuario operarlo y controlarlo. Aspectos de adaptación, facilidad de cambio e instalación pueden afectar la operabilidad, igualmente corresponde a la conformidad, tolerancia a error y control que concuerdan con las expectativas del usuario.

- **Atractibilidad:** Capacidad del producto de software para ser atractivo al usuario. Se refiere a los atributos del software deseados para hacer éste más atractivo al usuario, tales como el uso del color y los diseños gráficos.

2.6. LENGUAJE DE MODELADO UNIFICADO.

UML es un lenguaje estándar que sirve para escribir los *planos del software*, puede utilizarse para visualizar, especificar, construir y documentar todos los artefactos que componen un sistema con gran cantidad de software. UML puede usarse para modelar desde sistemas de información hasta aplicaciones distribuidas basadas en Web, pasando por sistemas empotrados de tiempo real. UML es solamente un lenguaje por lo que es sólo una parte de un método de desarrollo software, es independiente del proceso aunque para que sea optimo debe usarse en un proceso dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental.

UML es un lenguaje por que proporciona un vocabulario y las reglas para utilizarlo, además es un lenguaje de modelado lo que significa que el vocabulario y las reglas se utilizan para la representación conceptual y física del sistema.

UML es un lenguaje que nos ayuda a interpretar grandes sistemas mediante gráficos o mediante texto obteniendo modelos explícitos que ayudan a la comunicación durante el desarrollo ya que al ser estándar, los modelos podrán ser interpretados por personas que no participaron en su diseño (e incluso por herramientas) sin ninguna ambigüedad. En este contexto, UML sirve para *especificar*, modelos concretos, no ambiguos y completos.

Debido a su estandarización y su definición completa no ambigua, y aunque no sea un lenguaje de programación, UML se puede conectar de manera directa a lenguajes de programación como Java, C++ o Visual Basic, esta correspondencia permite lo que se denomina como ingeniería directa (obtener el código fuente partiendo de los modelos) pero además es posible reconstruir un modelo en UML partiendo de la implementación, o sea, la ingeniería inversa.

UML proporciona la capacidad de modelar actividades de planificación de proyectos y de sus versiones, expresar requisitos y las pruebas sobre el sistema, representar todos sus detalles así como la propia arquitectura. Mediante estas capacidades se obtiene una documentación que es válida durante todo el ciclo de vida de un proyecto.

Diagramas

Los diagramas se utilizan para representar diferentes perspectivas de un sistema de forma que un diagrama es una proyección del mismo. UML proporciona un amplio conjunto de diagramas que normalmente se usan en pequeños subconjuntos para poder representar las cinco vistas principales de la arquitectura de un sistema.

Diagramas de Clases

Muestran un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Estos diagramas son los más comunes en el modelado de sistemas orientados a objetos y cubren la vista de diseño estática o la vista de procesos estática (sí incluyen clases activas).

Diagramas de Objetos

Muestran un conjunto de objetos y sus relaciones, son como fotos instantáneas de los diagramas de clases y cubren la vista de diseño estática o la vista de procesos estática desde la perspectiva de casos reales o prototípicos.

Diagramas de Casos de Usos

Muestran un conjunto de casos de uso y actores (tipo especial de clases) y sus relaciones. Cubren la vista estática de los casos de uso y son especialmente importantes para el modelado y organización del comportamiento.

Diagramas de Secuencia y de Colaboración

Tanto los diagramas de secuencia como los diagramas de colaboración son un tipo de diagramas de interacción. Constan de un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar unos objetos a otros. Cubren la vista dinámica del sistema. Los diagramas de secuencia enfatizan el ordenamiento temporal de los mensajes mientras que los diagramas de colaboración muestran la organización estructural de los objetos que envían y reciben mensajes. Los diagramas de secuencia se pueden convertir en diagramas de colaboración sin pérdida de información, lo mismo ocurren en sentido opuesto.

Diagramas de Estados

Muestran una máquina de estados compuesta por estados, transiciones, eventos y actividades. Estos diagramas cubren la vista dinámica de un sistema y son muy importantes a la hora de modelar el comportamiento de una interfaz, clase o colaboración.

Diagramas de Actividades

Son un tipo especial de diagramas de estados que se centra en mostrar el flujo de actividades dentro de un sistema. Los diagramas de actividades cubren la parte dinámica de un sistema y se utilizan para modelar el funcionamiento de un sistema resaltando el flujo de control entre objetos.

Diagramas de Componentes

Muestra la organización y las dependencias entre un conjunto de componentes. Cubren la vista de la implementación estática y se relacionan con los diagramas de clases ya que en un componente suele tener una o más clases, interfaces o colaboraciones.

Diagramas de Despliegue

Representan la configuración de los nodos de procesamiento en tiempo de ejecución y los componentes que residen en ellos. Muestran la vista de despliegue estática de una arquitectura y se relacionan con los componentes ya que, por lo común, los nodos contienen uno o más componentes.

3. MÉTODO DE INVESTIGACIÓN.

3.1. TIPO DE INVESTIGACION.

La presente investigación es aplicada porque busca la utilización de los conocimientos obtenidos durante la formación profesional. Éste tipo de investigación está orientada a la solución de problemas prácticos, y corresponden a la aplicación de la investigación a problemas definidos en situaciones y aspectos específicos. Teniendo en cuenta lo anterior, la presente investigación busca aplicar el conocimiento adquirido durante toda la formación profesional en la IMPLEMENTACIÓN DE UN PROTOTIPO EN

SOFTWARE LIBRE PARA LA GESTION DE PRESTAMO DE MATERIALES en la Universidad de Cartagena.

La investigación es entendida como "*un proceso sistemático donde se indaga y se aplica al estudio de un fenómeno*"[1]. Por lo que, la presente investigación es de tipo descriptivo y, de corte cuantitativo porque busca recolectar datos para medir y evaluar estadísticamente la efectividad y eficacia del diseño y validación de un sistema de información para la GESTION DE PRESTAMO DE MATERIALES en la Universidad de Cartagena. Este tipo de investigación mide y evalúa qué características tiene el diseño de un sistema de información, qué información es necesaria para su actualización y qué tan efectivo es su diseño para la respectiva validación de su funcionalidad.

En consecuencia a lo dicho anteriormente, el tipo de investigación utilizado será **mixta (cualitativa – cuantitativa)** puesto que, ésta esencialmente, desarrolla procesos en términos descriptivos e interpreta acciones, expresiones, hechos funcionales relevantes y los sitúa en una correlación con el más amplio contexto social. El estudio es experimental porque en ella se manipulan y se experimenta con la variable independiente del proyecto "Diseño y validación de un sistema de información" elaborando diferentes modelos, estilos, presentaciones y esquemas para la presentación de la información, y de esta manera medir que tan efectivo y eficaz resulta para el registro y control del préstamo de materiales en la Universidad de

[1] HERNÁNDEZ S. Roberto y otros (2008). Mc Graw Hill. México Pág. 12.

Cartagena, analizando de esta manera los efectos en las variables dependientes “Laboratorios”.

En la presente investigación se utiliza el cuestionario cerrado para medir, describir y evaluar su diseño, las características y los efectos de la implementación de un software, la cual es la variable de interés, atendiendo a: características de su diseño y la selección de la información necesaria para su implementación. De igual manera, se hará empleo de la entrevista como punto de partida para conocer la opinión de docentes, directivos y estudiantes de ingeniería de sistemas acerca del diseño del mencionado sistema de información.

Cabe resaltar que para el diseño del sistema de información se utiliza un modelo de diseño de software y se emplea el lenguaje de programación más común actualmente como lo es PHP y una base de datos diseñada bajo las reglamentaciones de POSTGRESQL, atendiendo a los siguientes pasos para su desarrollo: Primero, realizar un análisis de requerimiento; segundo, realizar una especificación; tercero, diseño y arquitectura; cuarto, programación; quinto, puesta a prueba; sexto, documentación; y, por último, el mantenimiento.

3.2. POBLACION Y MUESTRA.

Dentro del enfoque cuantitativo la MUESTRA es considerada como “el subgrupo de la población del cual se recolectan los datos y deben ser representativos de dicha población.

En el presente proyecto la población está conformada por 450 estudiantes pertenecientes al programa de ingeniería de sistemas de todos los semestres, de igual manera se incluye a 11 docentes, un coordinador de programas, para un total de 462 personas que conforman la población total. Esta población se caracteriza porque sus integrantes son personas entre los 17 y los 43 años de edad con un conocimiento básico en el manejo de tecnologías informáticas. La muestra escogida corresponde 8 estudiantes.

4. MARCO METODOLÓGICO.

Para implementar los servicios del prototipo para el sistema de gestión de préstamo de materiales en la Universidad de Cartagena se utilizó como metodología de desarrollo el prototipado evolutivo.

La principal razón por la cual se eligió la metodología de desarrollo de prototipo evolutivo es por lo que el sistema de gestión de préstamos de materiales en la Universidad de Cartagena está cambiando constantemente, presentando nuevas necesidades por parte de los usuarios o en su defecto modificando los servicios ya existentes, así de esta manera nos encontramos no frente a sistema estático sino por el contrario ante un sistema dinámico que sufre cambios constantemente; el procedimiento seguido por la metodología utilizada se describe en los siguientes items.

- Se puntualizaron las necesidades globales del software realizando reuniones entre los desarrolladores, los administradores de las salas de laboratorios y el curso muestra que serán los usuarios del sistema, estos estuvieron aportando ideas durante todo el desarrollo del mismo, se identificaron los requisitos y se determinaron los subprocesos donde era necesario hacer mayor énfasis.

- Se realizó el Diseño del Prototipo teniendo en cuenta primeramente los aspectos relevantes del código como por ejemplo, métodos de consultas, inserción, actualización y borrado en la base de datos; luego se prosiguió a diseñar la interfaces de usuario como son los formularios de entrada/salida de datos.

- El prototipo es evaluado por los desarrolladores del proyecto, los administradores de las salas de laboratorios y el curso muestra.

- Se produce un proceso interactivo en el que el prototipo es “depurado” (Refinamiento del prototipo) para que satisfaga las necesidades del usuario, al mismo tiempo que facilita a los desarrolladores una mejor comprensión de lo que hay que hacer para poder entregar el producto final requerido.

4.1. ANÁLISIS DE REQUISITOS.

Uno de los elementos que se tienen en cuenta para el análisis de requerimientos es el considerado como el del requerimiento funcional para este se determinan las posibles transformaciones que se producen en un sistema debido a la entradas y como es su

efecto en sus posibles salidas, para el estudio del prototipo en esta dimensión se procedió a hacer un análisis comparativo con cinco aplicativos, software relacionados primero bajo el licenciamiento de software libre y segundo bajo el concepto de la gestión de préstamos o asignación de recursos, enmarcado en la técnica utilizada en la ingeniería de requerimientos donde se consideran que para el desarrollo del prototipo se deben identificar los requerimientos que son conocidos, para luego señalar áreas en las que será necesario las posibles implementaciones, los criterios utilizados fueron: Los programas enfocados a universidades, asignación de horas específicas, asignación individual del recurso, dado un recurso permite consultar su disponibilidad, permite tener varios tipos de usuario y uso vía web.

	Tablix	Openctt	Timefinder	Visual timetabler	Fet
Enfocado a universidades	NO	NO	SI	SI	SI
Asignación de Hora específica	NO	NO	SI	NO	NO
Asignación individual del recurso	NO	NO	NO	NO	NO
Dado un recurso consultar su disponibilidad	NO	NO	NO	NO	NO
Varios tipos de usuario	NO	NO	NO	NO	NO
Uso vía web	NO	NO	NO	SI	SI

TABLA 1. Análisis de requerimientos.

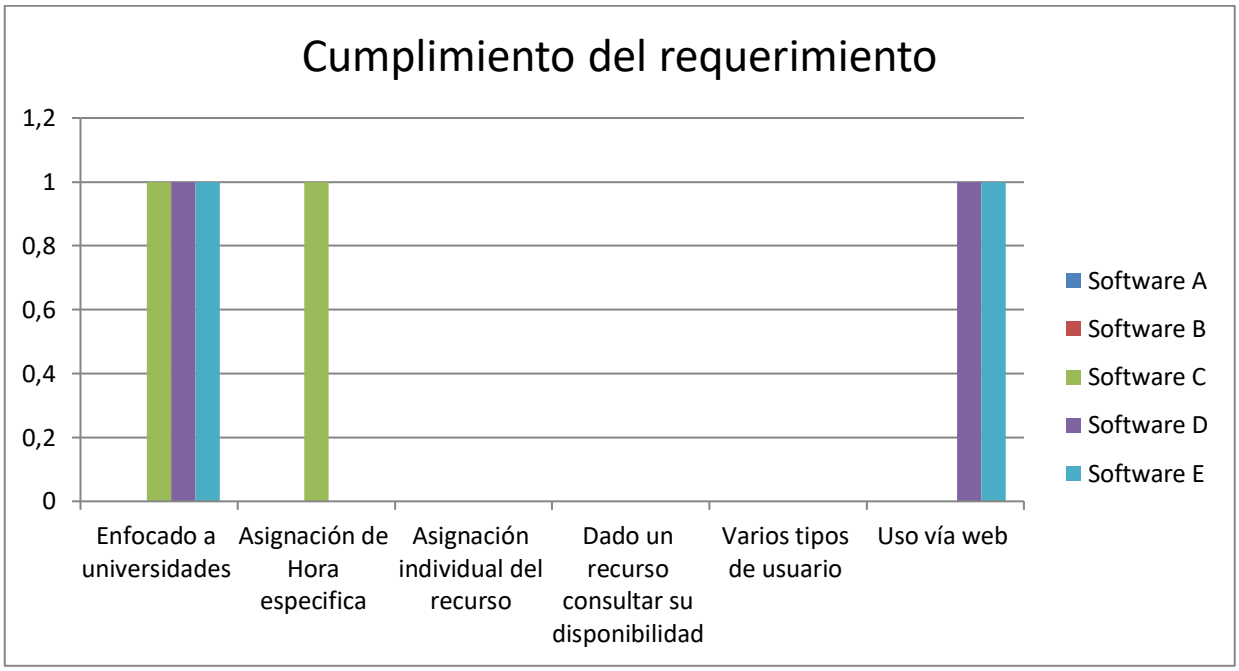

FIGURA 5. Análisis de cumplimiento del requerimiento

La primera característica hace referencia si el aplicativo está enfocado a escuela u universidad como lo requerimos, la cumplen tres de ellos.

La segunda característica hace referencia si en el aplicativo la asignación del recurso se hace en rangos de horas preestablecidas o se puede hacer en una hora cualquiera como lo requerimos, solamente la cumple uno de ellos.

La tercera característica hace referencia si en el aplicativo la asignación del recurso se hace a grupos previamente conformados o se puede hacer a una persona individual como lo requerimos, no la cumple ninguno de ellos.

La cuarta característica hace referencia si en el aplicativo dado un recurso se puede consultar su disponibilidad como es requerido, ninguno de ellos la cumple.

La quinta característica hace referencia si el aplicativo permite varios tipos de usuarios como es requerido, ninguno de ellos la cumple.

La sexta característica hace referencia si el aplicativo puede ser usado vía web como es requerido, la cumplen dos de ellos.

4.1.1. REQUISITOS FUNCIONALES.

A continuación se plasman los requerimientos finales los cuales surgieron a partir de reuniones y pruebas con el cliente y el director de proyecto, gracias a la presentación de prototipos y la realimentación con el cliente se obtuvieron los siguientes requerimientos los cuales se cumplieron para el prototipo final.

