UNIVERSIDAD NACIONAL JOSÉ MARÍA ARGUEDAS

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

I N F O R M E D E T E S I S

RESPONSIVE WEB DESIGN PARA PLANIFICACIÓN Y RECOMENDACION TURÍSTICA APLICANDO INTELIGENCIA COLECTIVA

LÍNEA DE INVESTIGACIÓN	:	GESTIÓN DE LA INFORMACIÓN Y EL CONOCIMIENTO
ÁREA PRIORIZADA PNCYT	:	01060002
AUTOR	:	JAINE ALEXIA MONZÓN FLORES
ASESOR	:	ING. IVÁN SORIA SOLÍS

ANDAHUAYLAS – APURÍMAC

PERÚ

2015

DEDICATORIA

Con todo mi cariño y amor a mi madre, por estar conmigo siempre, por enseñarme a crecer y a que si caigo debo levantarme, por apoyarme y guiarme, por ser la base que me ayudo a llegar hasta aquí, a mis hermanas por su amor incondicional y por entender mi estado de ánimo tan cambiante, por siempre mi corazón y mi agradecimiento.

Jainé Alexia

AGRADECIMIENTO

A mi madre por darme la vida y cuidarme.

A mis hermanas por su apoyo incondicional.

A mis profesores por ser mis segundos padres, contribuyendo siempre con mi superación profesional y moral, y en especial al Ingeniero Edwin Octavio Ramos Velasquez, quien hoy no se encuentra entre nosotros pero a quien llevamos siempre presente, por su incansable labor en lograr el éxito profesional en sus estudiantes.

Muchas gracias.

“A veces podemos pasarnos años sin vivir… en absoluto, y de pronto toda nuestra vida… se concentra en un solo instante”

TABLA DE CONTENIDOS

LISTA DE TABLAS

LISTA DE GRÁFICOS

RESUMEN

El objetivo general de la investigación tecnológica, fue, el de desarrollar una idea original de desarrollo web dedicado al turismo en Andahuaylas, aplicando teorías de inteligencia colectiva que se caracteriza como aquello que supera el pensamiento individual, y filtrado colaborativo que consiste en recomendar elementos que han gustado a usuarios con preferencias similares, además se caracteriza por poseer un entorno RWD (Responsive Web Desing, Diseño web Adaptable), esto quiere decir que la aplicación tendrá la capacidad de adaptarse a cualquier dispositivo móvil (Smartphone, Tablet, laptop, y otros).

La metodología de desarrollo de software que fue empleada para el desarrollo de esta aplicación web RWD, fue, la metodología Ágil XP, ya que nos permitió tener una constante comunicación con todos los integrantes del equipo del proyecto, esto nos permitió realizar todas modificaciones necesarias hasta conseguir el producto esperado.

Para el desarrollo de la aplicación RWD, hicimos uso de Frameworks, que agilizaron la programación y diseño de interfaces a todo el equipo de trabajo.

Como gestor de base de datos para la aplicación web RWD, tenemos el MySql, que es uno de los sistemas más usados para este tipo de proyectos de software.

Esta investigación es de Tipo Aplicada, ya que tiene la orientación se ser una web social, y al mismo tiempo una página institucional.

Palabras claves: Inteligencia Colectiva, Filtrado Colaborativo, Frameworks, Responsive Web Desing, Pearson.

ABSTRACT

The overall objective of the technological research was to develop an original idea dedicated to tourism development applying theories of collective intelligence characterized as that which exceeds the individual thinking, and collaborative filtering is to recommend items that have liked to users with similar preferences, further characterized by having a setting RWD (Responsive Web Desing, Web design Adaptable), this means that the application will have the ability to fit any mobile device (Smartphone, Tablet, laptop, etc.).

The methodology for developing software that was used to develop this RWD web application was, methodology Agile XP, and that allowed us to have constant communication with all members of the project team, this allowed us to make all necessary modifications to get the expected product.

For RWD application development, we made use of Frameworks, which streamlined programming and interface design to the whole team.

As database manager for RWD web application, we have the MySQL, which is one of the systems more used to this type of software projects data.

This type Applied research is because it has a social web guidance is being, and at the same time an institutional page.

Keywords: Collective Intelligence, Collaborative Filtering, Frameworks, Responsive Web Design, Pearson.

INTRODUCCIÓN

El Turismo se ha convertido en una de las actividades económicas más importante de nuestro país. El Turismo no sólo impacta al propio sector económico, sino que además influye fuertemente en los resultados de otros sectores, en un aumento de la calidad de vida de nuestros habitantes y una mantención de nuestros atractivos naturales y su conservación como medio ambiente limpio y puro.

En la actualidad existen un sin número de sectores en donde no son muy bien aprovechados los recursos turísticos, un ejemplo se da en la región de Apurímac, Provincia de Andahuaylas, en donde existe una variedad de recursos turísticos y culturales.

El Internet es una herramienta que nos permite conectarnos con el mundo entero a través de los portales web, páginas web y aplicaciones web.

La implementación de esta aplicación Responsive Web Desing para la planificación y recomendación turística, permite que el usuario pueda interactuar con la aplicación web en cualquier momento y desde donde se encuentre, y de esa manera pueda acceder , navegar, conocer, entre otros. Los recursos turísticos y culturales de una localidad, y al mismo tiempo también planificar visitas y recomendárselas a otros usuarios.

El informe del proyecto está organizado de la 0siguiente manera:

CAPÍTULO I: PLANTEAMIENTO DEL PROBLEMA DE ESTUDIO. En el que se exponen la fundamentación, descripción de la realidad problemática, formulación del problema, objetivos, justificación viabilidad y limitaciones.

CAPÍTULO II: ESTADO DEL ARTE. En esta parte exponemos los antecedentes de la investigación.

CAPÍTULO III: MARCO TEÓRICO. En esta parte se exponen todo lo relacionado con el marco teórico teniendo en cuenta investigaciones importantes de autores nacionales e internacionales que constituyeron la base científica de la investigación.

CAPÍTULO IV: INGENIERIA DEL PROYECTO. En esta parte, se describe que la ingeniería del proyecto se realizara según cada objetivo específico planteado en el proyecto.

CAPÍTULO V: RESULTADOS. Aquí se analiza y se interpreta los datos obtenidos, con respecto a cada objetivo específico.

CAPÍTULO VI: CONCLUSIONES. Aquí se muestran las conclusiones a las cuales se arribaron en base a los resultados obtenidos.

CAPÍTULO VII: RECOMENDACIONES. En esta parte se recomienda tener en cuenta algunos aspectos técnicos para extender los objetivos del presente trabajo.

CAPÍTULO I: PLANTEAMIENTO DEL PROBLEMA

1.1 Realidad problemática

Ya han pasado muchos años desde que por primera vez en 1993 se mostró al mundo el primer diseño de página web; a medida del paso de los años el diseño y codificación de estas páginas han ido evolucionado considerablemente y creando una gran variedad de sitios web administrables.

Hoy en día tenemos las páginas Responsive Web Desing (Páginas web adaptables), que se han convertido en una tendencia por su capacidad de adaptarse a cualquier dispositivo móvil, ya sean Smartphone, Tablet, Lap Top o cualquier otro dispositivo que tenga cualquier tipo de navegador web y posea acceso a internet.

El enfoque de desarrollo de sitios web basados en técnicas de inteligencia colectiva también se ha convertido en una tendencia que muchas redes sociales utilizan para agregar, consultar y recomendar información a los usuarios sobre sus posibles preferencias, y de esa forma crear relaciones residuales con los clientes o usuarios.

Existen una gran cantidad y variedad de sitios webs dedicados al comercio, comunidad virtual, sitios educativos, redes sociales, y otros. Existen también sitios webs dedicados a promover el turismo y la diversidad cultural de cada zona, sin embargo el 60% de estos no siempre son actualizados y un 10% se encuentran en estado de abandono por sus administradores, un ejemplo es el sitio web de DIRCETUR (Dirección Regional de Comercio Exterior y Turismo) de la provincia de Apurímac, en donde se registra que su última actualización se efectuó en el año 2008.

En la actualidad la mayoría de sitios web enfocados en el turismo, no permiten el intercambio de información entre los usuarios, haciendo que éste a su vez crea que su participación en el crecimiento del sitio no es de suma importancia, sin embargo sin desmerecer el esfuerzo

que realizan los sitios web tradicionales como es el brindarles información importante sobre lugares, hoteles, restaurantes, y otros servicios, hoy en día no es razón suficiente para que el usuario visite con mayor frecuencia el sitio web turístico.

1.2 Problema central

¿Es posible que una aplicación Responsive Web Design para planificación y recomendación turística aplicando inteligencia colectiva permita el intercambio de información entre personas?

1.3 Objetivo General

Desarrollar una aplicación Responsive Web Design para planificación y recomendación turística aplicando Inteligencia Colectiva.

1.3.1 Objetivos específicos

1. Diseñar una base de datos para almacenar la información requerida por la aplicación Responsive Web Design.
2. Implementar inteligencia colectiva mediante un procedimiento de filtrado colaborativo para obtener sugerencias de lugares turísticos.
3. Desarrollar el módulo interacción con el turista para compartir y consultar información de diversos atractivos turísticos y planificar visitas a dichos atractivos.
4. Realizar una prueba piloto de la aplicación para determinar su adaptabilidad.

1.4 Justificación

En los países más desarrollados, está demostrado que el ímpetu en la aplicación de nuevas tecnologías y la incorporación de un sitio web, esta correlacionado de manera positiva con el incremento de las ventas, la productividad y el valor de mercado de las empresas.

Está demostrado también que es suficiente 30 segundos para saber si un sitio web es aceptable por el usuario, para que este a su vez

pueda recomendarlo en su entorno, caso contrario no volverá a visitarlo y será muy corta su interacción con el sitio web.

En el presente proyecto se pretende desarrollar un sitio web dedicado al turismo, con un entorno web adaptable a cualquier dispositivo ya sea Smartphone, Tablet, computadoras portatiles, y otros, siempre y cuando estos posean un navegador web y acceso a internet continuo durante el periodo de navegación en el sitio web, aplicando también técnicas de inteligencia colectiva.

El propósito es desarrollar una idea original de desarrollo web, que permita a los usuarios compartir, consultar y sugerir información de la diversidad turística y cultural de su zona, y al mismo tiempo hacerles notar que su participación es importante en el crecimiento de este sitio web.

En principio se realizara una prueba piloto con usuarios de la zona de Andahuaylas, debido a ellos en la presentación del proyecto se mostrara en su mayoría información sobre la diversidad turística y cultural de ésta zona.

1.5 Viabilidad

1.5.1 Viabilidad económica

El mayor gasto del estudio estará concentrado en los gastos para la recolección de información sobre la diversidad turística y cultural de la zona de Andahuaylas en donde se realizará la primera prueba piloto, y también en el pago por servicios a un programador.

1.5.2 Viabilidad Técnica

Para el desarrollo de la aplicación “Responsive Web Design para planificación y recomendación turística aplicando inteligencia colectiva” se dispone de toda la tecnología, los

medios y la infraestructura necesaria para llevar a cabo dicho desarrollo.

1.5.3 Viabilidad de recursos humanos

Para el desarrollo de la aplicación “Responsive Web Design para planificación y recomendación turística aplicando inteligencia colectiva” y debido a que se utilizara como metodología de desarrollo de software la metodología ágil XP (programación extrema), se dispone de los servicios de un programador para la programación en parejas.

1.6 Limitación del estudio

- Servicio de Internet: En la elaboración de aplicaciones web es fundamental tener acceso continuo a internet para probar la funcionalidad del software.
- Desastres naturales: que provoquen la ausencia de fluido eléctrico en la Zona de Andahuaylas, y en la recolección de información nueva para la alimentación de nuestra base de datos.
- Escasa información disponible sobre sitios turísticos en la provincia de Andahuaylas.
- Posicionamiento y adopción de la tecnología propuesta.

CAPÍTULO II: ESTADO DEL ARTE

(Goge Rito, 2012), en su informe de tesis titulado “Elaboración de un Sitio Web para el Programa Nacional de Asistencia al Turista en el Instituto Guatemalteco de Turismo”, para obtener el grado de Ingeniero de Ciencias y Sistemas, en la Universidad de San Carlos de Guatemala, concluye que internet es uno de los principales medios de comunicación por medio del cual los turistas buscan información para sus viajes.

El uso de un gestor de contenidos en un sitio web permitió a los administradores y a los usuarios interactuar fácilmente debido a su naturaleza y a la configuración de sus módulos.

Esta información es muy importante que ayuda a la presente investigación con respecto a que tecnología utilizar para el desarrollo e implementación del sitio web y como es que los turistas buscan en ellas información antes de programar su viaje.

(Guiza Ezkauriatza, 2011) , su en su informe de tesis titulado “Trabajo Colaborativo en la Web” para obtener el grado de Doctora en Tecnología Educativa en la Universidad de las Islas Baleares, concluye que el trabajo colaborativo no es la panacea, pero si un proceso ad hoc para ser utilizado con las nuevas herramientas de la web 2.0 así como las nuevas generaciones de herramientas que la evolución tecnológica constantemente nos oferta.

Internet y la web son y seguirán siendo, con sus futuras y novedosas tecnológicas, la puerta a un aprendizaje en constante actualización. Permitiendo así nuevos tipos de comunidades de construcción del conocimiento, en donde tanto niños, jóvenes como adultos alrededor del mundo puedan colaborar en proyectos y aprendan unos de otros.