1. Agregar, modificar y dar de baja elementos de laboratorio en el sistema.
2. Agregar, modificar y dar de baja a estudiantes de las carreras que utilizan estos laboratorios.
3. Agregar, modificar y dar de baja los auxiliares que van a estar usando el sistema en los laboratorios.

4. Llevar el registro del mantenimiento realizado a cada elemento del laboratorio.
5. Controlar el préstamo a los estudiantes de los elementos de laboratorio con penalización de restricción para posteriores préstamos por incumplimiento de entrega a tiempo.
6. Permitir que los estudiantes reserven los elementos de laboratorio.
7. Mostrar estadísticas de uso de los elementos.
8. Monitorear daños en los elementos por parte de los estudiantes.
9. Asignar 3 tipos de usuario, Administrador General, Administrador Local y Estudiante.
10. Asignar al perfil de administrador General la gestión de los inventarios de todos los laboratorios de ingenierías.
11. Asignar al perfil de administrador Local la gestión de préstamos e inventarios de los elementos del laboratorio a cargo.
12. El sistema debe diferenciar entre los materiales que se prestan por días, por semanas o por meses.
13. El sistema debe responder consultas acerca del número de elementos de laboratorio, tipo de elementos, etc., que se encuentran en el laboratorio o que se encuentren en préstamo.
14. El sistema debe proporcionar información actualizada acerca de los materiales que no se encuentren disponibles para su préstamo.

4.1.2. REQUISITOS NO FUNCIONALES

1. El sistema debe responder a las solicitudes del usuario en un tiempo máximo de 5 segundos.
2. El sistema no debe permitirle el acceso a zonas restringidas del sistema a usuarios no autorizados.
3. El sistema debe ser eficiente en el manejo de los recursos.
4. El sistema debe estar disponible las 24 horas, es decir que el computador donde este alojado el software debe permanecer encendido.
5. El sistema debe ser intuitivo para que cualquier persona sea capaz de usarlo.
6. El computador en el que este alojado el software, debe contar con protección de antivirus.

4.1.3. DIAGRAMAS DE CASOS DE USO.

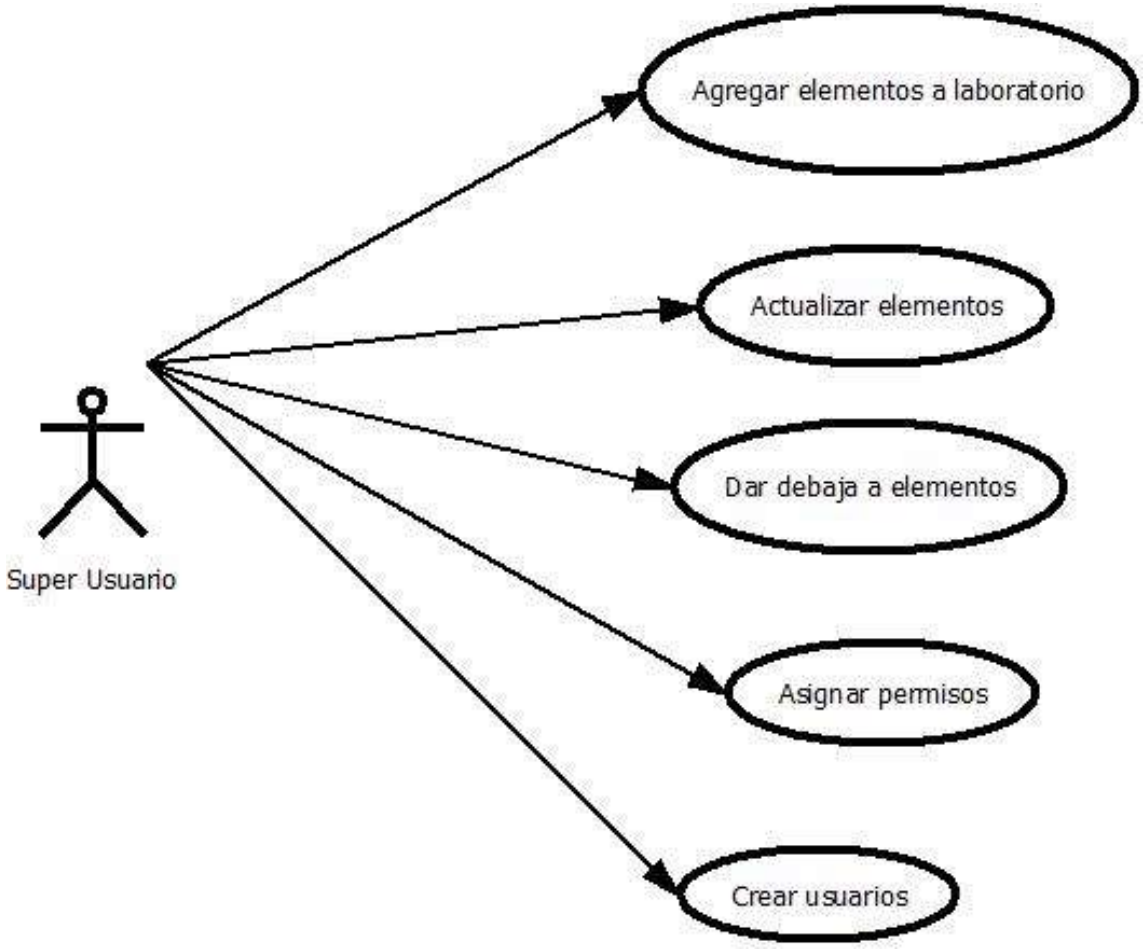

Figura 6. Diagrama de Casos de Uso: Súper Usuario.

Fuente. Autor

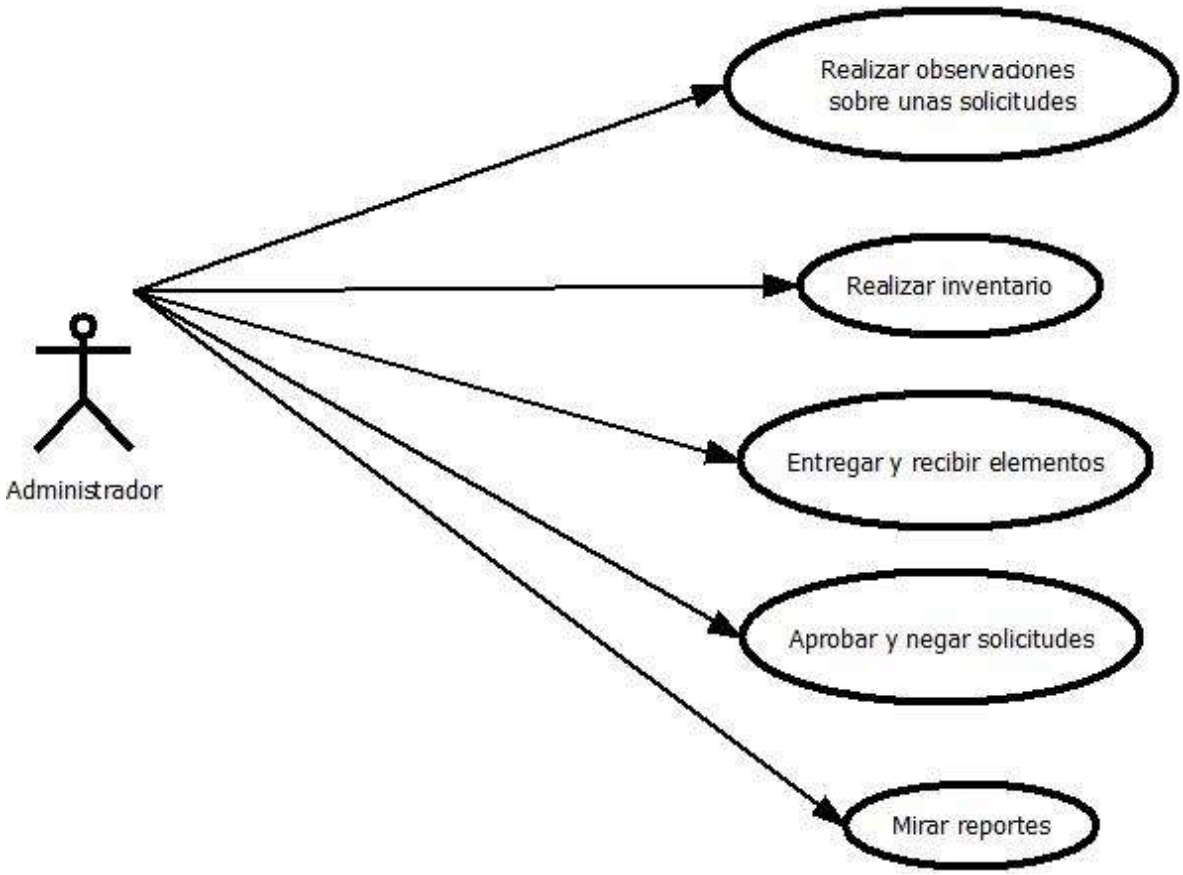

Figura 7. Diagrama de Casos de Uso: Usuario Administrador del sistema.

Fuente. Autor

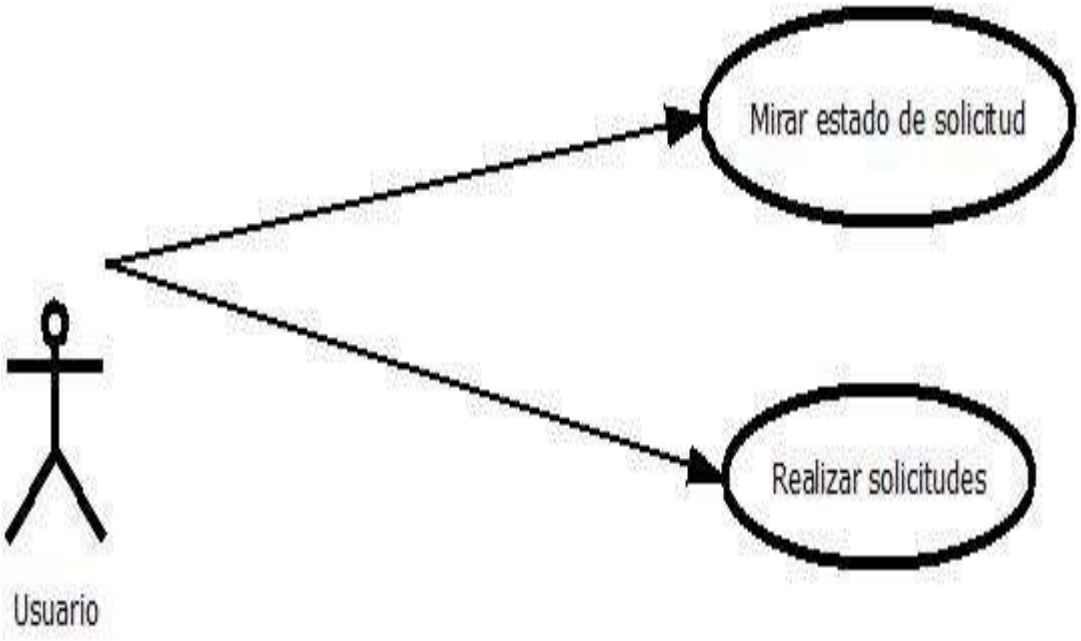

Figura 8. Diagrama de Casos de Uso: Usuario Normal.

Fuente. Autor

4.2. DIAGRAMA ENTIDAD/RELACIÓN DE LA BASE DE DATOS.

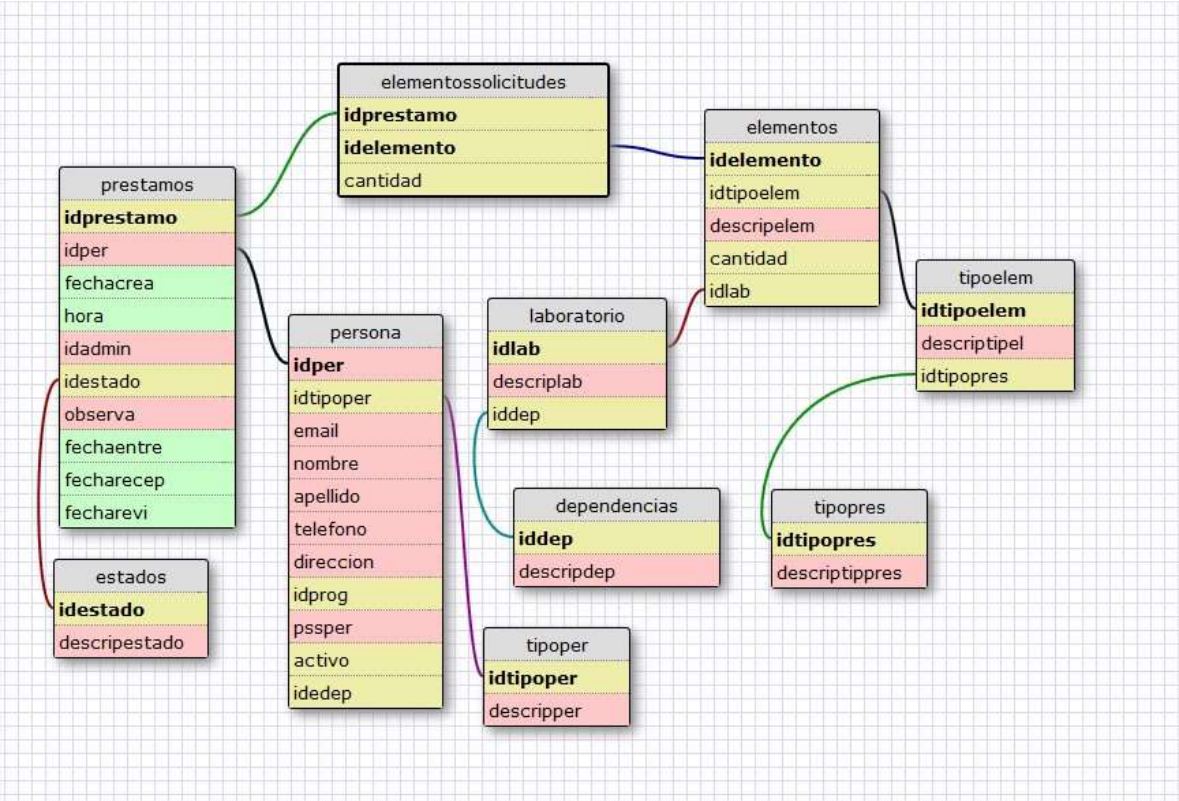

Figura 9. Modelo de Datos.