La información vertida por este informe corrobora a la información que se tiene sobre la inteligencia colectiva en la cual está basado este trabajo, que es el que no todos sabemos sobre algo, y es necesario trabajar en equipo para que estos puedan compartir información entre los mismos,

también que el internet a través de sitios web o cualquier otra herramienta permite que se compartan información sin necesidad de estar en un mismo lugar.

(Formoso Lopéz, 2013), en su informe titulado “Técnicas Eficientes para la Recomendación de Productos Basadas en Filtrado Colaborativo” para obtener el grado de Doctor en Tecnologías de Información y Comunicación en la Universidad de Coruña, concluye que es posible contar con técnicas de recomendación precisas y eficaces sin necesidad de recurrir a complejos modelos con multitud de factores y parámetros a estimar. Las ventajas de utilizar algoritmos sencillos son numerosas e incluyen: una mejor comprensión del funcionamiento del algoritmo y sus características, una mayor eficiencia, una mayor eficiencia, una implementación más sencilla y menor dependencia de contar con suficientes datos para entrenar y adaptar el sistema al dominio en que va a ser usado.

También mencionan que las ventajas que ofrecen este tipo de técnicas tienen una mayor importancia que el conseguir mejorar la precisión unas pocas décimas a cambio de complicar significativamente el diseño del algoritmo, que sin embargo ha sido el objetivo perseguido por gran parte de los trabajos que podemos encontrar en la literatura.

En esta investigación se ha demostrado que es posible desarrollar algoritmos sencillos que no sólo son capaces de competir con técnicas mucho más complejas en aspectos tales como la precisión de las recomendaciones, sino que incluso ofrecen resultados superiores cuando se tiene en cuenta el funcionamiento global del algoritmo en multitud de condiciones, su evolución con la densidad de información.

Según (Betarte, Machado, & Molina, 2006), en su informe titulado “PGMúsica Sistema de Recomendación de Música” para obtener el grado de Técnico en computación en la Universidad de la Republica Uruguay, El trabajo realizado en este proyecto se enfocó en el estudio de los sistemas de recomendación analizando particularmente la técnica de

filtrado colaborativo. En ese sentido, se analizaron los algoritmos más usados y con los que se han obtenido mejores resultados, así como las métricas más utilizadas para medir su efectividad. Se evaluaron las problemáticas asociadas a este tipo de sistemas y algunas soluciones que han sido planteadas.

El algoritmo de filtrado colaborativo implementado en este proyecto está basado en memoria; para el cálculo de similitud entre usuarios se utilizó el coeficiente de correlación de Pearson. Si bien este algoritmo produce buenos resultados cuando es aplicado sobre grandes cantidades de información, tiene en la performance su principal limitación. El parámetro más importante a definir del algoritmo implementado es la cantidad máxima de vecinos que se considera en el cálculo de predicción de un ítem. Por esta razón, se realizaron pruebas que permitieron comprobar la existencia de un patrón de comportamiento del error medio absoluto en función de la cantidad máxima de vecinos, lo que permitió definir un valor adecuado para el parámetro. Por otra parte, debido a la dificultad en la recolección de un conjunto de datos considerable, no se pudieron realizar pruebas sobre el cálculo de puntajes a partir de información implícita, el cual afecta la calidad de las recomendaciones.

CAPÍTULO III: MARCO TEÓRICO

3.1 Sitio web

Según (Hobbs, 1999), un sitio web es un conjunto de páginas web (Documentos electrónicos, considerado con la unidad básica de WWW), relacionadas entre sí. Se entiende como página web tanto al fichero que contiene el código HTML (lenguaje de marcas de hipertexto) como todos los recursos que comparten en la página (imágenes, sonidos, código, JavaScript, entre otros).

Por otro lado (Goge Rito, 2012), menciona que toda la información que se encuentra en un sitio Web, puede ser actualizada constantemente, a cada momento, esto permite estar informado al último minuto sobre cada acontecimiento de la página web. Dependiendo de la temática del sitio web, la información puede ser estática (no cambia la información) o bien puede ser interactivas (Puede ser actualizada mensualmente, semanalmente, diariamente, o bien a cada minuto)

3.1.1 Clasificación de sitios web.

Existe una gran variedad de aspectos por los cuales se pueden organizar los sitios web, lo más comunes son los que se presentan a continuación.

Tabla 01: Clasificación de sitios web según algunos aspectos.

DINAMISMO	AUDIENCIA	APERTURA	OBJETIVO	
Interactivos	Públicos	Estructura abierta	Comerciales	Comercio Electrónico
Estáticos	Extranet	Estructura cerrada	Buscadores	Wiki
	Intranet	Estructura Semicerrada	Comunidad Virtual	Educativos
			Sitios Webblog	Portales Web
			Personales	Informáticos

Fuente: (Goge Rito, 2012)

3.2 Diseño web

Según (Mariño Campos, 2005), Se alimenta constantemente de fuentes con el diseño gráfico y las artes visuales, la programación de aplicaciones informáticas, el diseño de interfaces, la redacción de contenidos, la animación tradicional y otras muchas, en general consiste en la actividad de planificar, diseñar e implementar un sitio web.

3.2.1 Diseño web adaptable (Responsive Web Design).

Según (Ramos Martín & Ramos Martín, 2014), es la capacidad de nuestra web de adaptarse a todos los dispositivos que la accedan (Móviles, tabletas, ordenadores, y otros) y visualizarse de manera óptima en cualquiera de ellos, esto va a hacer que nuestro sitio sea más usable para todo el mundo que lo visite y tenga más éxito.

Según (Labrada Martinez & Salgado Ceballos, 2013), las posibilidades y beneficios de este método han sido aceptadas y adoptadas por una gran mayoría de desarrolladores de páginas. Trabajar con proporciones en lugar de pixeles, en el posicionamiento de los componentes del sitio, marca un cambio sustantivo para su despliegue en áreas cambiantes o pantallas diversas.

Gráfico 01: Responsive Web Desing.

Fuente: (Labrada Martinez & Salgado Ceballos, 2013)

3.3 Metodología de desarrollo de software

Según (Leyva Cortés, Prieto Tinoco, Samplo de la Torre, & Garzón Villar, 2006), una metodología de desarrollo es una recopilación de técnicas y procedimientos estructurados en fases para la producción de productos software de manera eficaz y englobando todo el ciclo de vida del mismo.

- Indica que hacer, como, cuando y quien debe hacerlo.
- Determina las etapas y controles a aplicar.

3.3.1 Metodologías de desarrollo Ágil

Según (Schenone, 2004), Las metodologías agiles son un enfoque revolucionario que consiste en un desarrollo altamente productivo, en el que participan grupos pequeños de programadores utilizando herramientas que generaban código en forma automática tomando como entradas sintaxis de alto nivel.

Las metodologías agiles que destacan son las siguientes:

- XP- Extreme Programming
- Scrum
- Crystal Clear
- DSDM
- FDD
- ASD
- XBreed

3.3.2 Principios de los métodos agiles

Según (Sommerville & Alfonso Galipienso, 2005), los principios de la metodología ágil son las siguientes que a continuación se detallan:

1. El desarrollo incremental se lleva a cabo a través de entregas del sistema pequeñas y frecuentes y por medio de un enfoque para la descripción de requerimientos basado en las historias de cliente o escenarios que pueden ser la base para el proceso de planificación.
2. La participación del cliente se lleva a cabo a través del compromiso a tiempo completo del cliente en el equipo de desarrollo. Los representantes de los clientes participan en el desarrollo y son los responsables de definir las pruebas de aceptación del sistema.
3. El interés en las personas, en vez de en los procesos, se lleva a cabo a través de la programación en parejas, la propiedad colectiva del código del sistema, y un proceso de desarrollo sostenible que no implique excesivas jornadas de trabajo.
4. El cambio se lleva a cabo a través de las entregas regulares del sistema, un desarrollo previamente probado y la integración continua.
5. El mantenimiento de la simplicidad se lleva a cabo a través de la refactorización constante para mejorar la calidad de código y la utilización de diseños sencillos que no prevén cambios futuros en el sistema.

3.3.3 XP- Programación Extrema

Según (Pressman), es el más destacado de todos los procesos ágiles de desarrollo de software. Al igual que éstos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad.

Se considera que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos.

Los valores originales de la programación extrema son:

- Simplicidad.
- Comunicación.
- Retroalimentación.
- coraje.

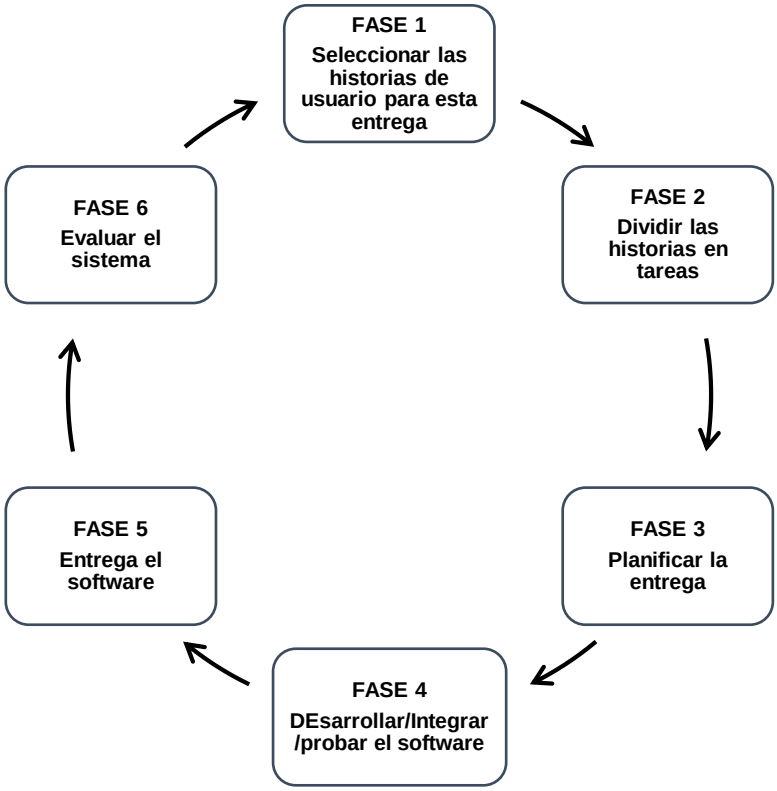
a) Características fundamentales:

- Desarrollo iterativo e incremental.
- Pruebas unitarias continuas.
- Programación en parejas.
- Integración del equipo con el cliente o usuario.
- Corrección de todos los errores.
- Refactorización del código.
- Propiedad de código compartida.
- Simplicidad.

b) Ciclo de entrega en la programación extrema

- El cliente define los requerimientos del sistema (historias de usuario).
- Se plantea una metáfora del sistema, para dividir las historias en tareas.
- El programador estima el esfuerzo necesario para su implementación y entrega.
- El programador construye sistema.
- El cliente evalúa el sistema
- Se vuelve a comenzar si es necesario.

Gráfico 02: Ciclo de entrega en la programación extrema



Fuente: (Sommerville & Alfonso Galipienso, 2005)

3.4 Base de datos

Según (Campos Paré, 2002), indica que, una base de datos es un conjunto estructurado de datos que representa entidades y sus interrelaciones. La representación será única e integrada, a pesar de que debe permitir utilizaciones diversas y simultaneas.

Por otro lado (Aguilar Domínguez, 2009), en su informe detalla que una base de datos es un sistema de registro basado en la computadora, o sea, un sistema cuyo propósito principal es guardar y mantener información. En otras palabras es un repositorio para almacenar datos, el cual generalmente se encuentra integrado y compartido.

Se hace referencia a integrado, ya que aunque los datos pueden encontrarse separados, físicamente en diferentes lugares de almacenamiento, lógicamente son vistos como una unidad, buscando con ello eliminar la redundancia. Por compartido se entiende que

varios usuarios pueden tener acceso a los mismos datos y usarlos de la misma o de diferente manera al mismo tiempo.

3.4.1 Arquitectura de una base de datos

Para (Aguilar Domínguez, 2009), la arquitectura de una Base de Datos se encuentra dividida en tres niveles: el nivel interno, el nivel conceptual y el nivel externo. El nivel interno se encuentra cercano al almacenamiento físico, es decir se refiere a como se encuentra actualmente almacenada la información. El nivel externo se encuentra cercano a los usuarios, se refiere a la manera en que los datos son vistos por los usuarios individuales. El nivel conceptual es el nivel de interconexión entre los otros dos.

La Base de Datos es un componente lógico, es decir no es tangible, sin embargo, los dispositivos en los cuales los datos se almacenan, si son tangibles y se conoce como hardware. El hardware esta puede estar compuesto en unidades de almacenamiento, como discos o dispositivos externos en los cuales reside la Base de Datos.

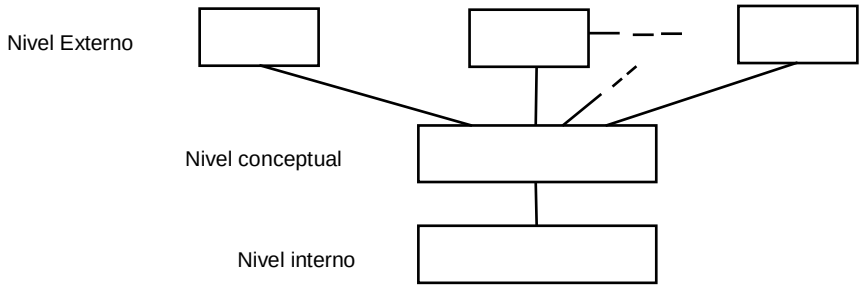
Entre la Base de Datos y los usuarios finales del sistema, debe existir una aplicación que sirva de enlace entre ambos, es aquí donde entra el software, el cual tiene la finalidad de facilitar la interacción entre ambos. Generalmente este software es conocido como SGBD (Sistema de Gestión de Bases de Datos), el cual funciona como regulador entre la Base de Datos y los usuarios.

Existen diferentes tipos de usuarios de una Base de Datos, sin embargo, se pueden agrupar en 3 grupos:

- El programador de la aplicación. Es aquel usuario que se dedica a establecer comunicación entre el usuario final y la Base de Datos.

- El usuario final. Es aquel que se dedica a acceder a la Base de Datos desde una computadora.
- El administrador de la Base de Datos. Es aquel que se dedica a dar mantenimiento a la Base de Datos.

Gráfico 03: Arquitectura de base de datos MCV



Fuente: (Aguilar Domínguez, 2009)

3.4.2 Componentes de una base de datos

Según (Nevado Cabello), los componentes de una base de datos son:

- **Los datos:** El componente fundamental de una base de datos son los que se están interrelacionando entres sí, formando un conjunto con un mínimo de redundancia.
- **El software:** Los datos, para que puedan ser utilizados por diferentes usuarios y diferentes aplicaciones, deben estar estructurados y almacenados de forma independiente de las aplicaciones. Para ello se utiliza un software o conjunto de programas que actúa de interfaz entre los datos y las aplicaciones. A este software se le denomina Sistema de Gestión de Base de Datos (SGBD).
- **Recursos humanos:** Son aquellas personas entre informáticos (Preparan la base de datos) y usuarios

(usan la base de datos) que interactúan con la base de datos.

3.5 Diseño de base de datos

En el informe presentado (Aguilar Domínguez, 2009), describe que el diseño de una Base de Datos es un factor importante dentro del proceso de elaboración de la misma, ya que la principal razón por la que se debe ocuparse es para que exista coherencia, integridad y exactitud de los datos. Si el diseño de una Base de Datos es incorrecto, será difícil acceder a determinados datos y se corre el riesgo de que las búsquedas produzcan resultados inexactos, por que el usuario no sabe si los datos que está recibiendo son correctos.

3.6 Base de datos relacionales

Según (León Osorio, 2008), Una base de datos relacional es una colección de elementos de datos organizados en un conjunto de tablas formalmente descritas desde la que se puede acceder a los datos o volver a montarlos de muchas maneras diferentes sin tener que reorganizar las tablas de la base.

3.7 Modelo Entidad- Relación

(León Osorio, 2008), También menciona que el modelo entidad relación (e-r) es la percepción de un mundo real que consiste en un conjunto de objetos básicos llamados entidades y de unas relaciones entre estos objetos. Se le utiliza para esquematizar la estructura lógica general de lo que será la base de datos. Es bastante empleado en la documentación correspondiente al requerimiento de una base de datos.

- Entidades: Una entidad es un objeto que existe y puede distinguirse de otros objetos.
  Las entidades, pueden ser de dos tipos: Concretas, cuando representan algo tangible (Personas, equipos, libros) o

Abstractas, cuando se utilizan para cosas intangibles (Conceptos de pagos, asignaturas, oficios).

Un Conjunto de entidades es una agrupación de entidades del mismo tipo.

Una entidad está representada mediante un conjunto de atributos; para cada atributo existe un rango de valores permitidos conocidos como dominio de atributo.

- Atributo: Un atributo es una función que mapea un conjunto de entidades dentro de un dominio, para lo cual cada entidad se describe por medio de un conjunto de parejas (atributo, valor del dato).

  Existen también los atributos denominados derivados cuyo valor se puede obtener a partir de otros atributos.

- Relación: Una relación es el vínculo que permite definir una dependencia entre los conjuntos de dos o más entidades. Esto es la relación entre la información contenida en los registros de varias tablas. Las relaciones son definidas de forma natural en un diagrama relacional para expresar un modelo cognitivo que dará lugar posteriormente a las interrelaciones de las entidades.

Una base de datos relacional es por consiguiente, una agrupación de conjunto de entidades, cada uno de los cuales contiene cualquier número de entidades del mismo tipo.

3.4.3 Modelo Conceptual

El primer paso para diseñar una Base de Datos es la elaboración del Modelo conceptual, el cual consiste en abstraer la información del mundo real con la que se cuenta y los procesos que se desean automatizar para disminuir la carga de trabajo en las personas, así como facilitar el manejo de los datos.

Por lo anterior, la recolección de requisitos que se deben cubrir con la implementación de la Base de Datos es importante, ya que indica que tipo de información es la que se va a procesar, así como las necesidades de los futuros usuarios ya que son ellos quienes tendrán interacción con la Base de Datos. Posteriormente es deseable organizar los requisitos en grupos para facilitar el proceso, es decir, agruparla de acuerdo a sus características o bien, de acuerdo al manejo que se va a dar.

Toda la información recabada, así como los requisitos a cubrir deben interpretarse en un diseño que los satisfaga. Para ello hay que identificar los conceptos más relevantes, así como tener en cuenta que se pueden omitir algunos detalles, para facilidad en el manejo de los datos y simplicidad en el diseño, sin afectar el resultado.

3.4.4 Modelo Lógico

El diseño lógico describe el tamaño, la forma y los sistemas necesarios para lo que será la Base de Datos con base en las necesidades de información y operación de los requerimientos.

La información con la que se cuenta del paso anterior, debe descomponerse en datos más sencillos, ya que de esta manera la información es más fácil de almacenar, buscar y manipular cuando es requerida. Este paso es iterativo, ya que debe realizarse hasta que se haya hecho con toda la especificación de lo que se requiere. En caso de que algún requerimiento sea demasiado grande, puede elaborarse un esquema propio e incorporarlo posteriormente al esquema general que representa a toda la especificación.

Una vez realizado lo anterior, se puede realizar el diseño lógico de la Base de Datos. Esto se hace mediante la

implementación de un Diagrama Entidad-Relación, el cual permite ver cuáles son las entidades (tablas), sus atributos (campos o columnas), sus identificaciones (llave primaria) y sus asociaciones (a través de llaves foráneas). Este modelo debe normalizarse para eliminar las redundancias y generalizaciones.

3.4.5 Modelo Físico

Es la implementación física de la Base de Datos empleando el software del SGBD. Una vez realizado el diseño de las tablas que formarán parte de la Base de Datos, es necesario realizar una revisión y ver una vez más si se cumple con el proceso de normalización de las mismas. En caso de existir algún error, es necesario realizar la corrección pertinente. Finalmente, cuando se han creado las tablas, establecido las relaciones y los niveles apropiados de integridad de datos, la Base de Datos está completa.

Una vez que se tiene la Base de Datos, se deben implementar las aplicaciones que permitirán interactuar fácilmente con los datos almacenados, asegurándose de que estas aplicaciones proporcionarán información oportuna y precisa.

3.8 Normalización de base de datos.

Según (Nevado Cabello), la normalización es un conjunto que hace referencia a las relaciones. Básicamente, el principio de normalización indica que las tablas de las bases de datos eliminaran las incoherencias y redundancias, y minimizaran la ineficiencia.

Las bases de datos se describen como incoherentes cuando sus datos se introducen de forma incoherente o cuando los datos de una tabla no coinciden con los datos introducidos en otra tabla.

Una base de datos eficaz no permite aislar los datos exactos que desea. Una base de datos que almacene todos sus datos en una tabla

obligara a pasar por innumerables campos simples para recuperar un dato. Por otra parte, una base de datos completamente normalizada almacena cada información en su propia tabla e identifica cada información con su propia clave principal.

La teoría mate temática que hay detrás de la normalización es rigurosa y compleja, aunque las comprobaciones que se pueden aplicarse para determinar si un diseño tiene sentido son sencillas.

Se puede decir que la normalización es una serie de reglas que involucran análisis y trasformación de las estructuras de datos en relaciones que exhiben propiedades únicas de consistencia, mínima redundancia y máxima estabilidad.

3.8.1. Primera forma normal:

Se considera primera forma normal, si:

- Todos los atributos son atómicos. Un atributo es atómico si los elementos del dominio son simples e indivisibles.
- La tabla contiene una clave primaria única.
- La clave primaria no contiene atributos nulos.
- No debe existir variación en el número de columnas.
- Los Campos no clave deben identificarse por la clave (Dependencia Funcional).
- Debe Existir una independencia del orden tanto de las filas como de las columnas, es decir, si los datos cambian de orden no deben cambiar sus significados.

Esta forma normal elimina los valores repetidos dentro de una Base de Datos.

3.8.2. Segunda forma normal:

Una relación R se encuentra en segunda forma normal si y sólo si está en primera forma normal y todos los atributos no clave, dependen funcionalmente de forma completa de la clave primaria.

En otras palabras podríamos decir que la segunda forma normal está basada en el concepto de dependencia completamente funcional. Una dependencia funcional $x \rightarrow y$ es completamente funcional si al eliminar los atributos A de X significa que la dependencia no es mantenida, esto es que $A \in X, X - \{A\} \nrightarrow Y$. Una dependencia funcional $x \rightarrow y$ es una dependencia parcial si hay algunos atributos $A \in X$ que pueden ser eliminados de X y la dependencia todavía se mantiene, esto es $A \in X, X - \{A\} \rightarrow Y$.

3.8.3. Tercera Forma Normal:

Se dice que una relación está en tercera forma normal si y sólo si está en segunda forma normal y todos los atributos no clave dependen de manera no transitiva de la clave primaria.

Un ejemplo de este concepto sería que, una dependencia funcional $X \rightarrow Y$ en un esquema de relación R es una dependencia transitiva si hay un conjunto de atributos Z que no es un subconjunto de alguna clave de R, donde se mantiene $X \rightarrow Z$ y $Z \rightarrow Y$.

3.9 Sistema de Gestor de Base de datos (SGBD)

Según (Campos Paré, 2002), el sistema de gestión de base de datos es un software encarga de crear y organizar la base de datos, y además atiende todas las solicitudes de acceso hechas a la base de datos tanto por el usuario como por las aplicaciones.

Por otro lado (Cobo, 2005), menciona que el SGBD es un software que permite a los usuarios procesar, describir, administrar y recuperar los datos almacenados en una base de datos.

3.10 MySQL

Según (Cobo, 2005), es un sistema de administración de base de datos relacionales rápido sólido y flexible. Es ideal para crear base de datos con acceso desde páginas web dinámicas, o para cualquier otra solución profesional que implique almacenar datos, teniendo la posibilidad de realizar múltiples y rápidas consultas.

1. Ventajas de MySQL

MySQL ofrece varias ventajas respecto a otros sistemas gestores de base de datos

- El programa está desarrollado en C y C++. Lo que facilita su integración en otras aplicaciones desarrolladas igualmente en esos lenguajes.
- Puede ser descargado gratuitamente de internet haciendo uso de la licencia GPL. Para aquellos que deseen que sus desarrollos basados en MySQL no sean “código abierto” existe también una licencia comercial.
- MySQL utiliza el lenguaje SQL que es el lenguaje de consulta más usado y estandarizado para acceder a bases de datos relacionales. Soporta la sintaxis estándar del lenguaje SQL para la realización de consultas de manipulación, creación y de selección de datos.
- Es un sistema cliente/servidor, permitiendo trabajar como servidor multiusuario y de subprocesador múltiple, es decir, cada vez que se estable una conexión con el servidor, el programa servidor crea un subproceso para manejar la solicitud del cliente, controlando el acceso simultaneo de un gran número de usuarios a los datos y asegurando el acceso solo a usuarios autorizados.

- MySQL dispone de un sistema sencillo de ayuda en línea, y de un monitor que permite realizar todas las operaciones desde la línea de comandos del sistema, sin necesitar ningún tipo de interface de usuario gráfica. Esto facilita la administración remota del sistema utilizando telnet.
- Es portable, es decir, puede ser llevado a cualquier plataforma informática. MySQL está disponible es más de veinte plataformas diferentes incluyendo las distribuciones más usadas de Linux, sistemas operativos Mac X, UNIX y Microsoft Windows
- Es posible encontrar gran cantidad de software desarrollado sobre MySQL o que soporte MySQL. En concreto, son de destacar diferentes aplicaciones open source para la administración de las bases de datos a través de un servidor web.

3.11 Framework

Es “una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras un Framework se puede considerar como una aplicación genérica incompleta y confiable a la que podemos añadirle las últimas piezas para construir una aplicación correcta”. (Lenguajes y Sistemas Informáticos)

Según (Vásquez, 2012), El núcleo de un Framework es un conjunto de clases o librerías mediante las cuales se pueda desarrollar aplicaciones web con navegación sensible a los efectos de cambio sobre el código de manera rápida, ordenada ya segura.