Fuente. Autor

4.2.1. TABLAS DE LA BASE DE DATOS.

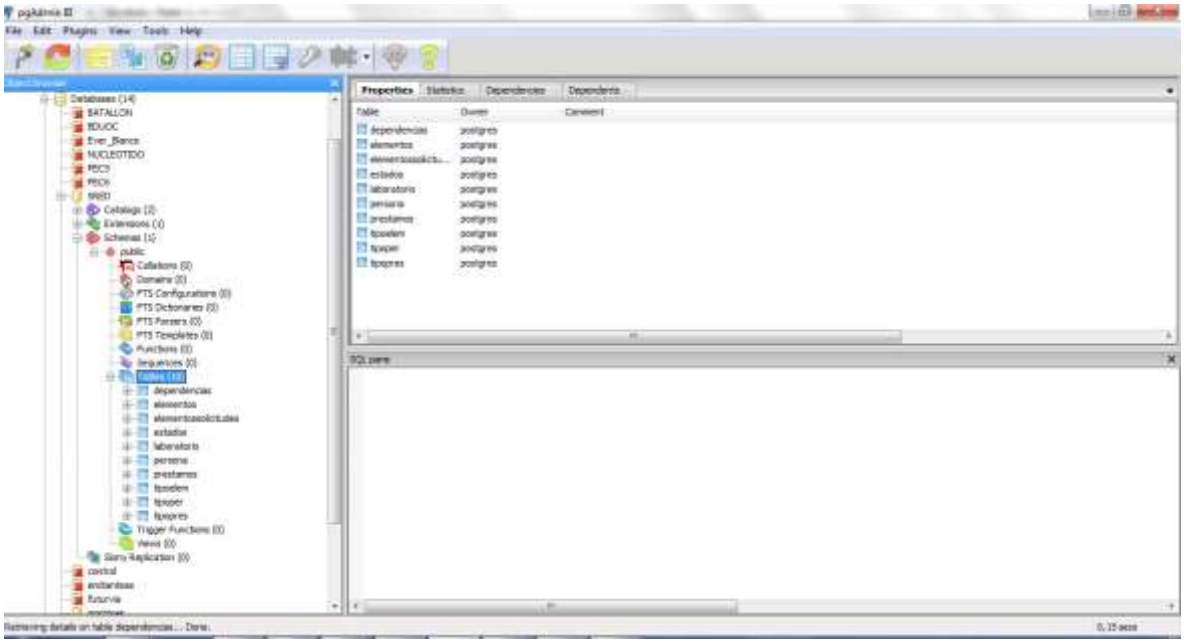

Figura 10. Tablas de la Base de Datos SRED.

Fuente. Autor.

4.3. ARQUITECTURA Y CLASES IMPLEMENTADAS

En las siguientes imágenes se muestran las clases, modelos, vistas y controladores obtenidos durante su implementación:

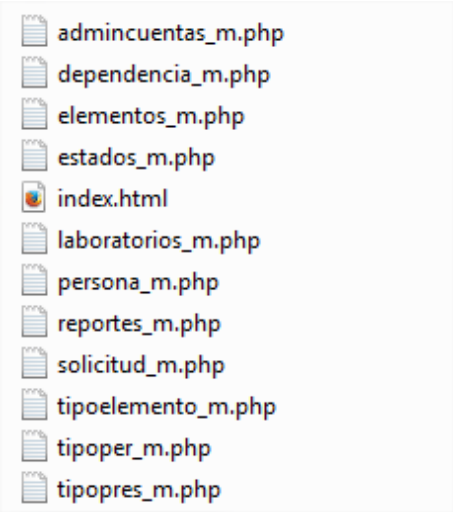

Figura 11. Modelos.

Fuente. Autor.

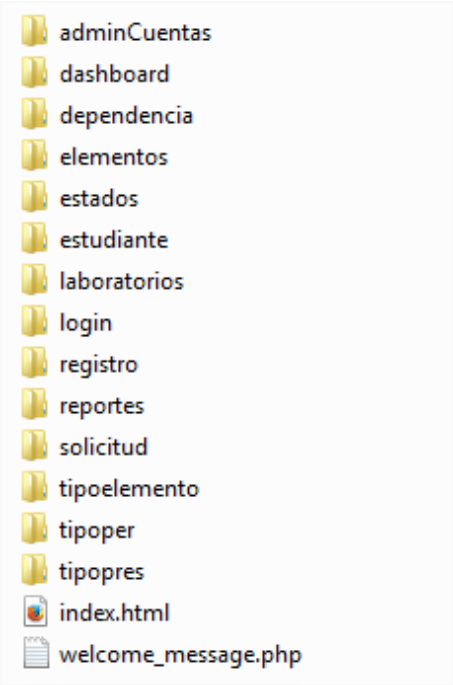

Figura 12. Vistas.

Fuente. Autor.

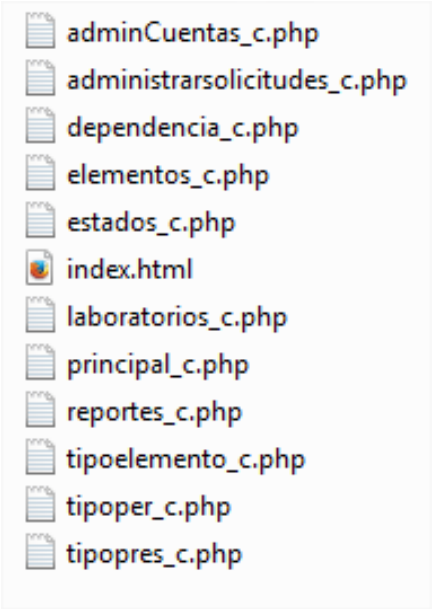

Figura 13. Controladores.

Fuente. Autor.

4.3.1. ESTRUCTURA DE CARPETAS DEL PROYECTO.

Los Directorios del sistema están organizados de tal manera que los archivos que se almacenen en ellos correspondan a lo que describe el nombre del directorio. Por ejemplo: El sistema cuenta con un Directorio llamado " **img** "; en éste se encuentran almacenadas todos los archivos .jpg, .gif, .png.

La estructura de carpetas de Sred se encuentra Distribuido de la siguiente forma:

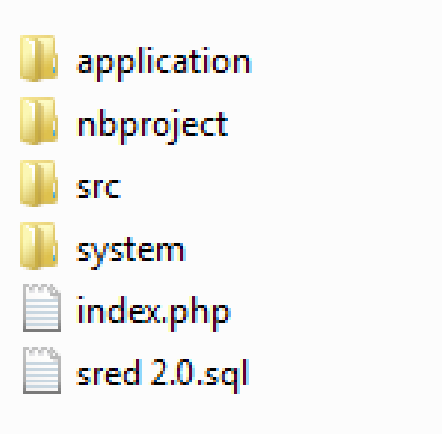

Figura 15. Carpetas principales del proyecto.

Fuente. Autor.

Para visualizar los diferentes recursos como imágenes del software pulse clic sobre la carpeta **src** al interior de esta carpeta encontrara todos los iconos e imágenes que se utilizan en el software, el interior de esta carpeta se encuentra distribuido de la siguiente forma:

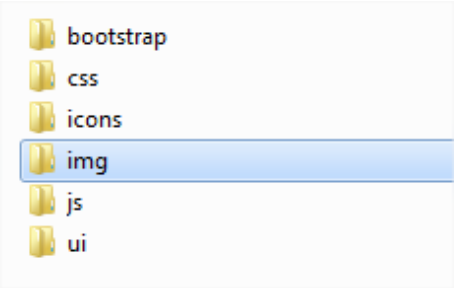

Figura 16. Carpeta src.

Fuente. Autor.

Al interior de la carpeta **application** se encuentran todos los **controladores**, **vistas** y **modelos** que son indispensables para el funcionamiento del software. El interior de esta carpeta se encuentra distribuida de la siguiente forma:

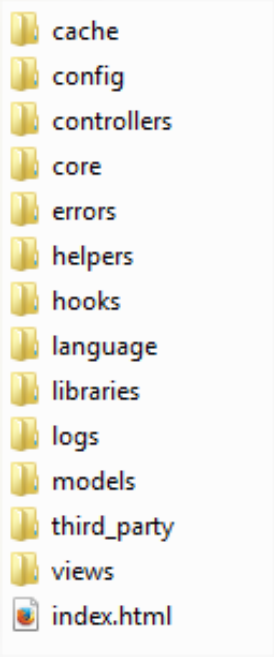

Figura 17. Carpeta application.

Fuente. Autor.

5. PRUEBAS DEL SISTEMA.

A continuación se presentan las pruebas aplicadas al sistema para garantizar que ha sido desarrollado correctamente, sin errores de diseño o programación.

5.1. PRUEBAS UNITARIAS DEL SISTEMA.

Tienen como objetivo garantizar que el software funciona correctamente y cumple con la especificación de requisitos, estas pruebas fueron practicadas por el desarrollador del sistema y posteriormente validadas por el director de proyecto.

A continuación se describe las pruebas de cada caso de uso de los servicios que fueron desarrollados y su estado.

5.1.1. VISTA DEL SÚPER USUARIO.

CASO DE USO: Agregar elementos al laboratorio.

Pulse clic en el botón **Nuevo** de la ventana **Elementos,** como podemos apreciar en la siguiente figura**.**

Figura 18. Interfaz: Agregar Elementos.

Fuente. Autor.

En la ventana **Agregar Elemento** digite la **Descripción** prueba 2, la **Cantidad**, de las listas despegables seleccione el **Laboratorio** y seleccione el **Tipo de Elemento**, pulse clic en el botón **Guardar,** como podemos apreciar en la siguiente figura.

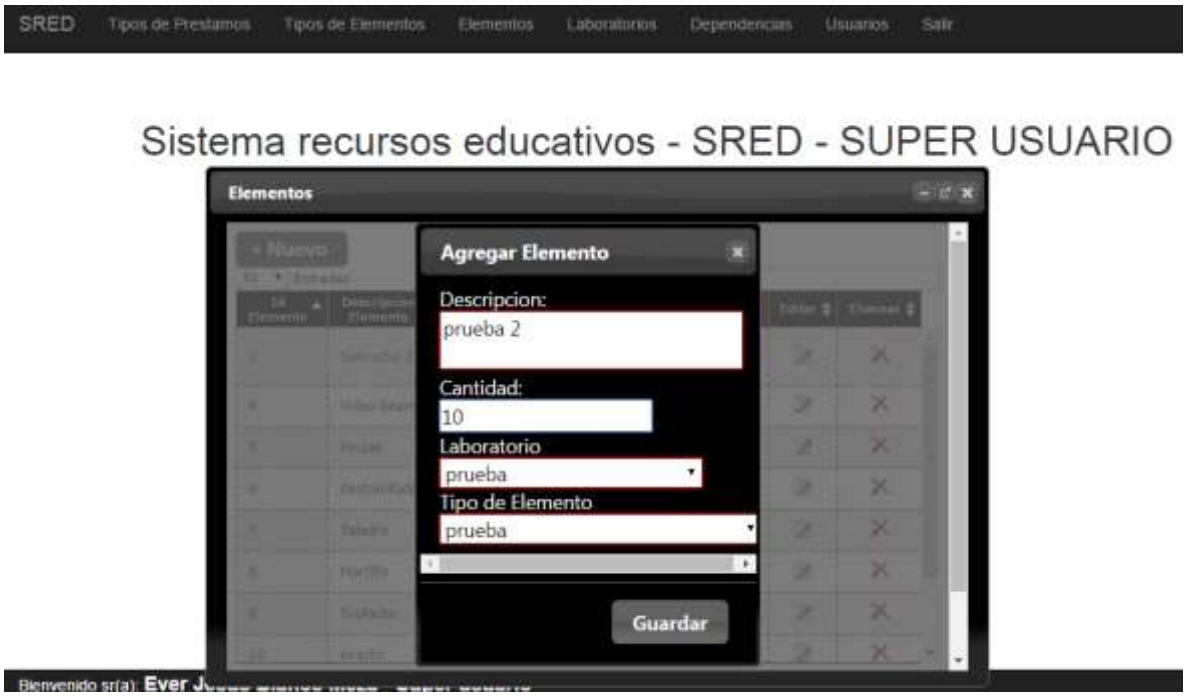

Figura 19. Interfaz: Agregar Elementos.

Fuente. Autor.

En la ventana **Elementos** podemos ver que el elemento Prueba 2 se agregó satisfactoriamente, como podemos apreciar en la siguiente figura.

Figura 20. Interfaz: Agregar Elementos.

Fuente. Autor.

CASO DE USO: Actualizar Elementos.

En la ventana **Elementos** pulse clic en el botón **Editar** correspondiente al elemento prueba 2, como podemos apreciar en la siguiente figura.

Figura 21. Interfaz: Actualizar Elementos.

Fuente. Autor.

En la ventana **Actualizar Elemento** cambie la **Descripción** del elemento prueba 2 por prueba 3, pulse clic en el botón **Guardar**, como podemos apreciar en la siguiente figura.

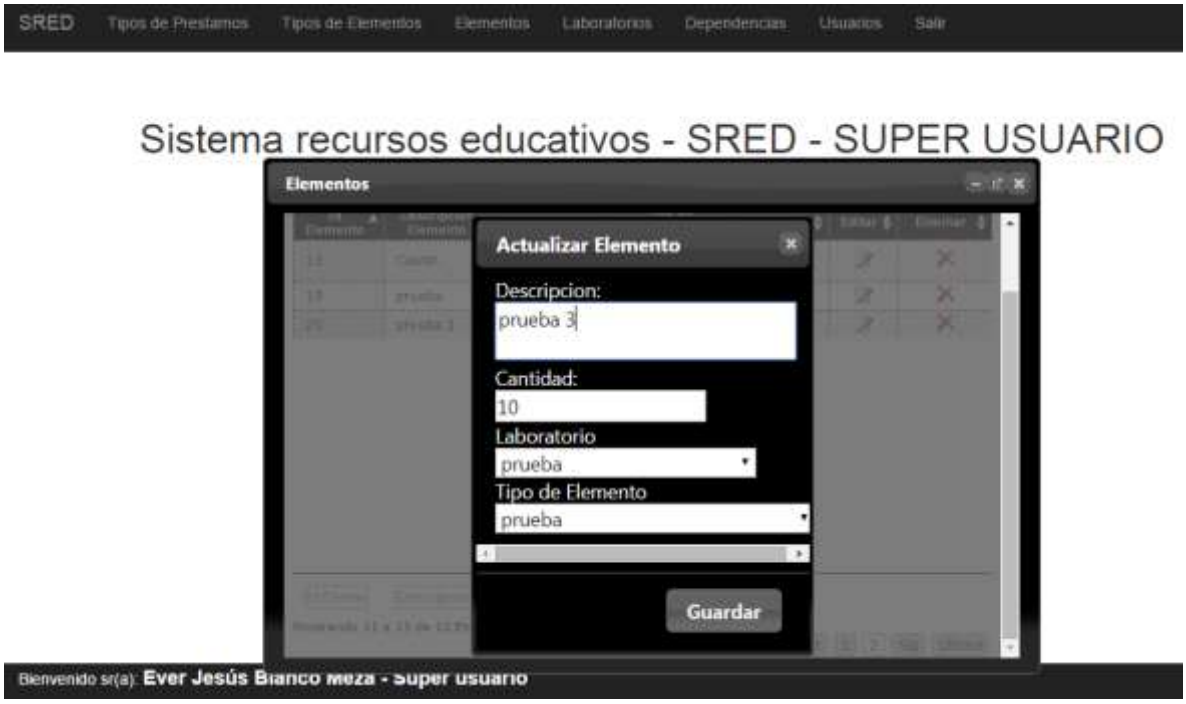

Figura 22. Interfaz: Actualizar Elementos.

Fuente. Autor.

En la ventana **Elementos** podemos ver que la **Descripción** del elemento prueba 2 fue cambiada por prueba 3 satisfactoriamente, como podemos apreciar en la siguiente figura.

Figura 23. Interfaz: Actualizar Elementos.

Fuente. Autor.

CASO DE USO: Dar de baja a elementos.

En la ventana **Elementos** pulse clic en el botón **Eliminar** correspondiente al elemento prueba 3, como podemos apreciar en la siguiente figura.

Figura 24. Interfaz: Eliminar Elementos.

Fuente. Autor.

En la ventana **Eliminar elemento** pulse clic en el botón **Aceptar**, como podemos apreciar en la siguiente figura.