3.5.1 Zend Framework (ZF)

(Eslava Muñoz, 2013), menciona que, existen numerosos Frameworks de PHP, entre los que se pueden mencionar Zend Framework, Cake, Symfony, Codelgnoter, Akelos, Prado, Zoop, y otros. Cada uno proporciona características

diversas que deberíamos conocer antes de saber cuál, seguidamente se muestra en una tabla comparativa entre algunos de los Frameworks más conocidos y populares:

Tabla 02: Framework más conocidos y populares

PHP Framework	PHP 4	PHP 5	MVC	Multiple	ORM	OB	Template	Caching	Validatio	Ajax	Auth	Modules	EDP
Akelos	X	X	X	X	X	X	X	X	X	X	X	X	-
CakePHP	X	X	X	X	X	X	-	X	X	X	X	X	-
CodeIgniter	X	X	X	X	-	X	X	X	X	-	-	-	-
DIY	-	X	X	-	X	X	X	X	-	X	-	-	-
PHPDevShell	-	X	X	-	X	X	X	X	X	X	X	X	-
Prado	-	X	X	X	X	X	X	X	X	X	X	X	X
QPHP	X	X	X	X	-	X	X	-	X	X	X	X	X
Seaguil	X	X	X	X	X	X	X	X	X	X	X	X	-
Symfony	-	X	X	X	X	X	-	X	X	X	X	X	-
WASP	-	X	X	-	-	X	X	-	X	X	X	X	-
Yii	-	X	X	X	X	X	X	X	X	X	X	X	X
Zend	-	X	X	X	X	X	X	X	X	X	X	X	-
Zoop	X	X	X	X	-	X	X	X	X	X	X	-	-

Fuente: (Eslava Muñoz, 2013)

Según (Eslava Muñoz, 2013), ZF es Framework de código abierto para desarrollar aplicaciones web y servicios web con PHP 5. ZF es una implementación que usa código 100% orientado a objetos. La estructura de los componentes de ZF es algo único, ya que cada componente está constituido con una baja dependencia de otros componentes. Esta arquitectura permite a los desarrolladores utilizar los componentes por separado. ZF ofrece un gran rendimiento y una robusta implementación MVC, una atracción de base de datos fácil de usar, y un componente de formularios que implementa la presentación de formularios HTML, validación y filtrado para que los desarrolladores puedan consolidar todas las operaciones usando de una manera sencilla la interfaz orientada a objetos.

3.5.2 Bootstrap

Según (Bootstrap, 2015), indica que Bootstrap es un Framework o conjunto de herramientas de software libre para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como, extensiones de JavaScript opcionales adicionales.

Bootstrap fue desarrollado por Mark Otto y Jacbod Thornton de Twitter, como un marco de trabajo (Framework) para fomentar la consistencia a través de herramientas internas. Antes de Bootstrap, se usaban varias librerías para el desarrollo de interfaces de usuario, las cuales guiaban a inconsistencias y a una carga de trabajo alta en su mantenimiento.

3.12 Turismo

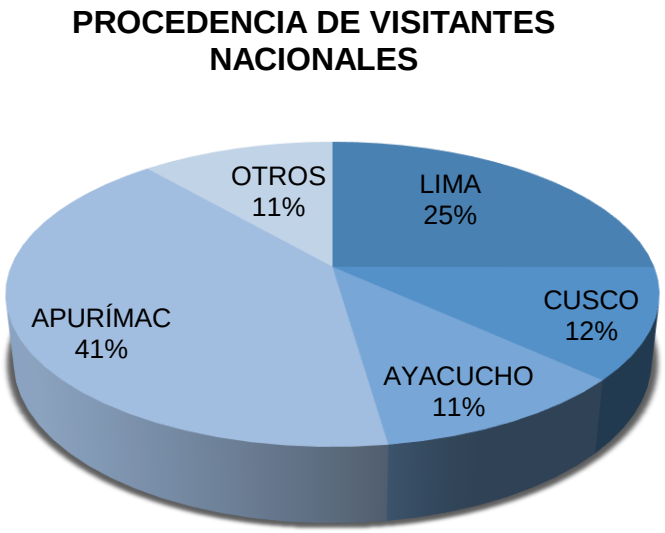
"El turismo es un sistema integrado en el que participan turistas, lugares, territorios sociales". (Knafou & Stock, 2003)

Según (Such Climent, 2008), define al turismo como actividad orientada a la gestión de todos los recursos de manera que se satisfacen las necesidades económicas, sociales y estéticas, respetando al mismo tiempo la integridad cultural, los procesos ecológicos esenciales, la diversidad biológica y los mecanismos de apoyo a la vida.

En la ciudad de Andahuaylas se muestran las siguientes estadísticas correspondientes a la afluencia de Turistas y la información que manejan sobre la Provincia de Andahuaylas.

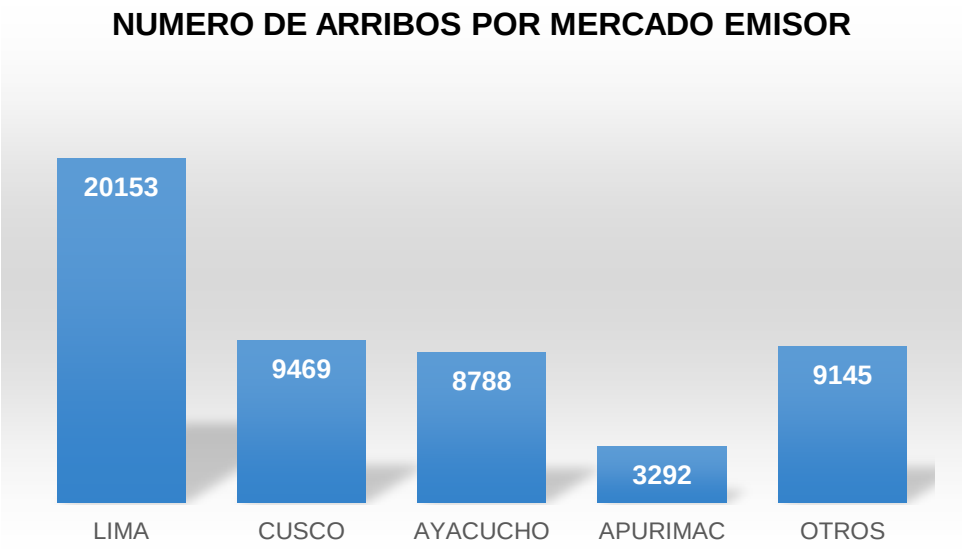
- Procedencia de Visitantes Anual

Gráfico 04: Procedencia de visitantes nacionales



Fuente: Encuesta Mensual a establecimientos de hospedaje 2014– MINCETUR

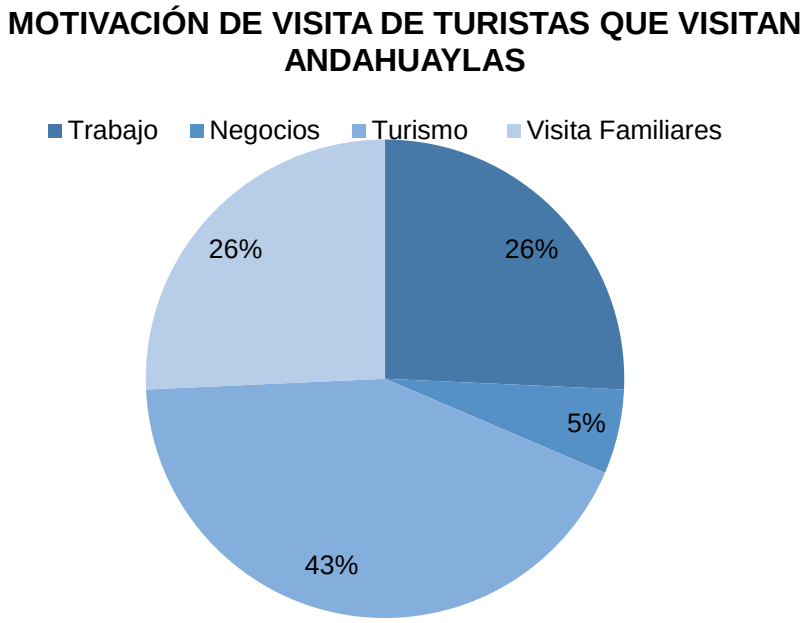
Gráfico 05: Número de Arribos por Mercado Emisor



Fuente: Encuesta Mensual a establecimientos de hospedaje 2014 – MINCETUR.

- Motivación de visitas

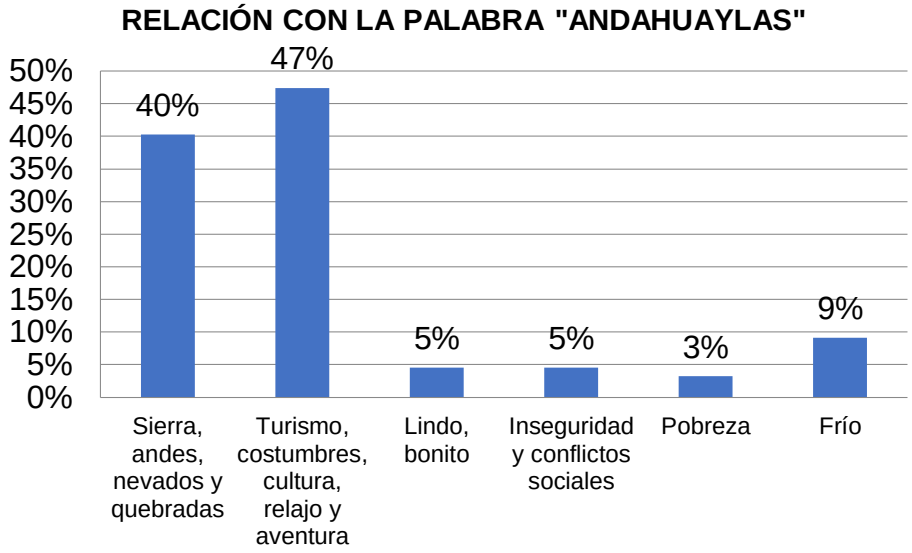
Gráfico 06: Motivación de visita de turistas que visitan Andahuaylas



Fuente: Plan de Desarrollo de Producto Turístico 2014 – Dircetur Andahuaylas

- Relación con la Palabra Andahuaylas y en qué departamento creen que esta.

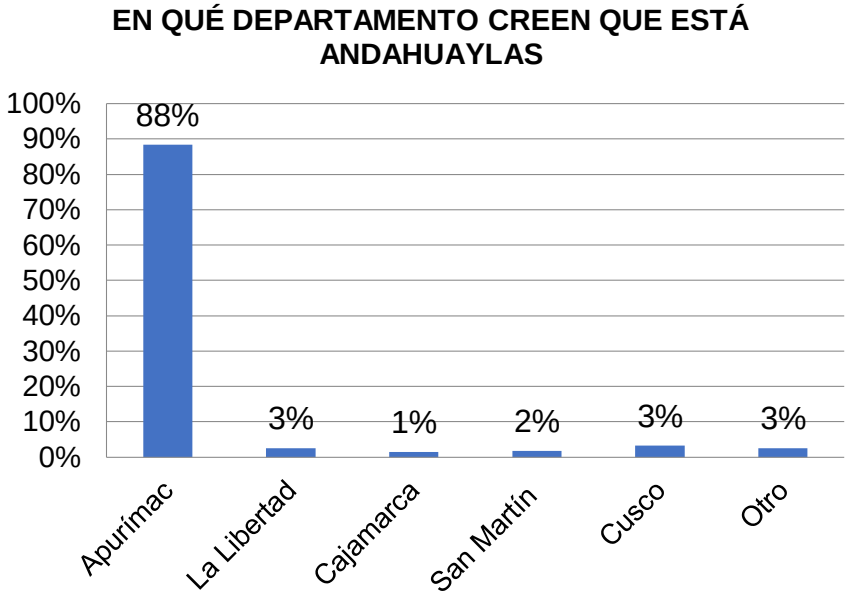
Gráfico 07: Relación con la palabra "Andahuaylas"



Fuente: Plan de Desarrollo de Producto Turístico 2014 – Dircetur Andahuaylas.

- Relación en qué departamento creen que esta Andahuaylas

Gráfico 08: Relación con la palabra "Andahuaylas"



Fuente: Plan de Desarrollo de Producto Turístico 2014 – Dircetur Andahuaylas

3.13 Inteligencia colectiva

(Ísmodes, 2006), menciona que distintos teóricos caracterizan a la inteligencia colectiva como aquello que supera al pensamiento individual o al de un grupo particular y que resulta de facilitar, a un relativamente número de personas.

Según (Pardo Kuklinsk & Cobo Romaní, 2007), resulta fundamental la inclusión y participación de los conocimientos de todos. "La web del futuro expresara la inteligencia colectiva de una humanidad mundializada e interconectada a través del ciberespacio" (Levy, 2003).

3.14 Filtrado colaborativo

Según (Betarte, Machado, & Molina, 2006), el filtrado colaborativo consiste en recomendar elementos que han gustado a usuarios con preferencias similares, basándose únicamente en el puntaje que estos

asignan a los ítems del sistema. Para ello las personas puntúan los elementos de acuerdo a sus intereses (a medida que esto sucede se va generando su perfil) y a partir de ellos se calculan las recomendaciones.

Algunos enfoques se basan en calcular la similitud entre usuarios, y a partir de sus preferencias se obtienen los elementos a recomendar. Para cada usuario se crea un conjunto de "vecinos cercanos": personas cuyas evaluaciones tienen grandes semejanzas a las del usuario que solicita la recomendación. Los resultados para los elementos no calificados por él, se predicen en base a la combinación de puntajes conocidos de los vecinos cercanos. Entonces el procedimiento general aplicado en estos sistemas se resume de la siguiente forma:

- Los usuarios expresan sus valoraciones sobre elementos del sistema, generalmente mediante una escala numérica.
- A partir de esa información, se intenta predecir el puntaje que daría el usuario que solicita la recomendación (usuario activo), a los elementos del sistema no conocidos hasta el momento por él.
- De las predicciones calculadas se seleccionan los elementos con valores más altos para realizar la recomendación. Existen muchas maneras de generar las predicciones mencionadas, desde diferentes enfoques y basando los cálculos en diferentes algoritmos.

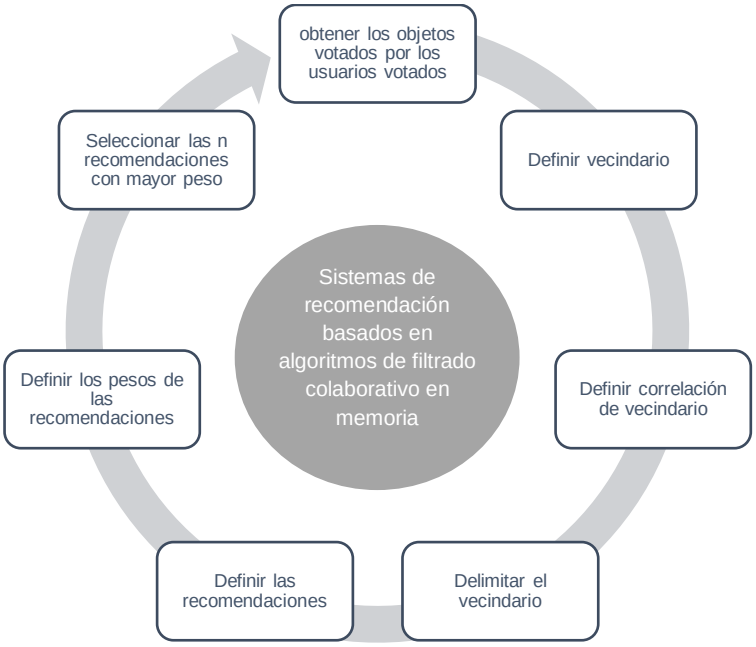
3.8.1 Algoritmos de filtrado colaborativo.

Según (Betarte, Machado, & Molina, 2006), el objetivo de los algoritmos de filtrado colaborativo es predecir el puntaje que un usuario dará a un ítem en particular, basándose en una base de datos que contiene los puntajes que otros usuarios han ingresado para los ítems del sistema, ignorando por completo los atributos o características de los usuarios y

elementos. Se pueden distinguir dos tipos generales de algoritmos de Filtrado Colaborativo:

- **Algoritmos basados en Memoria:** el cálculo de una predicción se realiza considerando la similitud entre el usuario activo y los demás usuarios del sistema. A estos algoritmos también se les denomina “basados en vecindario”.

Gráfico 09: Filtros colaborativos basados en memoria



Fuente: (Betarte, Machado, & Molina, 2006)

- **Algoritmos basados en Modelo:** se utilizan los puntajes brindados por los usuarios para entrenar un modelo probabilístico, el cual se utiliza para generar las predicciones. En este caso no se tiene en cuenta de forma explícita la similitud entre el usuario activo y los demás usuarios del sistema.

“La mayoría de los sistemas de filtrado colaborativo automático utilizan algoritmos basados en memoria para el cálculo de predicción. A su vez, para computar la medida de

similitud entre dos usuarios, se utilizan varios métodos, donde el coeficiente de correlación de Pearson y el vector de similitud son los más utilizados" (Betarte, Machado, & Molina, 2006).

CAPÍTULO IV: INGENIERÍA DEL PROYECTO

La ingeniería del proyecto se realizara de acuerdo a cada objetivo específico planteado en el proyecto.

4.1 Objetivo 1:

Diseñar una base de datos para almacenar la información requerida por la aplicación Responsive Web Design para la planificación y recomendación turística.

- Para el diseño de la base de datos utilizaremos la herramienta Erwin Data Modeler. Ya que nos ofrece una vista centralizada de las definiciones de los datos principales, lo que nos permitirá aprovechar la información como activo estratégico y gestionar de forma más eficaz los recursos de datos para ahorrar tiempo y dinero.
- Utilizaremos las reglas de integridad de datos, para preservar la integridad de los datos que serán almacenados en la BD en la mayor medida posible.
- Como sistema de gestor de base de datos utilizaremos MySQL relacional. ya que nos permitirá que varios usuarios acceder a los datos al mismo tiempo, además brindan facilidades eficientes y un grupo de funciones con el objetivo de garantizar la confidencialidad, la calidad la seguridad y la integridad de los datos que contienen, así como el acceso fácil y eficiente a los mismos.

4.2 Objetivo 2:

Implementar inteligencia colectiva mediante un procedimiento de filtrado colaborativo para obtener sugerencias de atractivos turísticos.

4.2.1 Requerimiento funcional:

Tabla 03: Requerimiento funcional –filtrado colaborativo

REQUERIMIENTO	DESCRIPCIÓN
Calificar con puntuaciones del 1 al 5 los lugares o festividades lo que visito y asistió respectivamente	Todos los usuarios deben poder recomendar atractivos turísticos con puntuaciones, de acuerdo a como fue su experiencia en su visita o su experiencia en dónde 0 = muy malo y 5 = muy bueno.

Fuente: Elaboración Propia

El análisis completo del sistema se encuentra detallado más específicamente en el Anexo N° 02.

4.3 Objetivo 3:

Desarrollar el módulo interacción con el turista para compartir y consultar información de diversos atractivos turísticos y planificar visitas a dichos atractivos.

4.3.1 Requerimiento funcional

Tabla 04: Requerimiento funcional – inteligencia colectiva

REQUERIMIENTO	DESCRIPCIÓN
Acceso al menú de planificación y recomendación turística	Todos los usuarios deben poder ingresar al módulo de recomendación y planificación turística.
Insertar y modificar información turística y cultural en imágenes, video y texto.	Todos los usuarios deben poder acceder al módulo de insertar y modificar la información turística que ellos estimen por conveniente.

Fuente: Elaboración Propia

El análisis completo del sistema se encuentra detallado más específicamente en el Anexo N° 02.

4.4 Objetivo 4:

Realizar una prueba piloto de la aplicación para determinar su adaptabilidad.

- Se adquirirá un hosting y un dominio para tener la aplicación en la web.
- Se comprobara su portabilidad ingresado como mínimo desde 3 dispositivos móviles diferentes.

4.5 Especificaciones adicionales

- La metodología que utilizaremos para el desarrollo de la aplicación, será la Metodología Ágil XP.
- Se hará uso de la herramienta tecnológica Zend Framework para la programación, ya que es una herramienta de código 100% abierto, y que tiene una serie de características como: Conexión a base de datos, programación MVC, soporta Ajax, Y otros, que serán requeridas por la aplicación que se pretende desarrollar
- Se hará uso de la herramienta Bootstrap para el diseño de interfaces, ya que es un Framework que a su vez es compatible con la mayoría de navegadores web, esta información es importante ya que nuestra aplicación está desarrollado bajo un enfoque Responsive Web Desing que pretende desarrollar un entorno adaptativo a todos los dispositivos móviles que a su vez poseen distintos navegadores webs.

CAPÍTULO V: RESULTADOS

A continuación se muestra cada resultado por objetivo específico.

5.1 Resultado 1:

Según el primer objetivo “Diseñar una base de datos para almacenar la información requerida por la aplicación Responsive Web Desing para la planificación y recomendación turística.”.

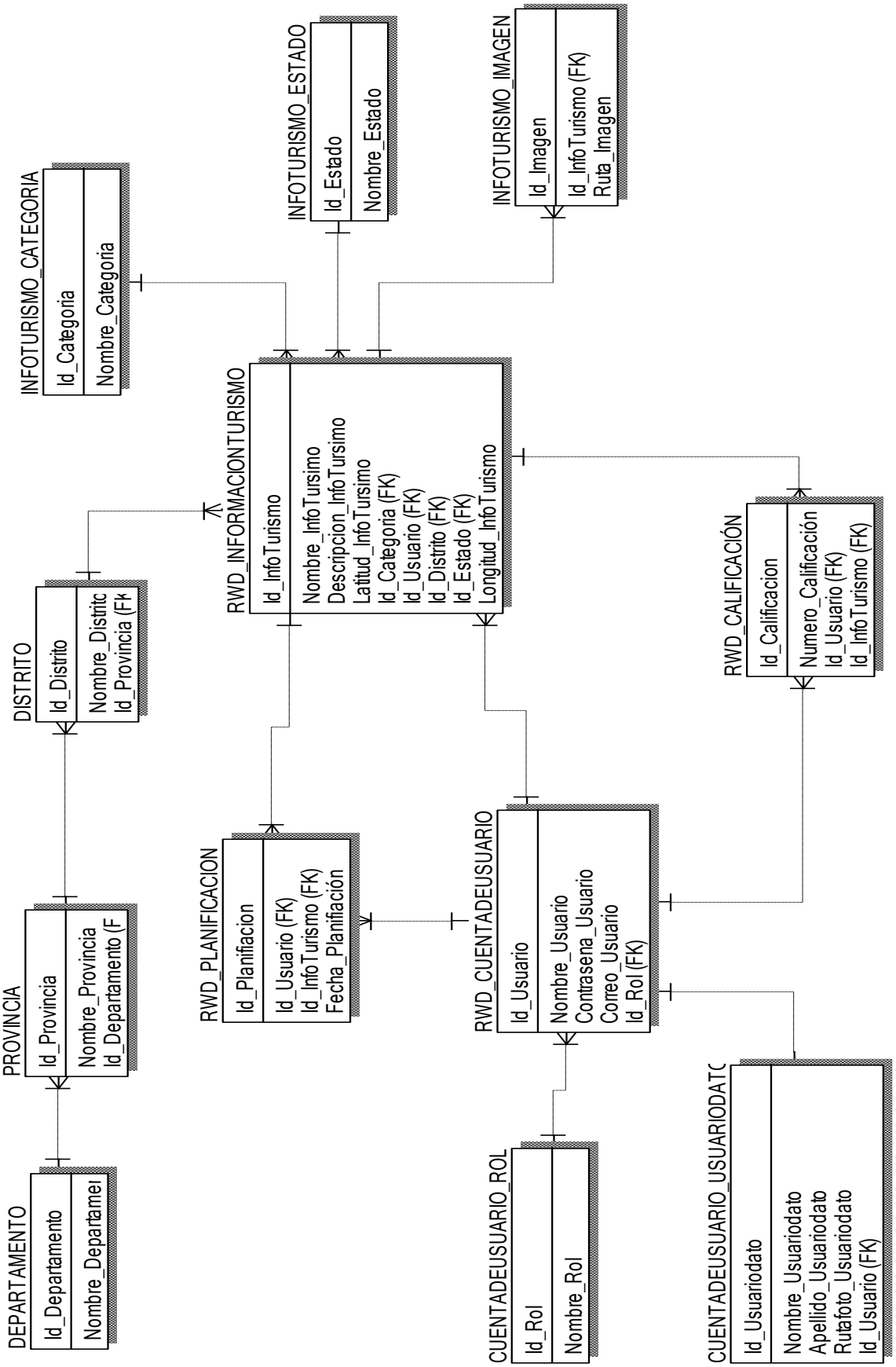
En el grafico N° 10 se muestra el diseño de base de datos requerida para el desarrollo de la aplicación. En siguiente diseño vemos 5 tablas principales que son las siguientes:

- RWD_CUENTADEUSUARIO: En esta tabla se almacena la información de los usuarios que interactúan con el sitio web pero con mayores privilegios según sea su rol asignado por el Administrador del Sitio.
- RWD_INFORMACIONTURISMO: En esta tabla se almacena toda la información concerniente a sitios turísticos.
- RWD_PUNTUACIONES: En esta tabla se almacenan todas las puntuaciones por sitio emitidas por cada usuario.
- RWD_PLANIFIACACIÓN: En esta tabla almacenamos todas las posibles visitas que el usuario quiera realizar a los diversos atractivos turísticos.
- RWD_CALIFICACIÓN: En esta tabla almacenamos todas las calificaciones emitidas por el usuario por cada atractivo turístico.

Existen otras tablas que se muestran en el diseño de la base de datos que también son importantes al igual que las principales y que ayudan a que la base de datos se encuentre normalizada hasta su tercera forma normal.

Cada una de las tablas cumplen una función importante, puesto que se evitada que sean almacenados datos nulos o vacíos.

Gráfico 10: Diseño de base de datos de la aplicación Responsive Web Desing



Fuente: Elaboración Propia

5.2 Resultado 2:

Según el segundo objetivo “Implementar inteligencia colectiva mediante un procedimiento de filtrado colaborativo para obtener sugerencias de lugares turísticos”.

Como se mencionó en la ingeniería del proyecto, para la implementación de este módulo hemos utilizado un algoritmo colaborativo basado en memoria que calcula la distancia entre usuarios o vecinos el cual se denomina coeficiente de correlación de Pearson.

Los motivos por el cual se ha hecho uso de este algoritmo basado en el coeficiente de Correlación Pearson son los siguientes:

- El algoritmo basado en la formula estadística del coeficiente de correlación de Pearson a diferencia que otros algoritmos que están agrupados dentro de los algoritmos basados en memoria, es uno de los más eficiente, ya que tiene la capacidad de no solo calcular la distancia la relación más cercana entre dos usuarios o vecinos de acuerdo a sus votos emitidos, si no el de también compararla con el grado de intensidad de voto para que la predicción sea más precisa. En el siguiente cuadro se explica más detalladamente sobre la intención de voto:

Tabla 05: Ejemplo Ilustrativo

	Sitio 1	Sitio 2	Sitio 3	Sitio 4	Sitio 5
Usuario 1	3	2	2	1	3
Usuario 2	5	4	4	3	5

Fuente: Elaboración Propia

Según el cuadro anterior se podría concluir que el Usuario 1 y el Usuario 2 son totalmente diferentes ya que según sus puntuaciones estos usuarios no coinciden en ningún voto y más allá de considerarlos similares son considerados diferentes, sin embargo, según el cálculo de la semejanza mediante el coeficiente de correlación de Pearson se concluye que estos 2 usuarios son parecidos, por lo mismo que a estos dos usuarios les puede gustar el mismo sitio, pero no lo expresan de la misma manera ni con la misma intensidad, también se concluye que el Usuario 1 es más

mezquino con sus puntuaciones y el Usuario 2 es más generoso con las mismas.