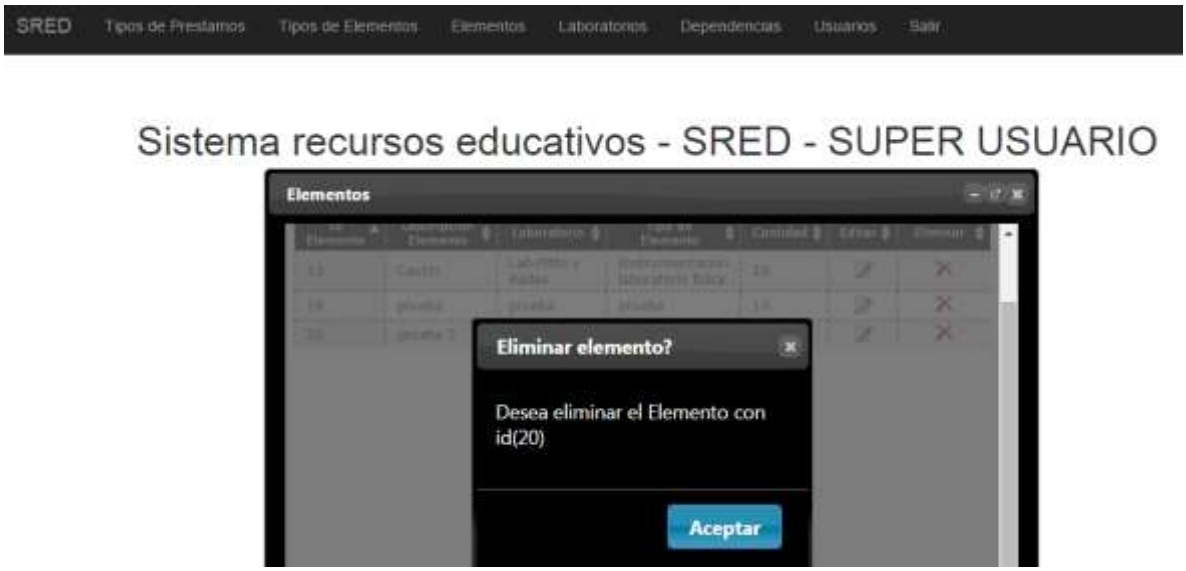

Figura 25. Interfaz: Eliminar Elementos.

Fuente. Autor.

En la ventana **Elementos** podemos ver que el elemento prueba 3 fue eliminado satisfactoriamente, como podemos apreciar en la siguiente figura.

Figura 26. Interfaz: Eliminar Elementos.

Fuente. Autor.

CASO DE USO: Asignar permisos.

Pulse clic en el menú **Usuarios** ubicado en la barra de menú principal, como podemos apreciar en la siguiente figura.

Figura 27. Interfaz: Administrar cuentas de usuarios.

Fuente. Autor.

En la ventana **Administrar cuentas usuarios** pulse clic en el botón **Editar cuenta** correspondiente al usuario Rafael, como podemos apreciar en la siguiente figura.

Figura 28. Interfaz: Administrar cuentas de usuarios.

Fuente. Autor.

En la ventana **Actualizar cuenta** pulse clic en la lista **Activo** y seleccione la opción **SI**, pulse clic en el botón **Guardar**, como podemos apreciar en la siguiente figura.

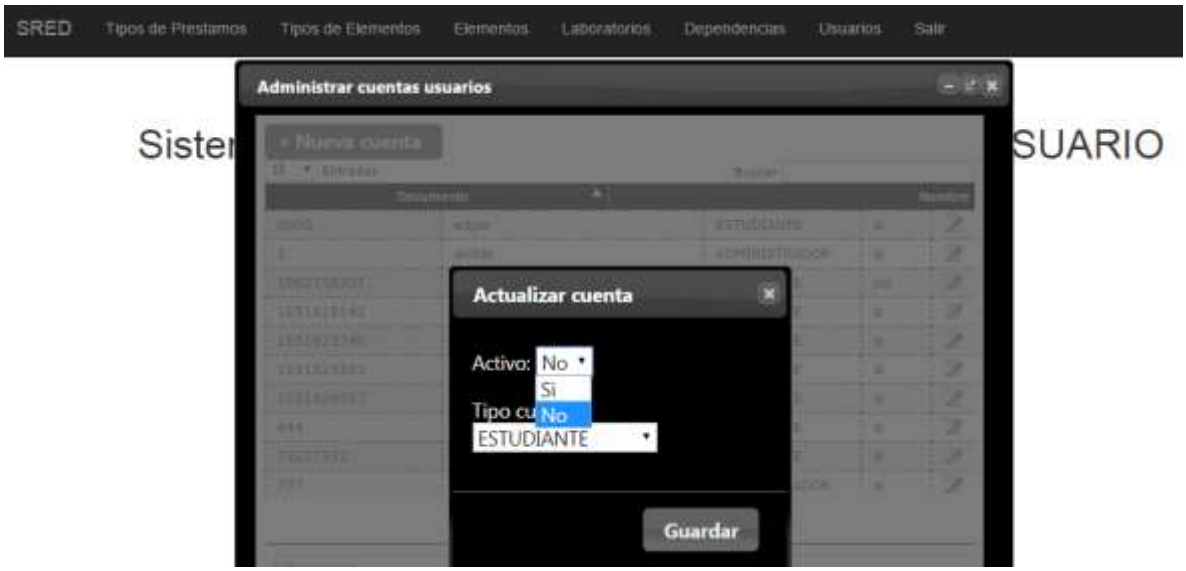

Figura 29. Interfaz: Administrar cuentas de usuarios.

Fuente. Autor.

En la ventana **Administrar cuentas usuarios** podemos ver que el usuario Rafael paso del estado no activo al estado si activo, como podemos apreciar en la siguiente figura.

Figura 30. Interfaz: Administrar cuentas de usuarios.

Fuente. Autor.

CASO DE USO: Crear usuarios.

Pulse clic en el menú **Usuarios** ubicado en la barra de menú principal, como podemos apreciar en la siguiente figura.

Figura 31. Interfaz: Administrar cuentas de usuarios.

Fuente. Autor.

En la ventana **Administrar cuentas usuarios** pulse clic en el botón **Nueva cuenta**, como podemos apreciar en la siguiente figura.

Figura 32. Interfaz: Administrar cuentas de usuarios.

Fuente. Autor

En la ventana **Agregar cuenta** ingrese los datos solicitados y pulse clic en el botón Guardar, como podemos apreciar en la siguiente figura.

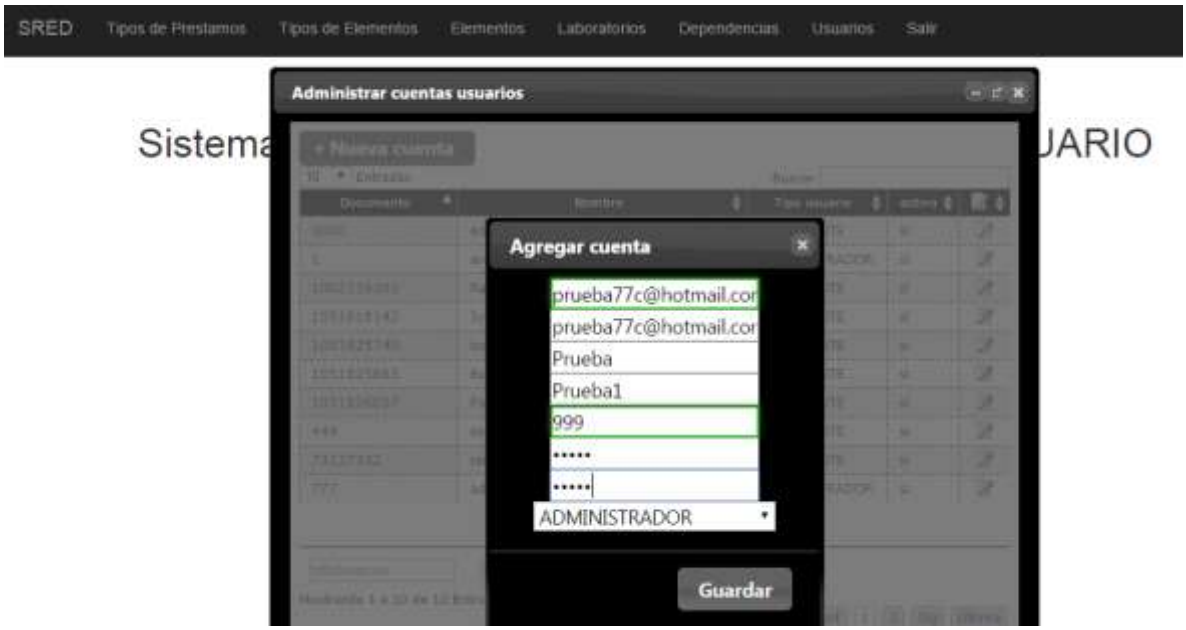

Figura 33. Interfaz: Administrar cuentas de usuarios.

Fuente. Autor.

En la ventana **Administrar cuentas usuarios** el usuario prueba fue agregado satisfactoriamente, como podemos apreciar en la siguiente figura.

Figura 34. Interfaz: Administrar cuentas de usuarios.

Fuente. Autor.

5.1.2. VISTA DEL USUARIO ADMINISTRADOR.

CASO DE USO: Aprobar solicitudes.

En la ventana principal pulsar clic sobre el botón editar correspondiente a la solicitud número 26 realizada por el usuario estudiante apestudiante, como podemos apreciar en la siguiente figura.

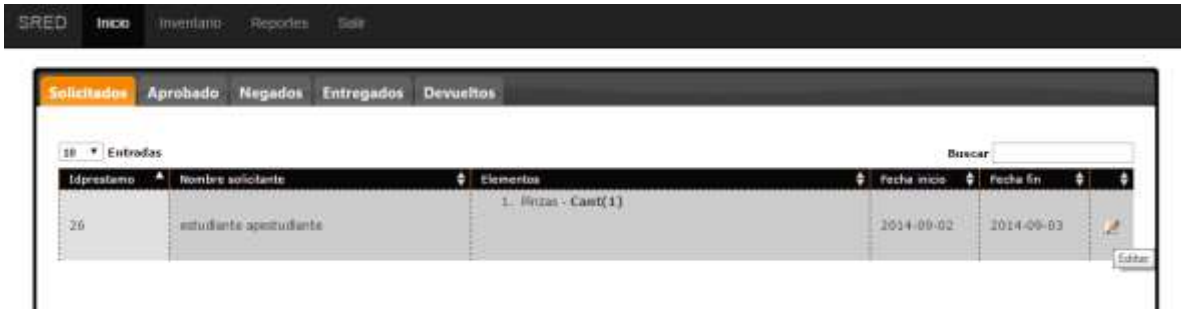

Figura 35. Interfaz: Usuario administrador.

Fuente. Autor.

En la ventana **Cambiar Estado** seleccione en la lista **Definir Estado de la solicitud** la opción **Aprobado**, como podemos apreciar en la siguiente imagen.

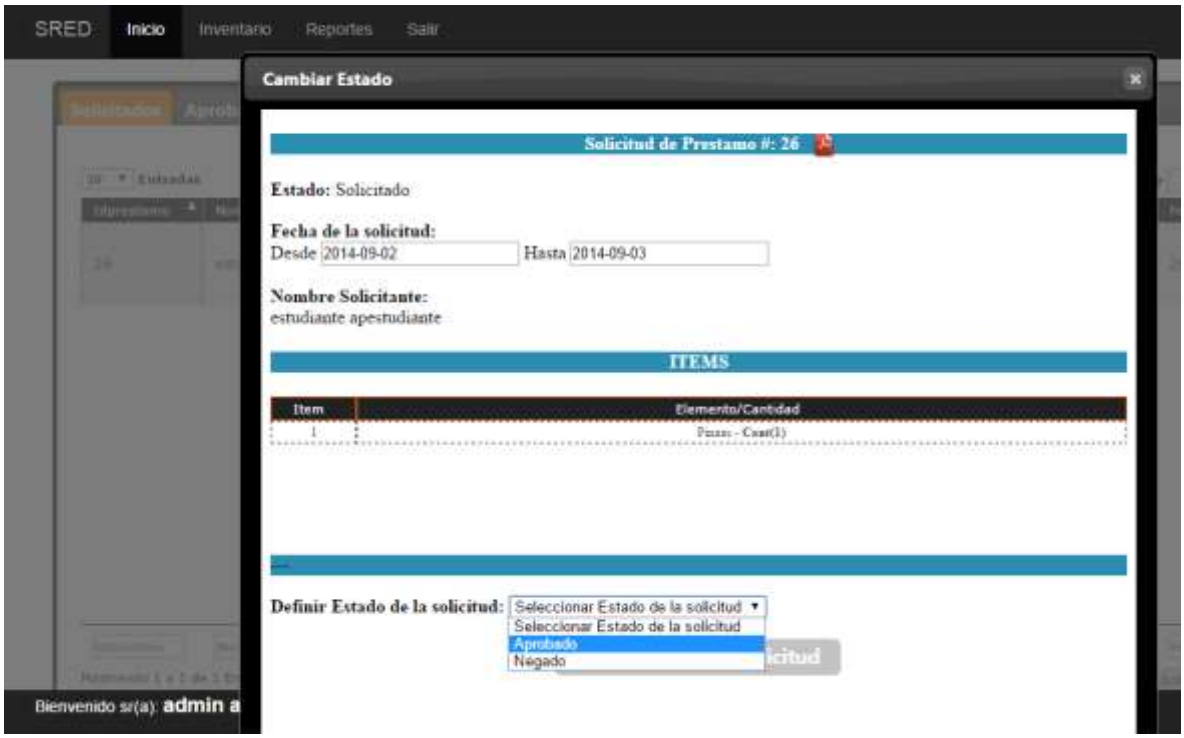

Figura 36. Interfaz: Cambiar estado de la solicitud.

Fuente. Autor

En la ventana **Cambiar Estado** pulse clic en el botón **Guardar Estado Solicitud**, como podemos apreciar en la siguiente imagen.

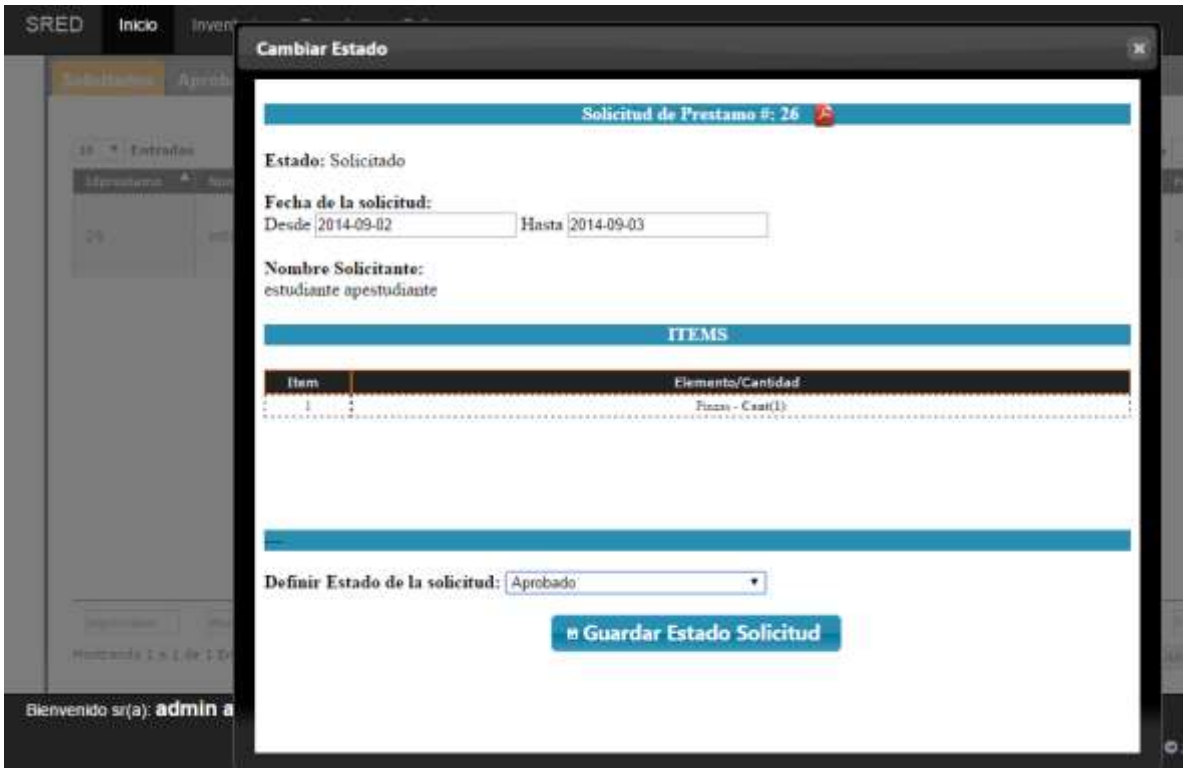

Figura 37. Interfaz: Cambiar estado de la solicitud.