La fórmula del coeficiente de correlación de Pearson es la siguiente:

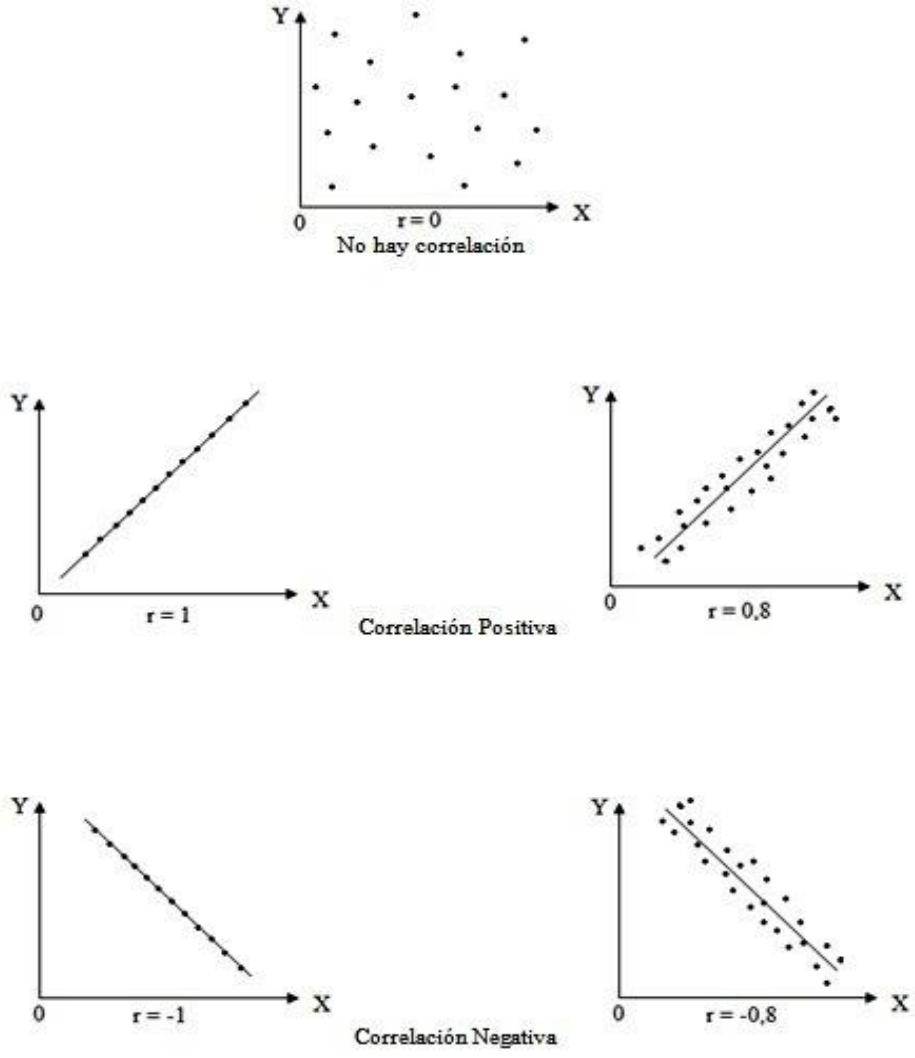
$$r_{xy} = \frac{n \sum xy - \sum x \sum y}{\sqrt{(n \sum x^2 - (\sum x)^2)(n \sum y^2 - (\sum y)^2)}}$$

La limitación de este algoritmo es la siguiente:

- Si el promedio de votaciones es igual a todas las votaciones emitidas por el usuario, el algoritmo no considera al usuario en sus cómputos para determinar vecinos cercanos, debido a que surge un proceso de cálculo indeterminado.

Diagrama de procesos del Algoritmo de correlación de Pearson:

Grafico N° 11: Tipos de Coeficientes de Correlación de Pearson



Fuente: Elaboración Propia

Tabla N° 06: Interpretación de los resultados obtenidos en el cálculo del coeficiente de correlación de Pearson.

Valor	Significado
-1	Correlación negativa grande y perfecta
-0,9 a -0,99	Correlación negativa muy alta
-0,7 a -0,89	Correlación negativa alta
-0,4 a -0,69	Correlación negativa moderada
-0,2 a -0,39	Correlación negativa baja
-0,01 a -0,19	Correlación negativa muy baja
0	Correlación nula
0,01 a 0,19	Correlación positiva muy baja
0,2 a 0,39	Correlación positiva baja
0,4 a 0,69	Correlación positiva moderada
0,7 a 0,89	Correlación positiva alta
0,9 a 0,99	Correlación positiva muy alta
1	Correlación positiva grande y perfecta

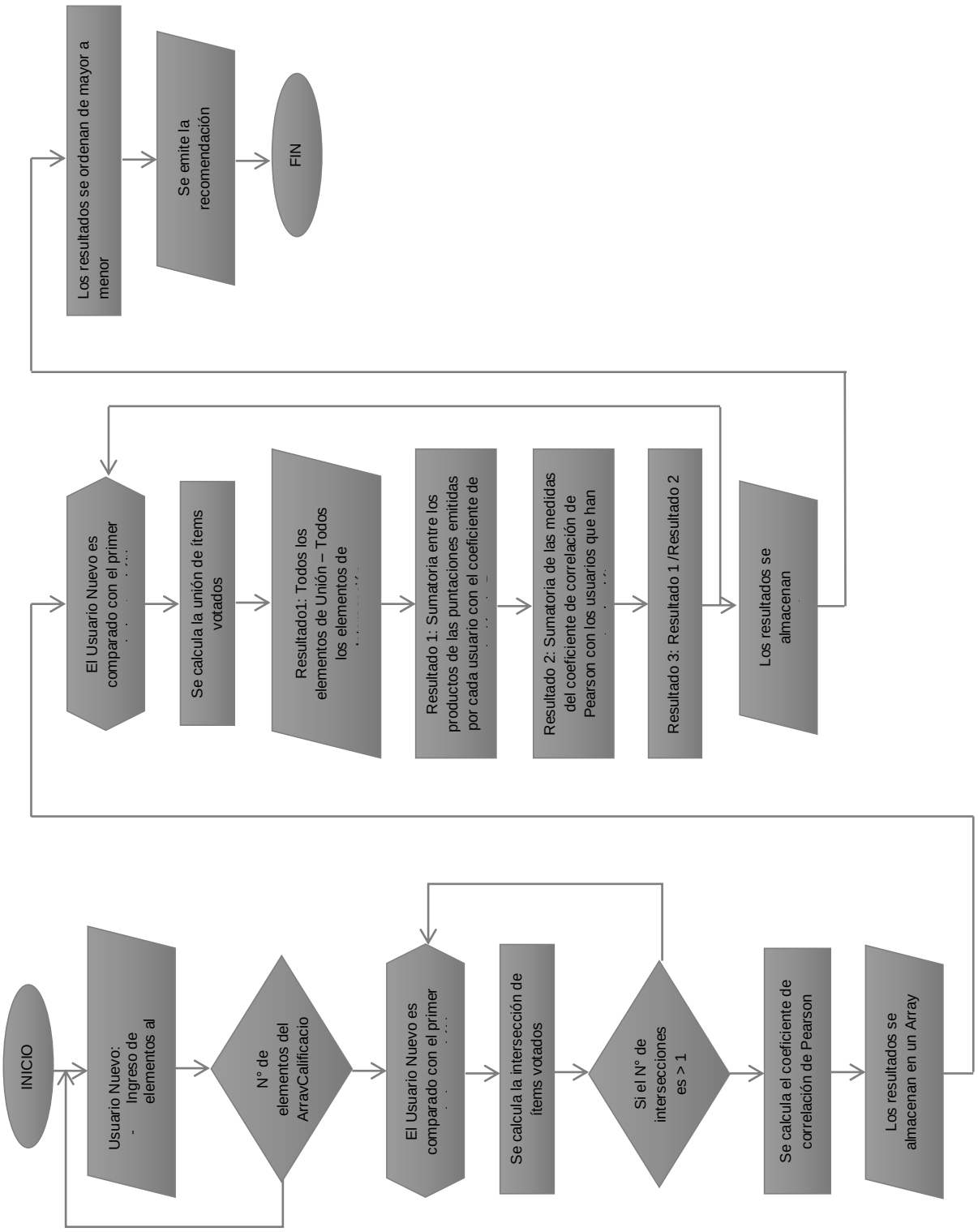
Fuente: Elaboración Propia

Si el resultado es 1, significa que la relación entre los usuarios es muy fuerte

Si el resultado es 0, significa que la relación entre usuarios es nula.

Si el resultado es -1, significa que no hay relación entre los usuarios.

Grafico N° 12: Diagrama de procesos del Algoritmo de correlación de Pearson:



Fuente: Elaboración Propia

En el siguiente conjunto de código, se muestra todos los procedimientos del algoritmo de filtrado colaborativo.

```
 //funcion que crea un lista  array[informacionturistica]=calificacion
    public function _crearlista($CalificacionUser)
    {
        $Array=array();
        $i=0;
        foreach($CalificacionUser as $row2){
              $Array[$row2['cod_turismo']]=$row2['turi_calificacion'];
        }
         return $Array;
    }
//funcion array almacena informacion turistica que califico el usuario
    public function _turismoUser($User)
    {
        $Array=array();
        $i=0;
        foreach($User as $row2){
              $Array[$i]=$row2['cod_turismo'];
              $i++;
        }
         return $Array;
    }
//funcion array devuelve calificaciones de información turistica en
comun de user y user nuevo
    public function _turismoIntersect($Data,$ArrayIntersect)
    {
        $Array=array();
        $i=0;
        foreach ($ArrayIntersect as $Turismo){
            foreach ($Data as $indice1 => $fil1){
                if($Turismo==$indice1){
                    $Array[$i]=$fil1;
                    $i++;
                }
            }
        }
         return $Array;
    }
//funcion array almacena informacion turistica que califico el usuario
    public function _pearson($x,$y)
    {
        $length= count($x);
        $mean1=array_sum($x) / $length;
        $mean2=array_sum($y) / $length;

        $a=0;
        $b=0;
        $axb=0;
        $a2=0;
```

```
        $b2=0;

        for($i=0;$i<$length;$i++)
        {
            $a=$x[$i]-$mean1;
            $b=$y[$i]-$mean2;
            $axb=$axb+($a*$b);
            $a2=$a2+ pow($a,2);
            $b2=$b2+ pow($b,2);
        }
        $denominador=sqrt($a2*$b2);
        //si el denominador es igual a 0
        if($denominador!=0){
            $corr= $axb /$denominador;
        }else{
            $corr=0;
        }
        //si el coeficiente de correlacion es negativo
        if($corr<0){
             return 0;
        }else{
           return $corr;
        }

    }
    public function _algoritmoAction($codUsuario)
    {
          // action body
        //traemos todos los usuarios del sistema menos el usuario en
session
        $AllUser=$this->db_usuario->TodoUsuario($codUsuario);
        //calificacion del usuario nuevo

        $Nuevo=$this->db_usuariocalificacion->CalificacionUsuario
($codUsuario);
//si exiten calificacion del nuevo usuario procedemos con el algoritmo

        if(count($Nuevo)>0){
        $Datos1=$this->_crearlista($Nuevo);
        $TurismoUserNuevo=$this->_turismoUser($Nuevo);


        $NuevoAllUser=array();
        $arrayTurismo=array();
        $i=0;
        foreach($AllUser as $row){
            $CalificacionUser=$this->db_usuariocalificacion-
>CalificacionUsuario ($row['cod_usuario']);
            $Datos=$this->_crearlista($CalificacionUser);
            $turismoUser =$this->_turismoUser($CalificacionUser);
           //intersecando que exista ciudades en comun entre las dos
personas
```

```
            $result = array_intersect($turismoUser,
$TurismoUserNuevo);

            // si existe alguna ciudad en comun
            if(count($result)>1){
                $Valor=0;
                //datos de la persona que entra en bucle
                //indice1 ciudad que voto fil1 el valor
                //datos1 usuario nuevo
                //indice2 ciudad que voto fil2 el valor

                //DISTANCIA EUCLIDEANA
                foreach ($Datos as $indice1 => $fil1){
                    foreach ($Datos1 as $indice2 => $fil2){
                        if($indice1==$indice2){
                            $Valor=$Valor + pow(($fil2-$fil1), 2);
                        }
                    }
                    //arrayturismo guarda todas las ciudades
                    calificadas;
                    $arrayTurismo[$i]=$indice1;
                    $i++;
                }
                $Distancia = (1/(1+sqrt($Valor)));

                //ALGORITMO DE PEARSON
                $InterUser=$this->_turismoIntersect($Datos,$result);
                $InterUserNuevo=$this-
>_turismoIntersect($Datos1,$result);
                 $Distancia=$this-
>_pearson($InterUser,$InterUserNuevo);

                 array_push($NuevoAllUser,
                                    array(
                                        "Usuario"=>
$row['cod_usuario'],
                                        "UsuarioN"=>
$row['usu_nombre'],
                                        "Ciudad" => $Datos,
                                       // "Distanciae" =>
1/(1+sqrt($Valor)),
                                        "Distanciae" => $Distancia,
                                    ));
            }
        }


        //ciudad unica elimina redundancia
        //$CiudadNuevoUser ciudades calificadas por el nuevo usuario
        $CiudadUnica=array_unique($arrayTurismo);
        $CiudadAll=$CiudadUnica;
        foreach($CiudadUnica as $indice => $row){
              foreach($TurismoUserNuevo as $row2){
```

```
                    if($row==$row2){
                     unset ($CiudadUnica[$indice]);
                    }
                }
            }
     //ciudadunica solo contiene ciudades que no voto el nuevo usuario
    //comparando ciudaddes
    //$NuevoAllUser contiene las distancias de cada usuario con
respecto al nuevo usuario
            $Recomendacion=array ();
            Foreach ($CiudadUnica as $row) {
                $suma=0;
                $SumDE=0;
               Foreach ($NuevoAllUser as $row2){
                   foreach($row2['Ciudad'] as $indice => $valor){
                   //usuario [i] califico el sitio $row
                       if($indice == $row){
                           $Multi=$row2['Distanciae']*$valor;
                           $suma=$suma+$Multi;
                           $SumDE=$SumDE + $row2['Distanciae'];
                       }
                   }

               }
               $Recomendacion[$row]=$suma/$SumDE;

            }

            return array($Recomendacion,$CiudadAll,$NuevoAllUser);
//              echo '<pre>';
//              print_r($CiudadUnica);
//              print_r($NuevoAllUser);
//
//              print_r($DistanciaE);
//             print_r($Recomendacion);
//              echo '<pre>';exit;
        }

    }
}
```

A continuación se realizara una comparación de resultados entre el cómputo manual en una hoja de cálculo de Excel y el los resultados obtenidos en el sitio web.