Fuente. Autor

En la ventana principal podemos ver como la solicitud número 26 realizada por el usuario estudiante apestudiante paso del estado **Solicitados** al estado **Aprobado**, como podemos apreciar en la siguiente imagen.

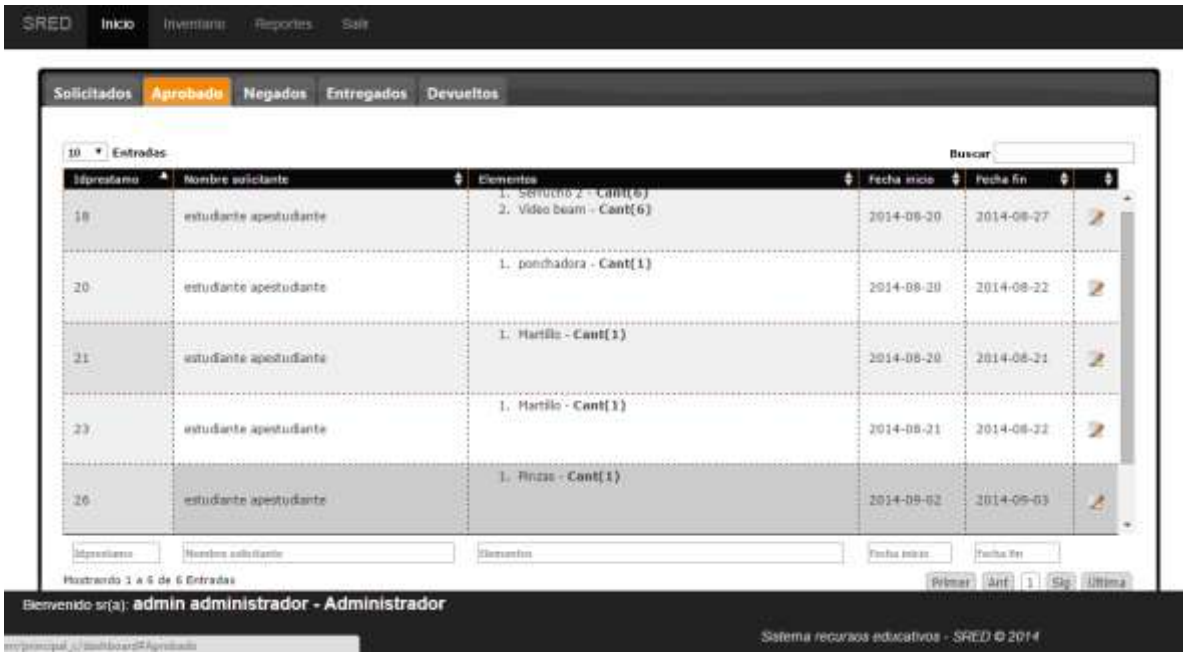

Figura 38. Interfaz: Usuario administrador.

Fuente. Autor.

CASO DE USO: Negar solicitudes.

En la ventana principal pulse clic sobre el menú **Inventario**, en la ventana **Imprimir inventario** podrá ver que no existen video beam disponibles para apartar, como podemos apreciar en la siguiente imagen.

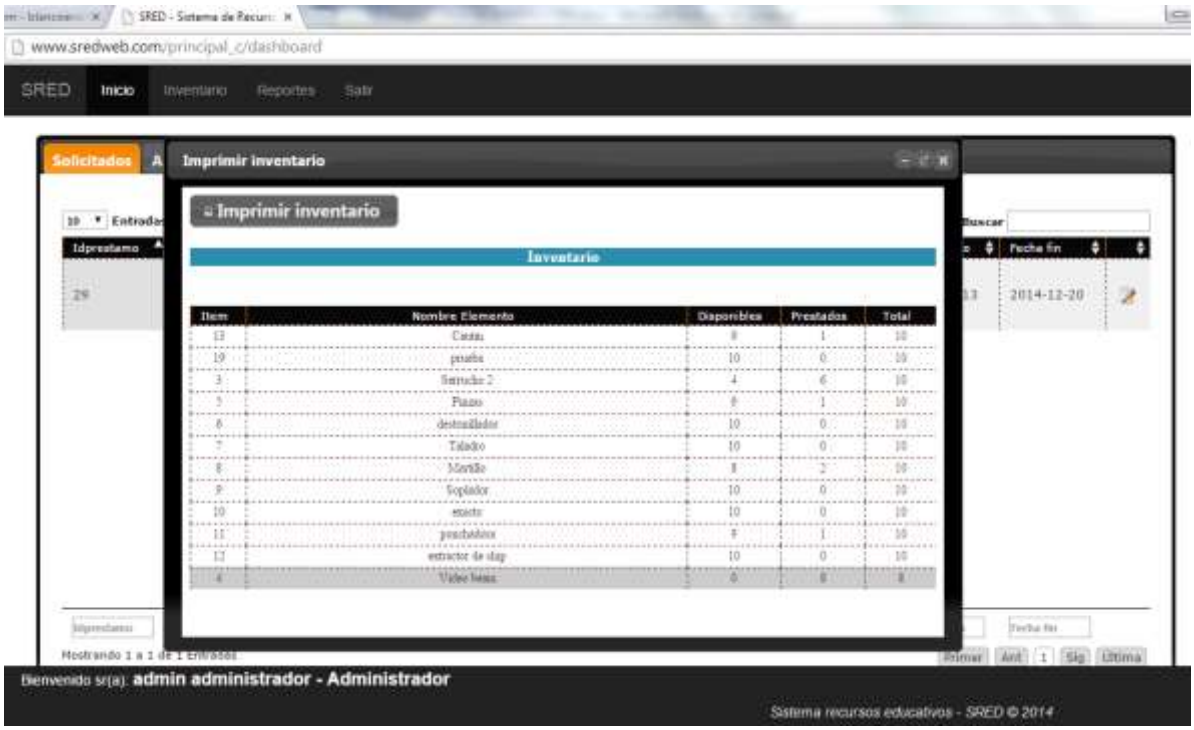

Figura 39. Interfaz: Inventario.

Fuente. Autor

En la ventana principal pulsar clic sobre el botón editar correspondiente a la solicitud número 29 de un video beam realizada por el usuario estudiante apestudiante, como podemos apreciar en la siguiente figura.

Figura 40. Interfaz: Usuario administrador.

Fuente. Autor.

En la ventana **Cambiar Estado** seleccione de la lista **Definir Estado de la solicitud** la opción **Negado**, como podemos apreciar en la siguiente imagen.

Figura 41. Interfaz: Usuario administrador.

Fuente. Autor

En la ventana **Cambiar Estado** digite en el cuadro de texto **Observaciones:** No hay video beam disponibles para apartar, luego pulse clic en el botón **Guardar Estado Solicitud**, como podemos apreciar en la siguiente imagen.

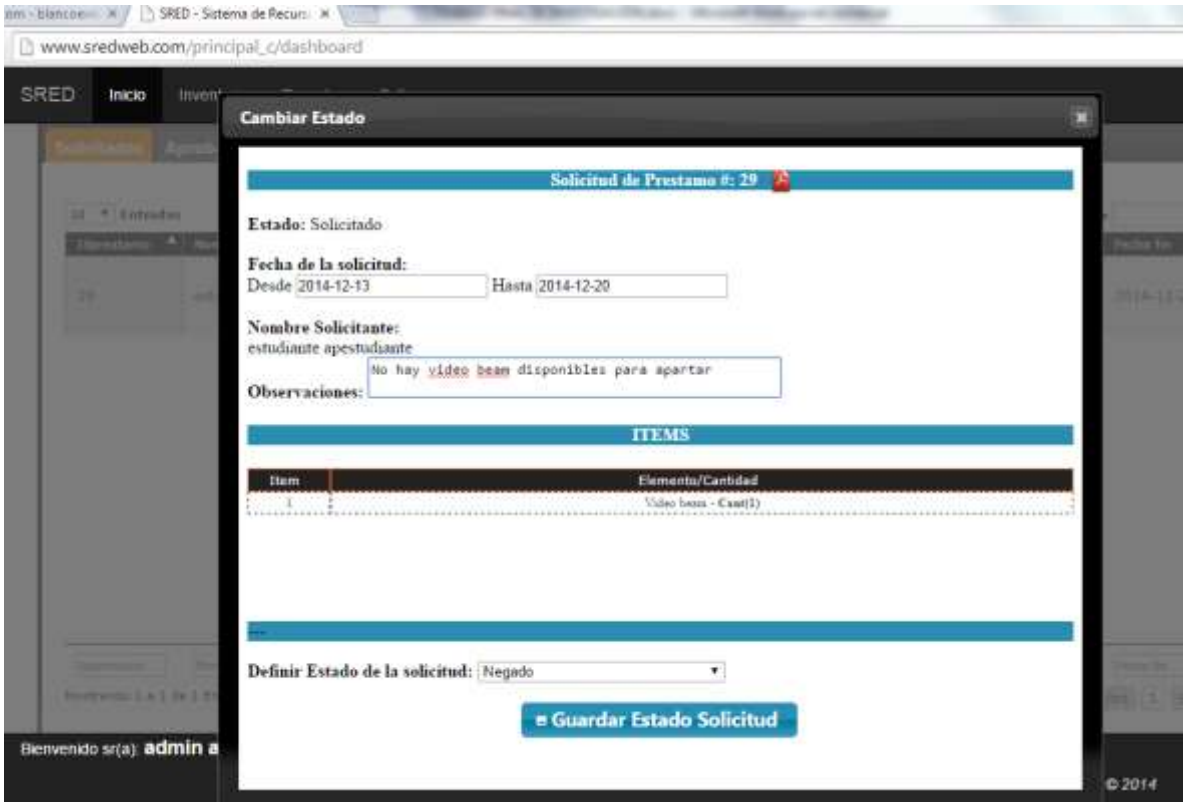

Figura 42. Interfaz: Usuario administrador.

Fuente. Autor.

En la ventana principal podemos ver como la solicitud número 29 de un video beam realizada por el usuario estudiante apestudiante paso del estado **Solicitados** al estado **Negados**, como podemos apreciar en la siguiente imagen.

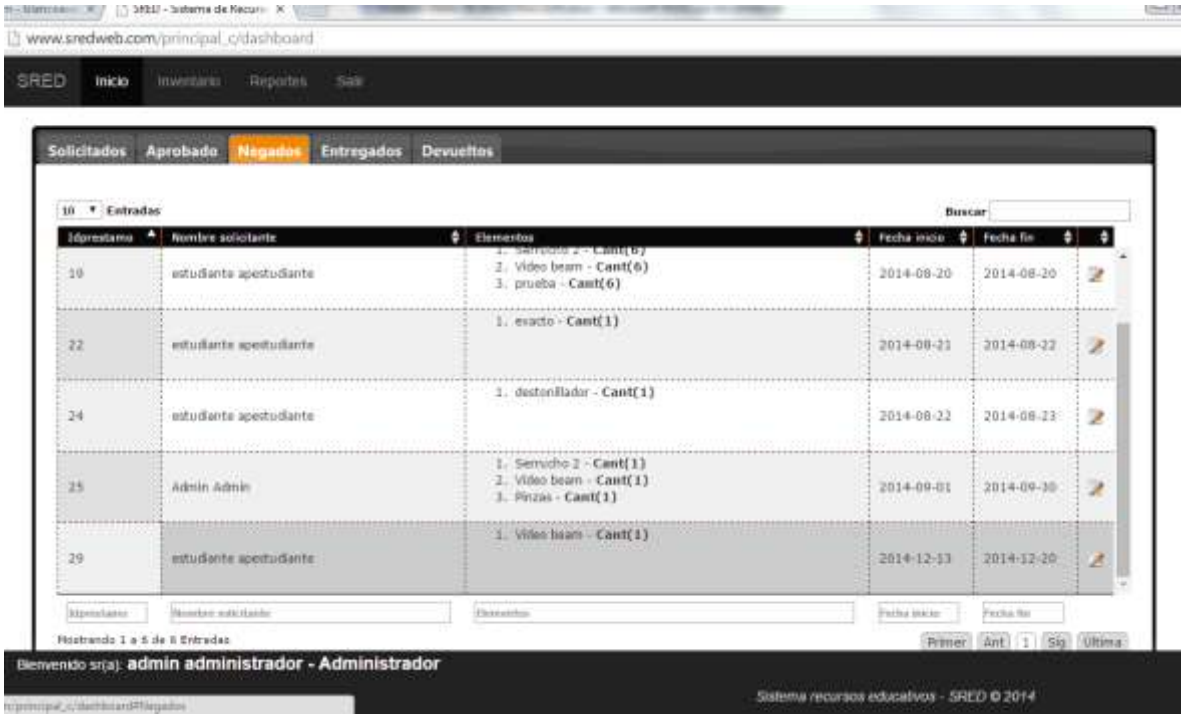

Figura 43. Interfaz: Usuario administrador.

Fuente. Autor.

5.1.3. VISTA DEL USUARIO NORMAL.

CASO DE USO: Realizar solicitudes.

En la ventana principal pulse clic sobre la opción de menú **Nueva solicitud**, como podemos apreciar en la siguiente imagen.

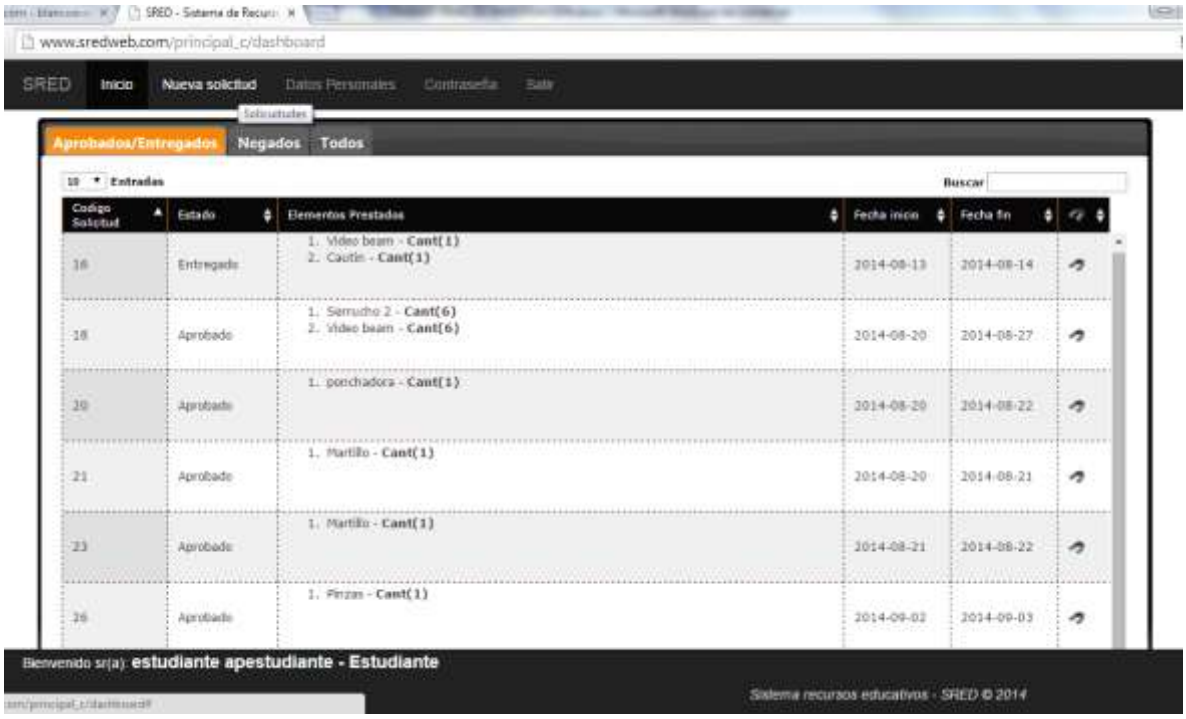

Figura 44. Interfaz: Usuario Normal.