Calculo manual: Se realizar el cálculo para el usuario Rafael en dos situaciones con 2 votaciones y posteriormente con 3 votaciones, a continuación se muestra el cuadro de calificaciones en el primer caso.

Primer caso

Tabla 07: Calificación de los usuarios primera prueba.

N°	Usuario	Sondor Raymi	Laguna de Pacucha	Bosque de Piedras	Baños termales de Hualalachi	Señor de Huanca	Pukllay	Plaza de Armas de Andahuylas
1	Maria	2	3	2	2	3	1	2
2	Reynaldo	4	4		5			
3	Andrea	5		5	4	3	3	4
4	Flora		3	2		2		2
5	Julio	4		3	4	4		
6	Carlos	1	2			1	2	
7	Jose	3	2	3	3			
8	Macarena	2	2	3	4	3	3	4
9	Pilar		2	4	1	2		3
10	Rafael		3		4			

1

2

Fuente: Elaboración Propia.

1 Son calificaciones o votaciones de cada usuario por cada atractivo turístico.

2 Es la calificación o votación emitida por el usuario Rafael.

Paso N° 1: Calculamos la distancia con cada usuario y hallamos el vecino más cercano.

- Veremos que el cálculo se realiza solo con usuarios o vecinos con los cuales el usuario Rafael tenido por lo menos 2 intersecciones utilizando la herramienta Excel.

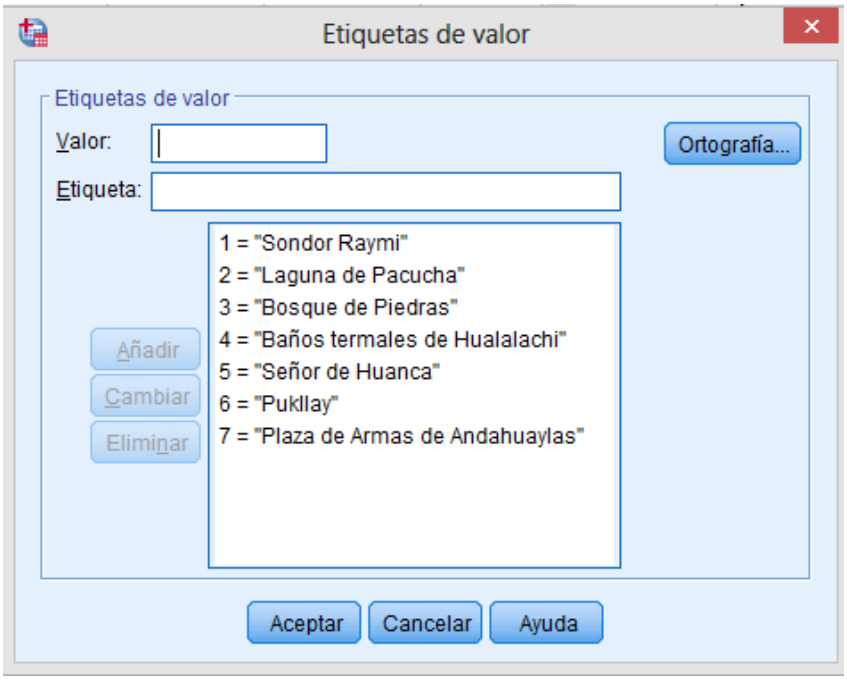
Tabla 08: Calculo de la distancia entre vecinos

	Laguna de Pacucha	Baños termales de Hualalachi	
Maria	3	2	
Rafael	3	4	
		C.C.P	-1.00
	Laguna de Pacucha	**Baños termales de Hualalachi**	
Reynaldo	4	5	
Rafael	3	4	
		C.C.P	1.00
	Laguna de Pacucha	**Baños termales de Hualalachi**	
Jose	2	3	
Rafael	3	4	
		C.C.P	1.00
	Laguna de Pacucha	**Baños termales de Hualalachi**	
Macarena	2	4	
Rafael	3	4	
		C.C.P	1.00
	Laguna de Pacucha	**Baños termales de Hualalachi**	
Pilar	2	1	
Rafael	3	4	
		C.C.P	-1.00

Fuente: Elaboración Propia.

Calculo del coeficiente de correlación de Pearson entre Usuarios utilizando la herramienta SPSS.

Grafico N° 13: Asignación de códigos a los atractivos turísticos en el SPSS.



Fuente: Elaboración Propia.

Grafico N° 14: Cuadro de datos ingresados en la herramienta SPSS.

Atractivos	Maria	Reynaldo	Andrea	Flora	Julio	Carlos	Jose	Macarena	Pilar	Rafael
1	2	4	5	.	4	1	3	2	.	.
2	3	4	3	3	.	2	2	2	2	3
3	2	.	5	2	3	.	3	3	4	.
4	2	5	4	.	4	.	3	4	1	4
5	3	.	3	2	4	1	.	3	2	.
6	1	.	3	.	4	2	.	3	.	.
7	2	.	4	2	.	.	.	4	3	.

Fuente: Elaboración Propia.

Tabla N° 09: Resultado del cálculo de coeficiente de correlación de Pearson entre usuarios. Con la herramienta SPSS.

		María	Reynaldo	Andrea	Flora	Julio	Carlos	Jose	Macarena	Pilar	Rafael
María	Correlación de Pearson	1	-,500	,000	,577	,000	-,302	-1,000**	-,296	-,320	-1,000**
	Sig. (bilateral)		,667	1,000	,423	1,000	,698	,000	,520	,599	.
	N	7	3	6	4	5	4	4	7	5	2
Reynaldo	Correlación de Pearson	-,500	1	-1,000**	.b	.b	.b	,500	1,000**	-1,000**	1,000**
	Sig. (bilateral)	,667		.	.	.	.	,667	,000	.	.
	N	3	3	2	1	2	2	3	3	2	2
Andrea	Correlación de Pearson	,000	-1,000**	1	.b	-,559	-,500	.b	-,297	,632	.b
	Sig. (bilateral)	1,000	.		,000	,327	,667	,000	,568	,368	.
	N	6	2	6	3	5	3	3	6	4	1
Flora	Correlación de Pearson	,577	.b	.b	1	.b	1,000**	-1,000**	-,816	-,522	.b
	Sig. (bilateral)	,423	.	,000		.	.	.	,184	,478	.
	N	4	1	3	4	2	2	2	4	4	1
Julio	Correlación de Pearson	,000	.b	-,559	.b	1	.b	.b	,000	-,945	.b
	Sig. (bilateral)	1,000	.	,327	.		,000	,000	1,000	,212	.
	N	5	2	5	2	5	3	3	5	3	1
Carlos	Correlación de Pearson	-,302	.b	-,500	1,000**	.b	1	-1,000**	,000	.b	.b
	Sig. (bilateral)	,698	.	,667	.	,000		.	1,000	.	.
	N	4	2	3	2	3	4	2	4	2	1
José	Correlación de Pearson	-1,000**	,500	.b	-1,000**	.b	-1,000**	1	,522	,189	1,000**
	Sig. (bilateral)	,000	,667	,000	.	,000	.		,478	,879	.
	N	4	3	3	2	3	2	4	4	3	2

Macarena	Correlación de Pearson	-,296	1,000**	-,297	-,816	,000	,000	,522	1	-,105	1,000**
	Sig. (bilateral)	,520	,000	,568	,184	1,000	1,000	,478		,867	.
	N	7	3	6	4	5	4	4	7	5	2
Pilar	Correlación de Pearson	-,320	-1,000**	,632	-,522	-,945	.b	,189	-,105	1	-1,000**
	Sig. (bilateral)	,599	.	,368	,478	,212	.	,879	,867		.
	N	5	2	4	4	3	2	3	5	5	2
Rafael	Correlación de Pearson	-1,000**	1,000**	.b	.b	.b	.b	1,000**	1,000**	-1,000**	1
	Sig. (bilateral)	.	.	.	.	.	.	.	.	.	
	N	2	2	1	1	1	1	2	2	2	2

Fuente: Elaboración Propia

De los resultados sombreados obtenidos en el Cuadro con respecto a la correlación de Pearson se concluye lo siguiente:

- Los Usuarios Rafael y María no se correlacionan o no tienen ninguna relación.
- Los Usuarios Rafael y Reynaldo se correlacionan o tienen una relación fuerte.
- Los Usuarios Rafael y José se correlacionan o tienen una relación fuerte.
- Los Usuarios Rafael y Pilar no se correlacionan o no tienen ninguna relación.

Paso N° 2: Calculo de la recomendación sitios al nuevo usuario

- Seguidamente se realiza el cálculo para saber que sitios serán recomendados al usuario para lo cual se coloca todos los sitios q aún no han sido visitados por el usuario y sus puntuaciones respectivas de los usuarios con los cuales se realizó el cálculo anterior.

Tabla 10: Calculo de la recomendación de sitio.

	Pearson (P)	Puntuación - Sondor Raymi	P* P-Sondor Raymi	Puntuación-Bosque de Piedras	P*P-Bosque de Piedras	Puntuación-Señor de Huanca	P*P-Señor de Huanca	Puntuación - Pukllay	P*P-Puntuación de Pukllay	Puntuación de Plza de Armas de Andahuylas	P*P-Plaza de Armas de Andahuylas
Maria	0.00	2	**0.00**	2.00	**0.00**	3	**0.00**	1	**0.00**	2	**0.00**
Reynaldo	1.00	4	**4.00**		**0.00**		**0.00**		**0.00**		**0.00**
Jose	1.00	3	**3.00**	3.00	**3.00**		**0.00**		**0.00**		**0.00**
Macarena	1.00	2	**2.00**	3.00	**3.00**	3	**3.00**	3	**3.00**	4	**4.00**
Pilar	0.00		**0.00**	4.00	**0.00**	2	**0.00**		**0.00**	3	**0.00**
Total			**9.00**		**6.00**		**3.00**		**3.00**		**4.00**
Sum.Pearson			**3.00**		**2.00**		**1.00**		**1.00**		**1.00**
Total/Sum. Pearson			**3.00**		**3.00**		**3.00**		**3.00**		**4.00**

Fuente: Elaboración Propia.

1 Se colocan los resultados del cálculo del Coeficiente de Correlación de Pearson.

2 Calificación de cada atractivo turístico por cada usuario.

3 Este resultado se obtiene al multiplicar el Coeficiente de Correlación de Pearson con la calificación de cada atractivo turístico por usuario.

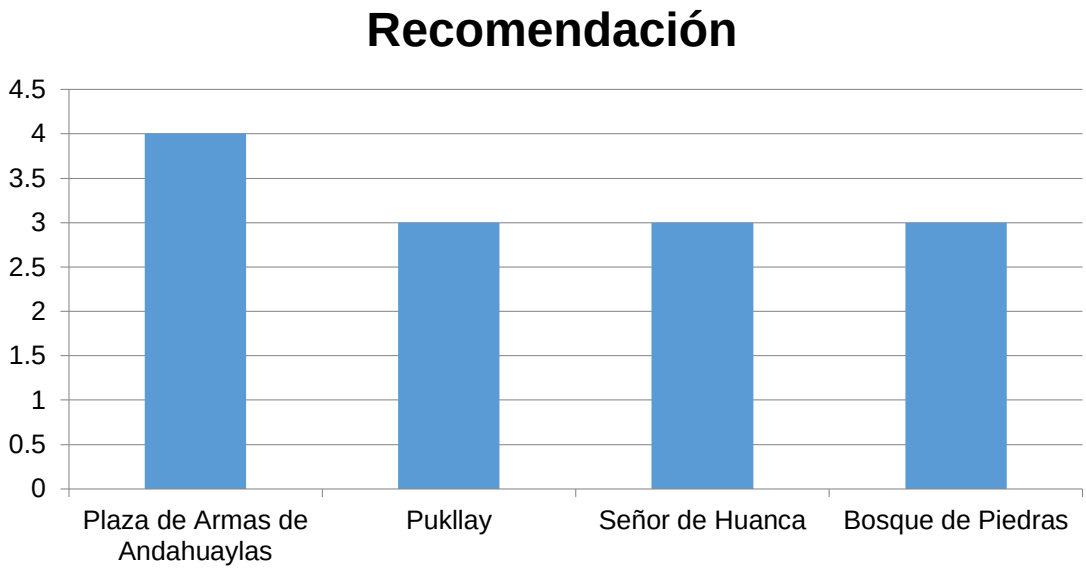
4 Sumatoria de producto del punto 3.