Fuente. Autor.

En la ventana **Crear Nueva Solicitud** pulse clic sobre el cuadro de texto **Desde** y seleccione la fecha de inicio, como podemos apreciar en la siguiente imagen.

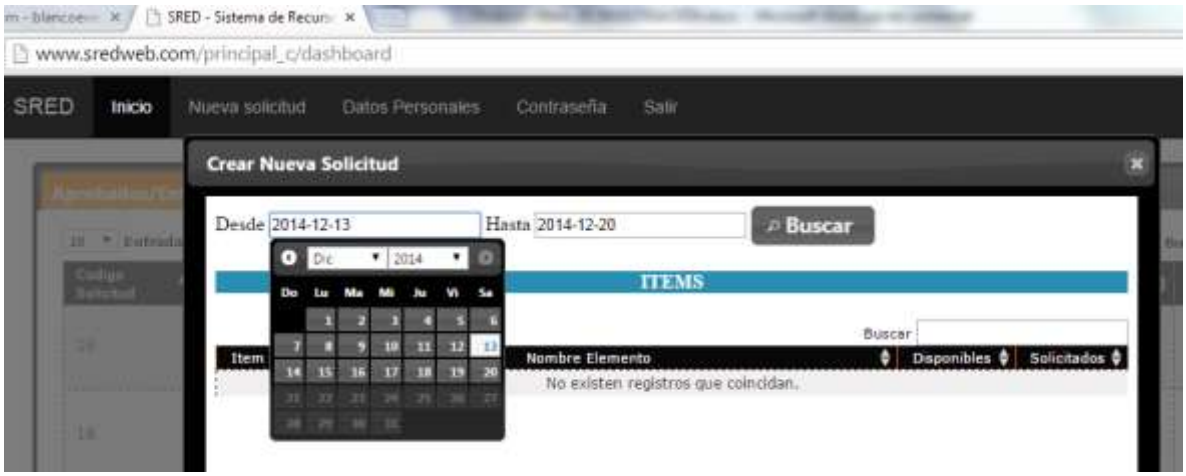

Figura 45. Interfaz: Crear Nueva Solicitud.

Fuente. Autor.

En la ventana **Crear Nueva Solicitud**, pulse clic sobre el cuadro de texto **Hasta** y seleccione la fecha final, como podemos apreciar en la siguiente imagen.

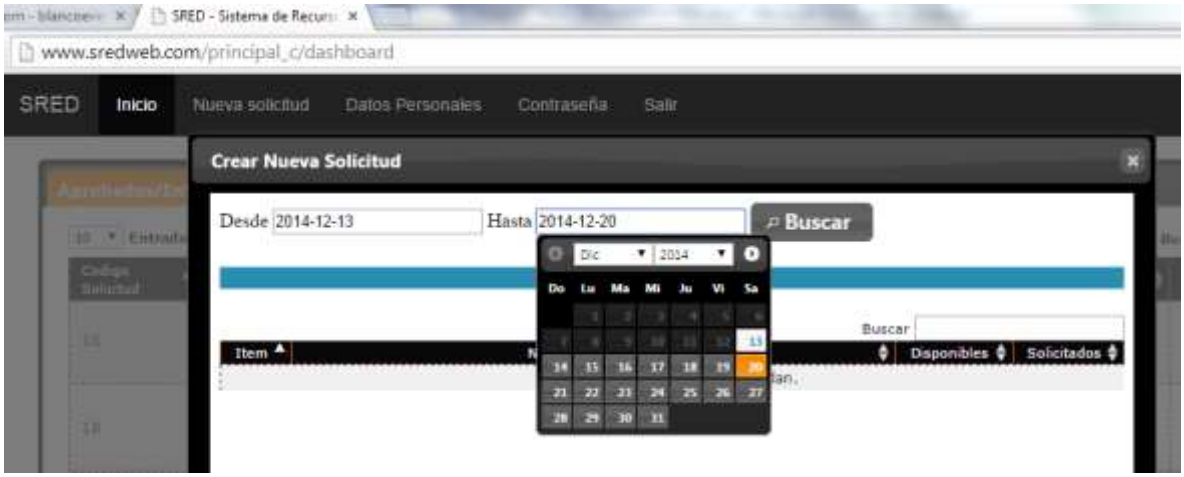

Figura 46. Interfaz: Crear Nueva Solicitud.

Fuente. Autor.

En la ventana **Crear Nueva Solicitud**, pulse clic sobre el botón **Buscar** para poder ver los elementos disponibles para apartar, como podemos apreciar en la siguiente imagen.

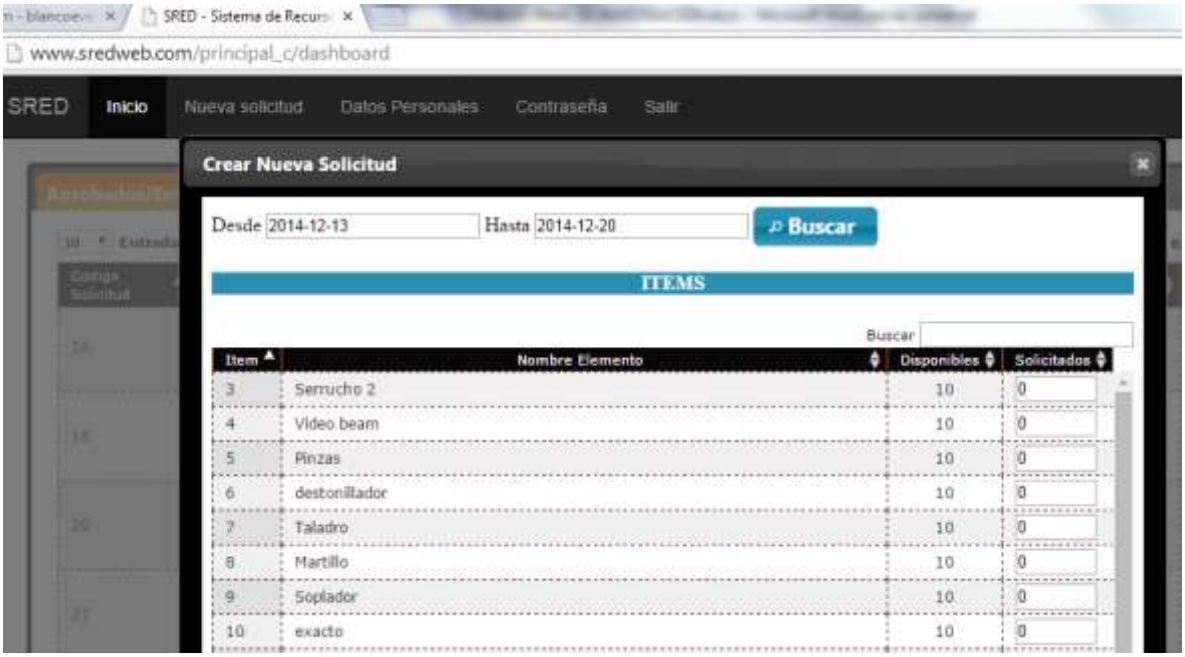

Figura 47. Interfaz: Crear Nueva Solicitud.

Fuente. Autor.

En la ventana **Crear Nueva Solicitud**, aparte el elemento Video beam digitando 1 en el cuadro **Solicitados**, como podemos apreciar en la siguiente imagen.

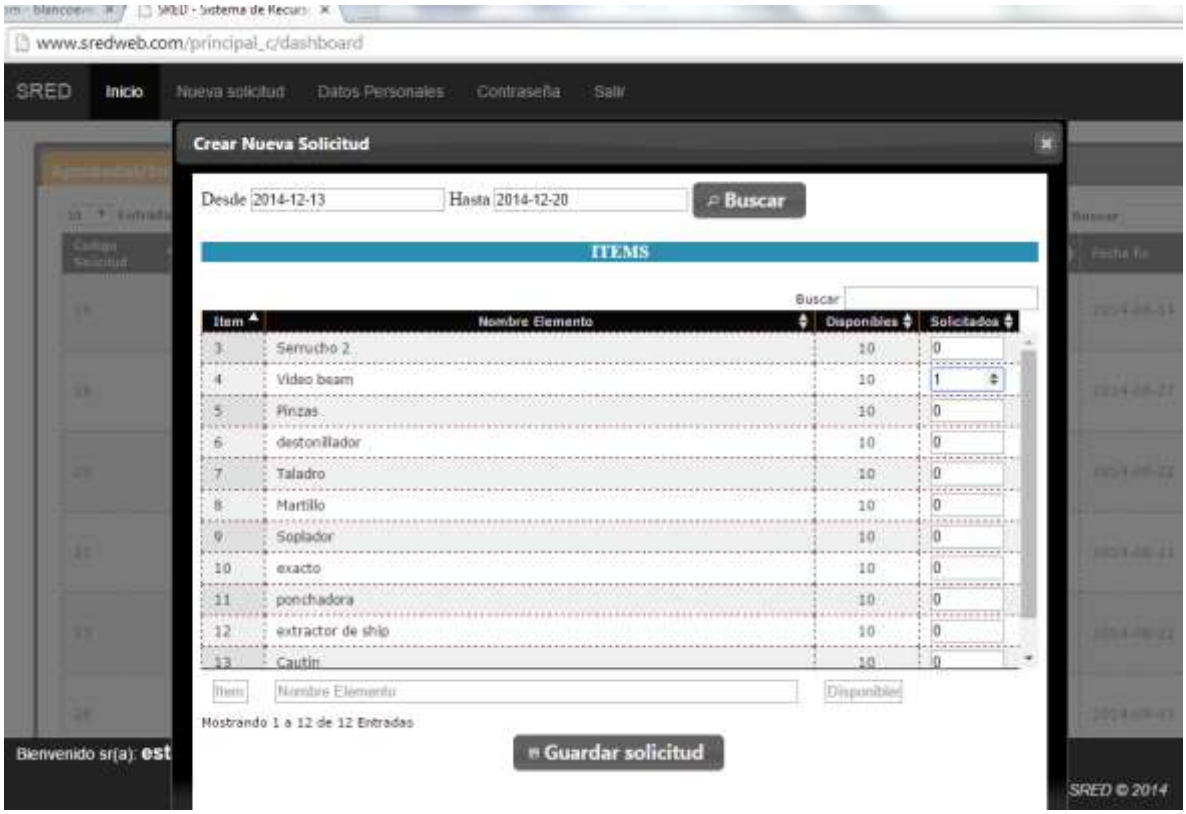

Figura 48. Interfaz: Crear Nueva Solicitud.

Fuente. Autor.

En la ventana **Crear Nueva Solicitud**, pulse clic sobre el botón **Guardar solicitud**, como podemos apreciar en la siguiente imagen.

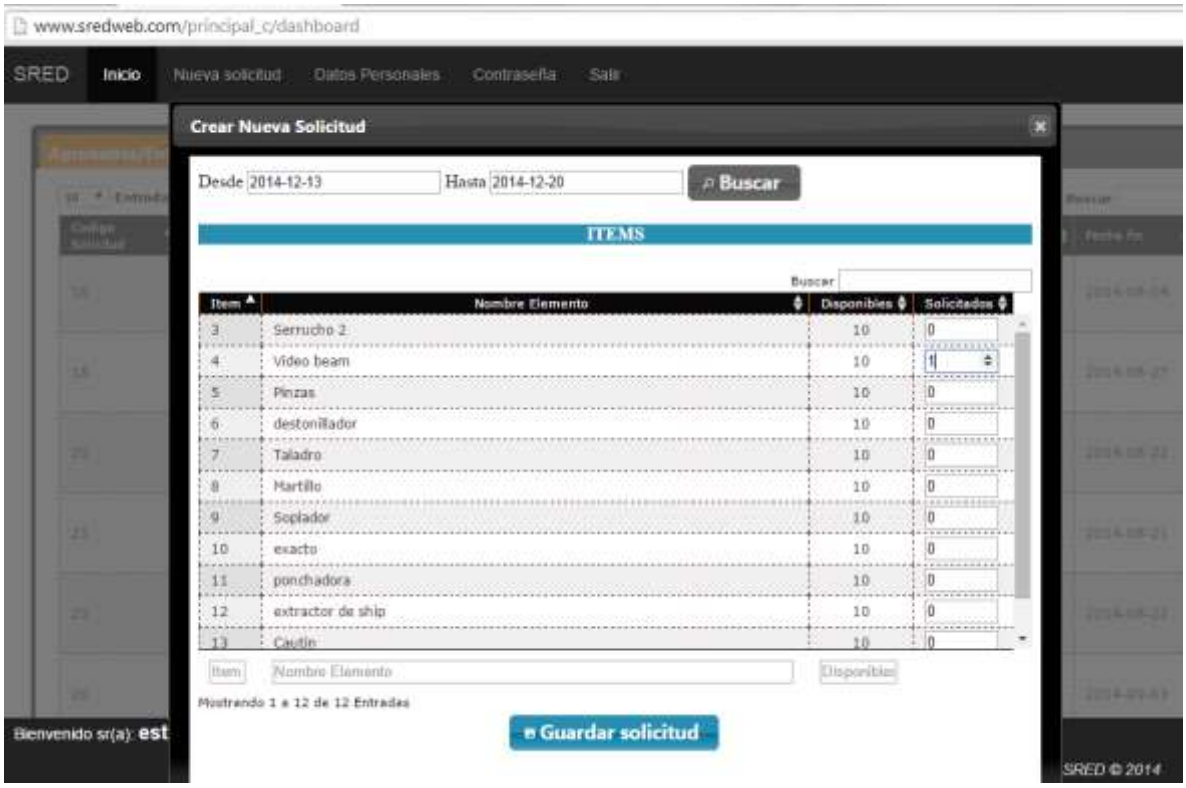

Figura 49. Interfaz: Crear Nueva Solicitud.

Fuente. Autor.

En la vista del usuario administrador en la ventana principal podemos ver que la solicitud de un video beam realizada en el punto anterior fue registrada satisfactoriamente, como podemos apreciar en la siguiente imagen.

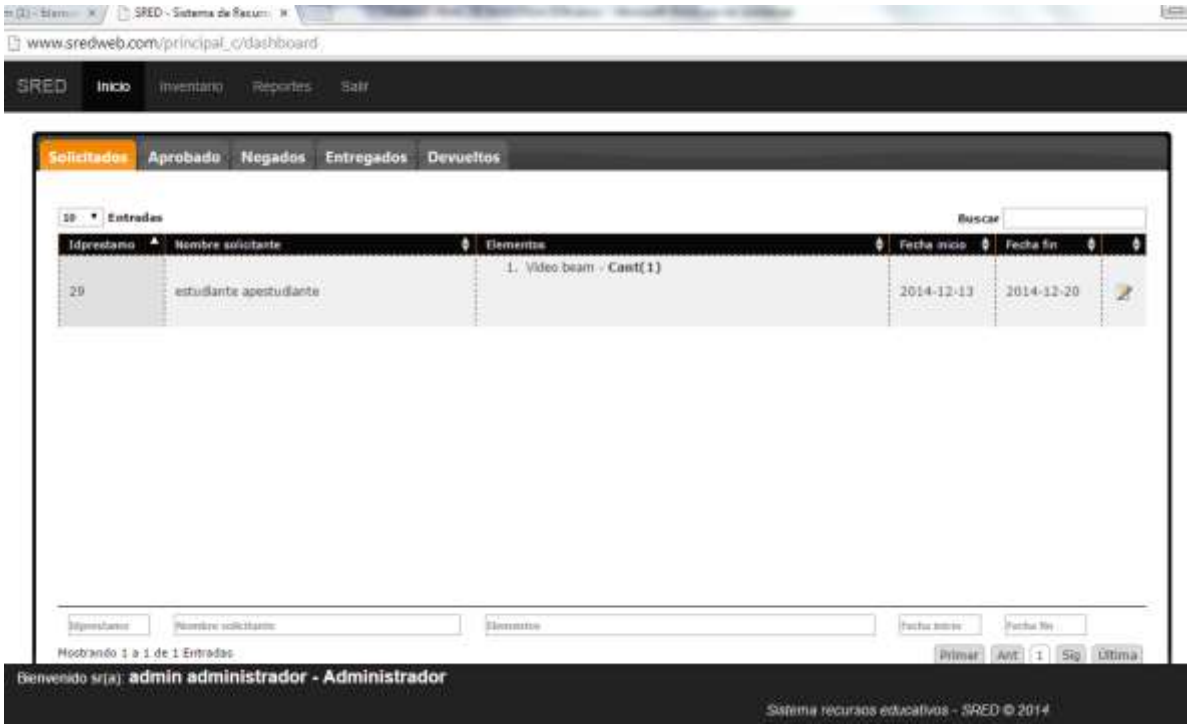

Figura 50. Interfaz: Solicitudes realizadas.

Fuente. Autor.

5.2. PRUEBAS DE USABALIDAD DEL SISTEMA.

La calidad del prototipo se evaluó según la Norma ISO/IEC 25010–Modelo de calidad (2005) en sus características de funcionalidad, y usabilidad en cada una de sus cuatro subcaracterísticas: Apropiabilidad, facilidad de aprendizaje, operabilidad y atractibilidad.

La metodología para la realización del test de usabilidad se hizo de forma similar al procedimiento hecho por Mercovich en el taller: Cómo hacer un test de usabilidad de un sitio [9]; la misma se presenta en las siguientes secciones:

- Planificación
- Test
- Análisis de los datos

Planificación

En esta etapa mediante un muestreo no probabilístico se procedió a seleccionar tres grupos representativos para los tres tipos de usuarios: Uno como súper usuario del sistema integrado por dos personas, otro como administrador del sistema integrado por dos personas y un tercer grupo como usuario común integrado por ocho personas.

Se diseñaron las tareas que serían realizadas por cada grupo de participantes acorde a su tipo de usuario; como por ejemplo realizar solicitud para el usuario común, aprobar y negar solicitudes para el usuario administrador y asignar permisos para el súper usuario.

Se definió que el sitio de aplicación de la prueba sería en los laboratorios de informática de la universidad de Cartagena realizando previamente la reservación de la sala.

Test

En el cuestionario se aplicaron los siguientes tipos de preguntas:

De afirmación con grado de aceptación de 1 a 5, significando que 1 está en completo desacuerdo con la afirmación de la pregunta y 5 está completamente de acuerdo con la afirmación de la pregunta.

De SI, NO; significando con SI que está completamente de acuerdo con la afirmación de la pregunta y NO que está en completo desacuerdo con la afirmación de la pregunta.

5.2.1. ENCUESTA DE USABILIDAD EN LA VISTA DEL ADMINISTRADOR.

1) ¿Entiende el propósito del sitio web?
   a) Si
   b) No

De la anterior pregunta el encuestado respondió SI.

Interpretación: El resultado fue favorable un 100%.

2) ¿Se sintió muy confiado en el manejo del sitio web?
   a) 1
   b) 2
   c) 3
   d) 4
   e) 5

De la anterior pregunta el encuestado respondió con grado de aceptación 4.

Interpretación: El resultado fue favorable un 100%.

3) ¿Para ser aprobada una solicitud el administrador debe primero?.
   a) Entregar el material
   b) Realizar la solicitud
   c) Verificar existencia en inventario

De la anterior pregunta el encuestado respondió verificar existencia en inventario.

Nota: La respuesta correcta es, verificar existencia en inventario.

Interpretación: La respuesta fue correcta en un 100%.

Para la subcaracterística **OPERABILIDAD:**

4) ¿Encontró demasiada inconsistencia en el sitio web?
a) 1
b) 2
c) 3
d) 4
e) 5

De la anterior pregunta el encuestado estuvo en completo desacuerdo calificando con 1.

Interpretación: El resultado fue totalmente favorable en un 100%.

5) ¿Encontró el sitio web muy grande al recorrerlo?
a) 1
b) 2
c) 3
d) 4
e) 5

De la anterior pregunta el encuestado estuvo en completo desacuerdo calificando con 1.

Interpretación: El resultado fue totalmente favorable en un 100%.

6) ¿Puede reubicarse fácilmente después de haber ocurrido un error?
a) Si
b) No

En la anterior pregunta el encuestado respondió SI.

Interpretación: El resultado fue favorable un 100%.

7) ¿Considera usted que el sistema mejora el proceso manual actual que se está llevando, y por qué?.
a) Si
b) No

En la anterior pregunta el encuestado respondió SI.

Interpretación: El resultado fue favorable un 100%.

Nota: Las respuestas del por qué, fue: La sistematización en el servicio.

Para la subcaracterística **ATRACTIBILIDAD:**

8) ¿Le gustará visitar con frecuencia este sitio web?
   a) 1
   b) 2
   c) 3
   d) 4
   e) 5

De la anterior pregunta el encuestado respondió con grado de aceptación 4.

Interpretación: El resultado fue favorable un 100%.

9) ¿El software permite realizar múltiples acciones a la vez?
   a) Si
   b) No

En la anterior pregunta el encuestado respondió SI.

Interpretación: El resultado fue favorable un 100%.

10) ¿El software permite finalizar una acción incompleta descartando cualquier cambio?
   a) Si
   b) No

En la anterior pregunta el encuestado respondió SI.

Interpretación: El resultado fue favorable un 100%.

Para la subcaracterística **FACILIDAD DE APRENDIZAJE:**

11) ¿Encontró el sitio web innecesariamente complejo?
    a) 1
    b) 2
    c) 3
    d) 4
    e) 5

De la anterior pregunta el encuestado estuvo en completo desacuerdo calificando con 1.

Interpretación: El resultado fue totalmente favorable en un 100%.

12) ¿Piensa que es fácil utilizar el sitio web?
    a) 1
    b) 2
    c) 3
    d) 4
    e) 5

De la anterior pregunta el encuestado respondió con grado de aceptación 4.

Interpretación: El resultado fue favorable un 100%.

13) ¿Necesito aprender muchas cosas antes de manejarme en el sitio web?
    a) 1
    b) 2
    c) 3
    d) 4
    e) 5

De la anterior pregunta el encuestado estuvo en completo desacuerdo calificando con 1.

Interpretación: El resultado fue totalmente favorable en un 100%.

5.2.2. ENCUESTA DE USABILIDAD EN LA VISTA DEL USUARIO NORMAL.

Para la subcaracterística **APROPIABILIDAD:**

1) Cuál es la precondición del caso de uso Nueva Solicitud que se debe ingresar para poder realizar la misma.
a) No tiene precondición
b) Ingresar un rango de fechas
c) Ingresar como usuario del sistema

De la anterior pregunta 6 encuestados respondieron ingresar un rango de fechas, 2 encuestados respondieron ingresar como usuario del sistema.

Nota: La respuesta correcta es, ingresar un rango de fechas.

Interpretación: El 75% de los encuestados respondieron correctamente, y el 25% de los encuestados respondió incorrectamente.

2) ¿Entiende el propósito del sitio web?
a) Si
b) No

De la anterior pregunta 8 encuestados respondieron SI, 0 encuestados respondieron NO.

Interpretación: El resultado fue favorable un 100%.

3) ¿Se sintió muy confiado en el manejo del sitio web?
a) 1
b) 2
c) 3
d) 4
e) 5

De la anterior pregunta 5 encuestados calificaron con grado de aceptación 5, y 3 encuestados calificaron con grado de aceptación 4.

Interpretación: El resultado fue completamente favorable un 62,5% y favorable un 37,5%.

Para la subcaracterística **OPERABILIDAD:**

4) Como considera que fue la velocidad del software
a) Mala
b) Regular
c) Buena
d) Excelente

De la anterior pregunta 6 encuestados calificaron con grado excelente, y 2 encuestados calificaron con grado buena.

Interpretación: El resultado fue excelente en un 75%, y bueno en un 25%.

5) ¿Le fue difícil realizar las tareas asignadas dentro del sistema?
a) Si
b) No

De la anterior pregunta 8 encuestados respondieron NO, 0 encuestados respondieron SI.

Interpretación: El resultado fue favorable un 100%.

6) ¿Encontró las diversas posibilidades del sitio web bastante bien integradas?
a) 1
b) 2
c) 3
d) 4
e) 5

De la anterior pregunta 5 encuestados calificaron con grado de aceptación 5, y 3 encuestados calificaron con grado de aceptación 4.

Interpretación: El resultado fue completamente favorable en un 62,5% y favorable un 37,5%.

7) ¿Considera usted que el sistema mejora el proceso manual actual que se está llevando, y por qué?.
c) Si
d) No

De la anterior pregunta 8 encuestados respondieron SI, 0 encuestados respondieron NO.

Interpretación: El resultado fue favorable un 100%.

Nota: Las respuestas del por qué, fueron: La comodidad de poder realizar una solicitud desde la casa, la organización en el servicio.

Para la subcaracterística **ATRACTIBILIDAD:**

8) ¿La interacción que tuvo con la interfaz fue agradable?
a) Si
b) No

De la anterior pregunta 8 encuestados respondieron SI, 0 encuestados respondieron NO.

Interpretación: El resultado fue favorable un 100%.

9) ¿Le gustará visitar con frecuencia este sitio web?

a) 1
b) 2
c) 3
d) 4
e) 5

De la anterior pregunta 6 encuestados calificaron con grado de aceptación 5, y 2 encuestados calificaron con grado de aceptación 4.

Interpretación: El resultado fue completamente favorable en un 75% y favorable un 25%.

10) ¿El software permite finalizar una acción incompleta descartando cualquier cambio?

a) Si
b) No

De la anterior pregunta 8 encuestados respondieron SI, 0 encuestados respondieron NO.

Interpretación: El resultado fue favorable un 100%.

Para la subcaracterística **FACILIDAD DE APRENDIZAJE:**

11) ¿Piensa que es fácil utilizar el sitio web?

a) 1
b) 2
c) 3
d) 4
e) 5

De la anterior pregunta 5 encuestados calificaron con grado de aceptación 5, y 3 encuestados calificaron con grado de aceptación 4.

Interpretación: El resultado fue completamente favorable un 62,5% y favorable un 37,5%.

12) ¿Imagina que la mayoría de las personas aprenderían muy rápidamente a utilizar el sitio web?

a) 1
b) 2
c) 3
d) 4
e) 5

De la anterior pregunta 5 encuestados calificaron con grado de aceptación 5, y 3 encuestados calificaron con grado de aceptación 4.

Interpretación: El resultado fue completamente favorable un 62,5% y favorable un 37,5%.

13) ¿Necesita aprender muchas cosas antes de manejar el sitio web?

a) 1
b) 2
c) 3
d) 4
e) 5

De la anterior pregunta 8 encuestados estuvieron en completo desacuerdo calificando con 1.

Interpretación: El resultado fue totalmente favorable en un 100%.

5.2.3. RESUMEN ANALITICO DE LA PRUEBA DE USABILIDAD.

El test de usabilidad aplicado a los tres tipos de usuarios para la subcaracterística de Apropiabilidad arrojo los siguientes resultados:

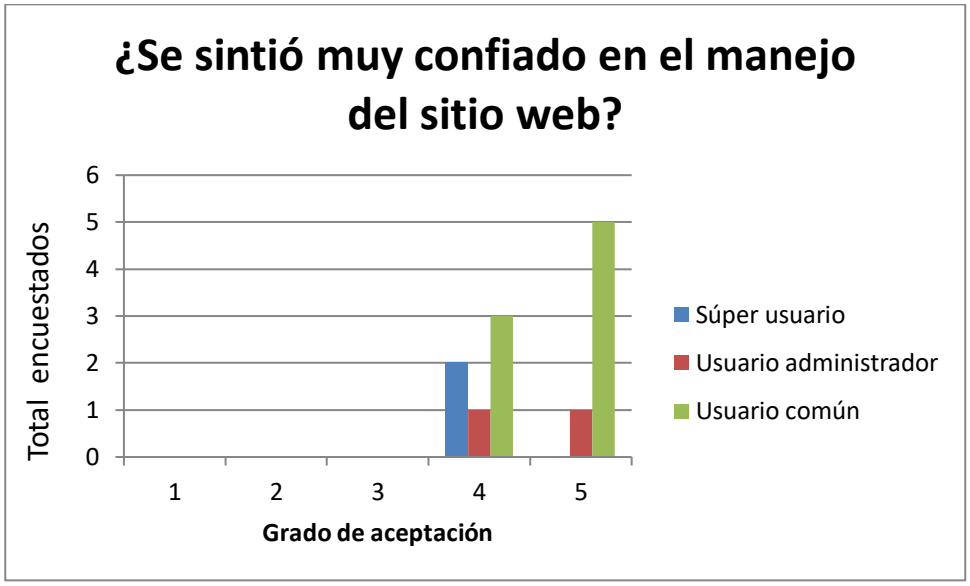

FIGURA 51. Resultados para la subcaracterística de Apropiabilidad

Definidos en el anterior gráfico el cual expresa que los súper usuarios ubicaron su grado de aceptación en el nivel 4, los usuarios administradores ubicaron su grado de aceptación en los niveles 4 y 5, 5 de los usuarios comunes ubicaron su grado de aceptación en el nivel 5, y 3 ubicaron su grado de aceptación en el nivel 4; se puede observar un alto grado de apropiabilidad en el sistema por parte de los usuarios encuestados ya que sus respuestas fueron ubicadas en los niveles 4 y 5 que son los grados más altos de aceptación.

El test de usabilidad aplicado a los tres tipos de usuarios para la subcaracterística Facilidad de aprendizaje arrojo los siguientes resultados:

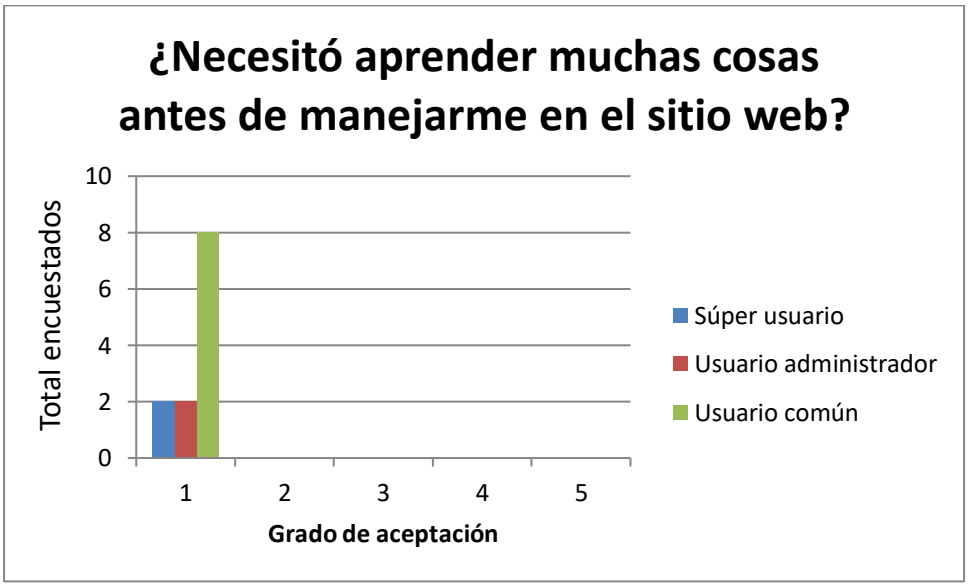

FIGURA 52. Resultados para la subcaracterística de Facilidad de aprendizaje

En esta grafica se observa como los súper usuarios consideran que no necesitan aprender muchas cosas para manejar el sistema, lo mismo consideraron los usuarios administradores y los usuarios comunes, ubicando sus respuestas en el nivel 1 que es el grado de aceptación más bajo a la afirmación que involucra tener que aprender muchas cosas para poder manejar el sitio web.

El test de usabilidad aplicado a los tres tipos de usuarios para la subcaracterística de Operabilidad arrojo los siguientes resultados:

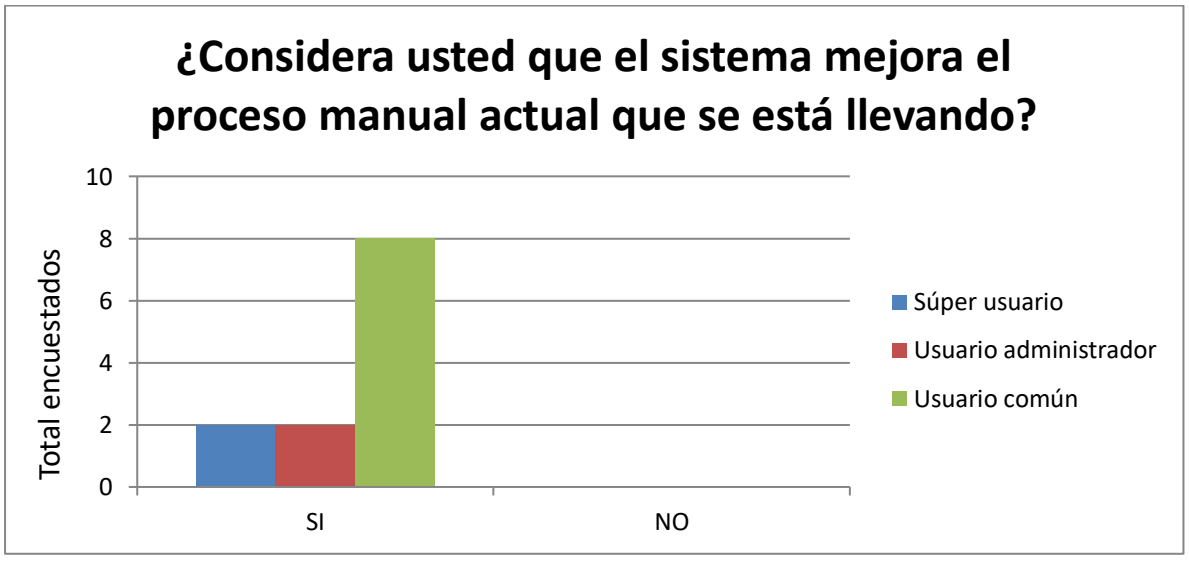

FIGURA 53. Resultados para la subcaracterística de Operabilidad

Las respuestas de los tres tipos de usuarios fue de conformidad con respecto a la afirmación de que el sistema implementado mejora el proceso manual que hasta entonces se estaba realizando.

El test de usabilidad aplicado a los tres tipos de usuarios para la subcaracterística de Atractibilidad arrojo los siguientes resultados:

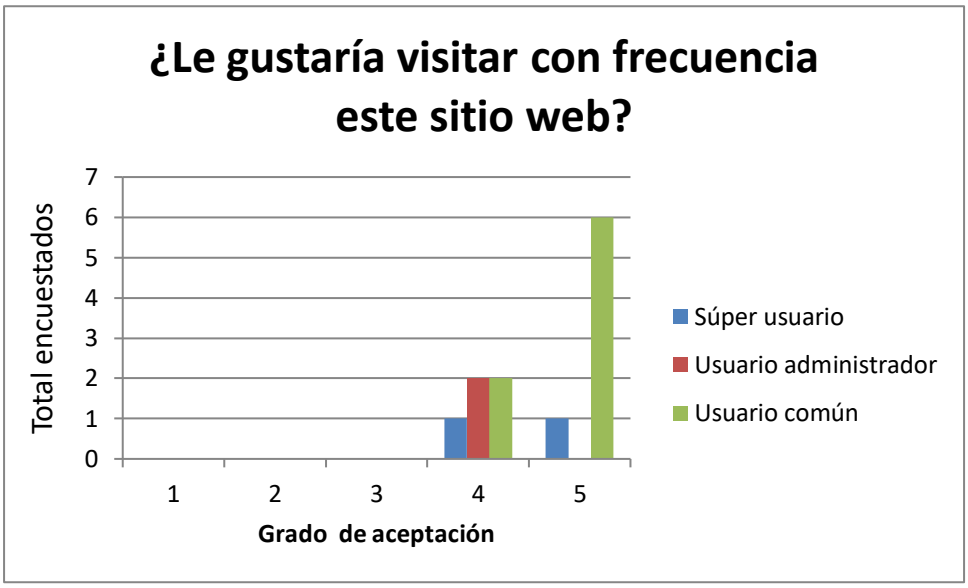

FIGURA 54. Resultados para la subcaracterística de Atractibilidad

El test arrojo altos índice en el gusto por la interacción de los usuarios, donde se puede apreciar que los súper usuarios ubicaron su grado de aceptación en el nivel 4 y 5, los usuarios administradores ubicaron su grado de aceptación en el nivel 5, 2 de los usuarios comunes ubicaron su grado de aceptación en el nivel 4, y 6 ubicaron su grado de aceptación en el nivel 5; se puede observar un alto grado de atractibilidad en el sistema por parte de los usuarios encuestados ya que sus respuestas fueron ubicadas en los niveles 4 y 5 que son los grados más altos de aceptación.

6. CONCLUSIONES.

En la ingeniería del software la elección de una metodología es fundamental para la gestión y administración de un proyecto, de esta depende que el mismo tenga viabilidad o fracase, la metodología a utilizar va acorde al problema a resolver lo mismo que al tipo de empresa, como ejemplo si es una empresa grande o pequeña, al seleccionar correctamente la metodología a aplicar la misma traza un camino recto en el proceso de desarrollo evitando en lo posible encontrar problemas en la realización del proyecto, de esta manera se pueden gestionar cada una de las etapas del proceso de desarrollo como partes de un engranaje y darles un tratamiento especial a cada una de ellas aplicándoles técnicas, métodos y utilizando herramientas específicas para cada etapa. La especificación de requerimientos es una de las etapas más importante en la construcción de cualquier sistema permite hacer gestión de tiempos, costos y por ende con base en estas variables la asignación de recursos dando viabilidad al proyecto. Además los requerimientos son un reflejo de las necesidades de los usuarios del sistema y por tanto deben quedar completos y bien especificados. Aplicando ingeniería de requerimientos se pudo determinar que los aplicativos actualmente desarrollados y que se relacionan con nuestro sistema no satisfacen la gran mayoría de nuestras necesidades y por tanto era más sencillo la implementación desde cero de un nuevo sistema que la adaptación de cualquier sistema al nuestro; de esta manera se dio solución a todas las necesidades de forma eficiente y eficaz, logrando una mejora notable en los procesos internos de la Universidad de Cartagena.

7. RECOMENDACIONES Y TRABAJOS FUTUROS.

- Para que la aplicación tenga una completa utilidad, es recomendable que la base de datos utilizada en el programa de gestión de préstamo de materiales sea enlazada con la base de datos del departamento de admisiones y registro académico de la Universidad de Cartagena, para poder obtener datos como: promedio, materias cursadas, condicionamiento académico, verificar si el estudiante está activo, etc.

- Se sugiere definir las políticas de asignación de recursos, dado que estas afectan las estructuras de datos y entidades de nuestro sistema.

- En la actualidad los aplicativos para dispositivos móviles son de uso común en la mayoría de personas, por esta razón se hace necesario el desarrollo de un sistema para este tipo de tecnología.

8. GLOSARIO.

ARQUITECTURA MODELO VISTA CONTROLADOR: MVC consiste de tres tipos de objetos. El Modelo, que son los objetos de la aplicación, también conocida como lógica de negocio, o lógica de aplicación. La Vista especifica la visualización de los datos, algunas veces conocida como lógica de presentación. El controlador es el coordinador entre estos dos últimos, es decir, define la forma en que la interfaz de usuario reacciona ante la entrada de usuario. MVC desacopla el concepto de interfaz de usuario y lógica de negocio para aumentar la flexibilidad y modularidad del software, posiblemente permitiendo que el código pueda ser reutilizado.

CGI: (*Common Gateway Interface*). Es una norma para establecer comunicación entre un servidor Web y un programa, de tal modo que este último pueda interactuar con Internet. También se usa la palabra CGI para referirse al programa mismo, que se ejecuta en tiempo real en un Web Server en respuesta a una solicitud de un navegador.

COOKIE: Es un pequeño documento en formato de texto que es grabado y acogido por el disco duro del computador del usuario. Se utiliza para mantener el estado de una aplicación o seguir la trayectoria del usuario en el sitio.

DOCUMENTO PDF: (*Portable Document Format*, formato de documento portátil). Es un formato de almacenamiento de documentos, desarrollado por la empresa Adobe Systems. Este formato es de tipo compuesto (imagen vectorial, mapa de bits y texto).

ESCALABILIDAD: Posibilidad de aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado (o mejorado) en cualquier momento, o se pueden añadir nuevos nodos a la red (clientes y/o servidores).

HIPERTEXTO: Cualquier texto disponible en el World Wide Web que contenga enlaces con otros documentos.

HTML: (*HiperText Markup Lenguaje,* Lenguaje de Marcado de Hipertexto). Lenguaje empleado para describir el interior de los documentos Web, basado en el uso de etiquetas. Permite describir hipertexto con enlaces (hiperlinks) que conducen a otros documentos o fuentes de información relacionadas y con inserciones multimedia (gráficos, sonido…).

HTTP: (*HiperText Transfer Protocol,* Protocolo de Transferencia de Hipertexto). Lenguaje empleado para describir cómo se envían los documentos HTML por Internet. HTTP proporciona las normas para que los navegadores hagan peticiones y los servidores entreguen respuestas.

INTERNET: Red global de comunicaciones que interconecta computadoras y bases de datos distribuidas por todo el planeta.

IP: (*Internet Protocol).* Protocolo que provee las funciones básicas de direccionamiento en Internet y en cualquier red TCP/IP (software de comunicación). El protocolo de Internet se encarga de poner una etiqueta con la dirección adecuada a cada paquete, ya que cada computador conectado a la red tiene una dirección de Internet única que lo distingue de cualquier otro computador en el mundo.

LINUX: S*istema operativo*. Es una implementación de libre distribución UNIX para computadores personales, servidores y estaciones de trabajo. Consta de componentes GNU y el kernell desarrollado por Linux Torvalds.

PÁGINA WEB: Servicio de Internet que permite el hipertexto (permite ir de una página a otra enlazando el hipermedia). Presenta documentos con texto, imagen estática y en movimiento, sonido, video, etc. y utiliza el estándar HTML.

PÁGINA WEB DINÁMICA: Página Web cuyo contenido es calculado por el servidor en el momento en que el usuario accede a ella. Normalmente el contenido se obtiene desde una base de datos.

PÁGINA WEB ESTÁTICA: Página Web con textos y otro tipo de archivos (imágenes, multimedia, etc.) que contiene toda la información necesaria y se muestra al tiempo que es solicitada.

PORTABLE: La portabilidad de un software se define como su grado de dependencia de la plataforma en la que corre. La portabilidad es mayor cuanto menor es su dependencia del software de plataforma.

SPAM: Se llama así al "bombardeo" con correo electrónico, es decir, mandar grandes cantidades de correo o mensajes muy largos. Correo basura no solicitado.

SCRIPT: Programa escrito en un lenguaje específico de programación que tiene una serie de instrucciones y normalmente funciona sobre otras aplicaciones que ya están en funcionamiento.

SERVIDOR WEB: Servidor que almacena las páginas de un sitio Web y envía páginas Web en respuesta a la peticiones HTTP hechas desde los navegadores de los clientes.

UML: (*Unified Modeling Language,* Lenguaje de Modelamiento Unificado). Es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML entrega una forma de modelar cosas

conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables.

URL: (*Uniform Resource Locator,* Localizador Uniforme de Recursos). Cadena de caracteres que definen la localización y el acceso a documentos de hipertexto o programas en Internet. Un URL está formado de la siguiente manera: Esquema://máquina/ruta.

WWW: (*World Wide Web).* Sistema de arquitectura Cliente/Servidor para distribución y obtención de información en Internet, basada en hipertexto e hipermedia.

9. BIBLIOGRAFÍA.

Alarcón, R. (2000). Diseño orientado a objetos con UML. Madrid: Grupo EIDOS.

Carcamo Sepulveda, J. (1997). *BASES DE DATOS RELACIONALES : UN ENFOQUE PRACTICO DE DISENO.* Bucaramanga: Universidad Industrial de Santander.

Kimmel, P. (2007). Manual de UML. México: McGraw Hill.

PIATTINI, M., CALVO-MANZANO, J., & CERVERA, J. y. (2004). Análisis y diseño de aplicaciones informáticas de gestión. México: Alfaomega, Ra-Ma.

Stallings, W. (2000). Sistemas Operativos. Madrid: Prentice Hall.

Date, C.J. (2001). Introducción a los sistemas de bases de datos (7ª ed.). Prentice Hall.

Silberschatz, A; Korth, H.F; Sudarshan, S. (1998). Fundamentos de bases de datos (3.a ed.). Madrid: McGraw-Hill.

ROGER S. PRESMAN. Ingeniería de software. Un enfoque Práctico. Editorial Mc. Graw Hill.

LARMAN, Craig. UML y Patrones. Prentice Hall.

SOMMER. Ingenieria de Software. 6 edición. 2002.

L. BASS, P. CLEMENS, R. KAZMAN, ADDISON WESLEY. Software Architecture in Practice.

SHARI L. PFLEEGER. Software Engineering the Producción of Quality Software. Macmillan Publishing Company.

G. BOOCH, J. RUMBAUGH, I. JACOBSON. ADDISON-WESLEY. El Lenguaje Unificado de Modelado. Iberoamericana.