5 Sumatoria del coeficiente de correlación de Pearson, pero solo de los que registran calificación.

6 División entre los resultados de punto 4 y el punto 5, este resultado es que indica cual sitio es el q será recomendado.

Según el cálculo realizado los atractivos recomendados para el Usuario Rafael serán mostrados en el siguiente orden:

Grafico 15: Reprecentación estadistica de resultados obtenidos.



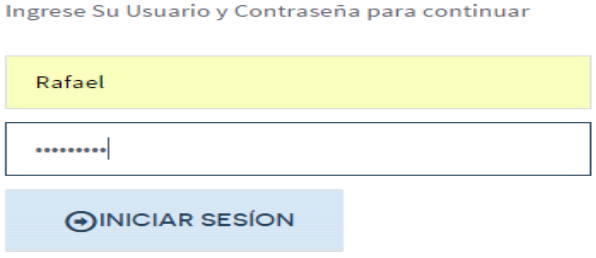
Fuente: Elaboración Propia.

Seguidamente se realizara la verificación con el módulo de recomendación del sitio web para comprobar la correcta funcionalidad del algoritmo.

Paso N° 1: Inicio de sesión:

- Solo los usuarios que tengan registradas una cuenta en el sitio web y hayan iniciado sesión podrán ser recomendados.

Grafico 16: Incio de sesión del sitio web.



Fuente: Elaboración Propia.

Paso N° 2: Calificación de Sitios.

- Las calificaciones solo la pueden realizar los usuarios que han iniciado sesión en el sitio web.

Grafico 17: Calificación de sitio turistico.

Fuente: Elaboración Propia.

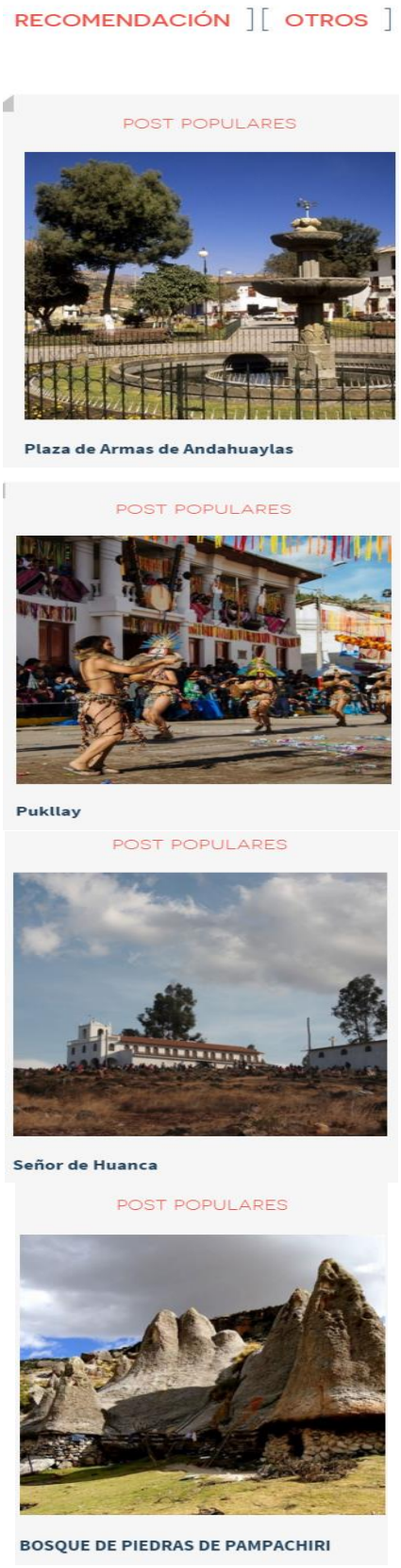
Grafico 18: Calificación de sitio turistico.

Fuente: Elaboración Propia.

Paso N° 3: Resultados obtenidos

- Estos resultados solo serán emitidos a usuarios que hayan calificado a por lo menos 2 sitios, caso contrario solo se le mostraran sugerencias de otros sitios aleatoriamente.

Grafico 19: Resultados obtenidos en el sitio web.



Se realizó una captura de pantalla a los resultados obtenidos en el sitio web, como muestra de que los resultados coinciden con los calculados realizados en la hoja de cálculo de Excel.

En primera instancia se muestra la plaza de armas de Andahuaylas que según el cálculo fue la más recomendada por los usuarios más cercanos.

Segundo se muestran en orden correlativo las otros atractivos turísticos que obtuvieron puntuaciones similares.

Fuente: Elaboración Propia

Segundo caso

Tabla 11: Calificación de los usuarios primera prueba.

N°	Usuario	Sondor Raymi	Laguna de Pacucha	Bosque de Piedras	Baños termales de Hualalachi	Señor de Huanca	Pukllay	Plaza de Armas de Andahuylas
1	Maria	2	3	2	2	3	1	2
2	Reynaldo	4	4		5			
3	Andrea	5		5	4	3	3	4
4	Flora		3	2		2		2
5	Julio	4		3	4	4		
6	Carlos	1	2			1	2	
7	Jose	3	2	3	3			
8	Macarena	2	2	3	4	3	3	4
9	Pilar		2	4	1	2		3
10	Rafael		3	5	4			

Fuente: Elaboración Propia.

1 Son calificaciones o votaciones de cada usuario por cada atractivo turístico.

2 Es la calificación o votación emitida por el usuario Rafael.

Paso N° 1: Calculamos la distancia con cada usuario y hallamos el vecino más cercano.

- Veremos que el cálculo se realiza solo con usuarios o vecinos con los cuales el usuario Rafael tenido por lo menos 2 intersecciones.

Tabla 12 Calculo de la distancia entre vecinos

	Laguna de Pacucha	Bosque de Piedras	Baños termales de Hualalachi	
María	3	2	2	
Rafael	3	5	4	
			C.C.P	-0.87

	Laguna de Pacucha	Baños termales de Hualalachi	
Reynaldo	4	5	
Rafael	3	4	
		C.C.P	1.00

	Bosque de Piedras	Baños termales de Hualalachi	
Andrea	5	4	
Rafael	5	4	
		C.C.P	1.00

	Laguna de Pacucha	Bosque de Piedras	
Flora	3	2	
Rafael	3	5	
		C.C.P	-1.00

	Bosque de Piedras	Baños termales de Hualalachi	
Julio	3	4	
Rafael	5	4	
		C.C.P	-1.00

	Laguna de Pacucha	Bosque de Piedras	Baños termales de Hualalachi	
Jose	2	3	3	
Rafael	3	5	4	
			C.C.P	0.87

	Laguna de Pacucha	Bosque de Piedras	Baños termales de Hualalachi	
Macarena	2	3	4	
Rafael	3	5	4	
			C.C.P	0.50

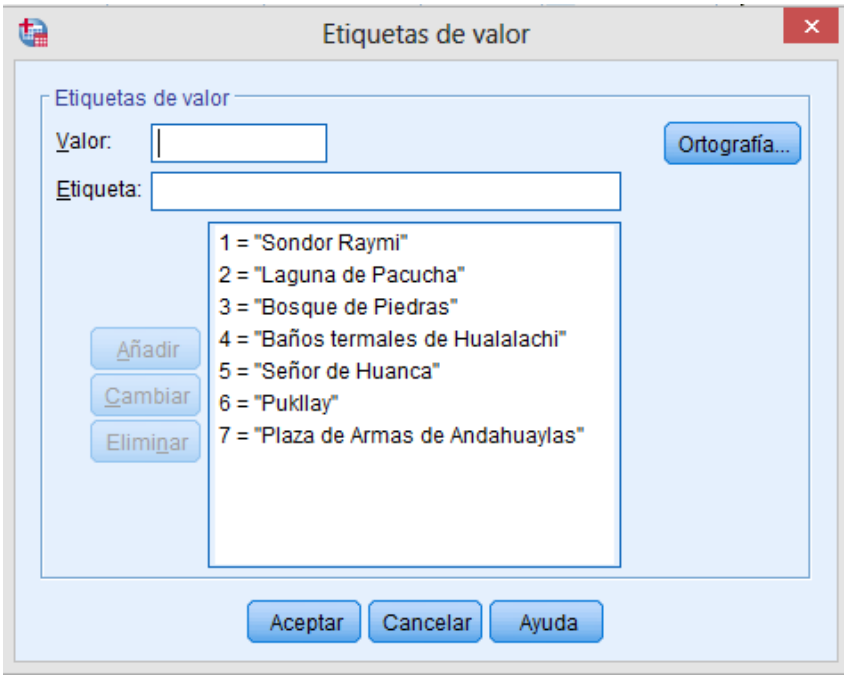
	Laguna de Pacucha	Bosque de Piedras	Baños termales de Hualalachi	
Pilar	2	4	1	
Rafael	3	5	4	
			C.C.P	0.65

Fuente: Elaboración Propia.

Paso N° 2: Calculo de la recomendación sitios al nuevo usuario

- Seguidamente se realiza el cálculo para saber que sitios serán recomendados al usuario para lo cual se coloca todos los sitios q aún no han sido visitados por el usuario y sus puntuaciones respectivas de los usuarios con los cuales se realizó el cálculo anterior.

Grafico N° 20: Asignación de Códigos a Los atractivos turísticos.



Fuente: Elaboración Propia.

Grafico N° 21: Cuadro de datos ingresados en la herramienta SPSS.

Atractivos	Maria	Reynaldo	Andrea	Flora	Julio	Carlos	Jose	Macarena	Pilar	Rafael
1	2	4	5	.	4	1	3	2	.	.
2	3	4	.	3	.	2	2	2	2	3
3	2	.	5	2	3	.	3	3	4	5
4	2	5	4	.	4	.	3	4	1	4
5	3	.	3	2	4	1	.	3	2	.
6	1	.	3	.	4	2	.	3	.	.
7	2	.	4	2	.	.	.	4	3	.

Fuente: Elaboración Propia.

Tabla 13: Resultado del cálculo de coeficiente de correlación de Pearson entre usuarios. Con la herramienta SPSS.

		María	Reynaldo	Andrea	Flora	Julio	Carlos	José	Macarena	Pilar	Rafael
María	Correlación de Pearson	1	-,500	,000	,577	,000	-,302	-1,000**	-,296	-,320	-,866
	Sig. (bilateral)		,667	1,000	,423	1,000	,698	,000	,520	,599	,333
	N	7	3	6	4	5	4	4	7	5	3
Reynaldo	Correlación de Pearson	-,500	1	-1,000**	.b	.b	.b	,500	1,000**	-1,000**	1,000**
	Sig. (bilateral)	,667		.	.	.	.	,667	,000	.	.
	N	3	3	2	1	2	2	3	3	2	2
Andrea	Correlación de Pearson	,000	-1,000**	1	.b	-,559	-,500	.b	-,297	,632	1,000**
	Sig. (bilateral)	1,000	.		,000	,327	,667	,000	,568	,368	.
	N	6	2	6	3	5	3	3	6	4	2
Flora	Correlación de Pearson	,577	.b	.b	1	.b	1,000**	-1,000**	-,816	-,522	-1,000**
	Sig. (bilateral)	,423	.	,000		.	.	.	,184	,478	.
	N	4	1	3	4	2	2	2	4	4	2
Julio	Correlación de Pearson	,000	.b	-,559	.b	1	.b	.b	,000	-,945	-1,000**
	Sig. (bilateral)	1,000	.	,327	.		,000	,000	1,000	,212	.
	N	5	2	5	2	5	3	3	5	3	2
Carlos	Correlación de Pearson	-,302	.b	-,500	1,000**	.b	1	-1,000**	,000	.b	.b
	Sig. (bilateral)	,698	.	,667	.	,000		.	1,000	.	.
	N	4	2	3	2	3	4	2	4	2	1
José	Correlación de Pearson	-1,000**	,500	.b	-1,000**	.b	-1,000**	1	,522	,189	,866
	Sig. (bilateral)	,000	,667	,000	.	,000	.		,478	,879	,333
	N	4	3	3	2	3	2	4	4	3	3

Macarena	Correlación de Pearson	-,296	1,000**	-,297	-,816	,000	,000	,522	1	-,105	,500
	Sig. (bilateral)	,520	,000	,568	,184	1,000	1,000	,478		,867	,667
	N	7	3	6	4	5	4	4	7	5	3
Pilar	Correlación de Pearson	-,320	-1,000**	,632	-,522	-,945	.b	,189	-,105	1	,655
	Sig. (bilateral)	,599	.	,368	,478	,212	.	,879	,867		,546
	N	5	2	4	4	3	2	3	5	5	3
Rafael	Correlación de Pearson	-,866	1,000**	1,000**	-1,000**	-1,000**	.b	,866	,500	,655	1
	Sig. (bilateral)	,333	.	.	.	.	.	,333	,667	,546	
	N	3	2	2	2	2	1	3	3	3	3

Fuente: Elaboración Propia.

- Los Usuarios Rafael y María no se correlacionan o no tienen ninguna relación.
- Los Usuarios Rafael y Reynaldo se correlacionan o tienen una relación fuerte.
- Los Usuarios Rafael y Andrea se correlacionan o tienen una relación fuerte.
- Los Usuarios Rafael y Flora no se correlacionan o no tienen una relación fuerte.
- Los Usuarios Rafael y Julio no se correlacionan o no tienen una relación fuerte.
- Los Usuarios Rafael y José se correlacionan o tienen una relación alta.
- Los Usuarios Rafael y Macarena se correlacionan o tienen una relación moderada.
- Los Usuarios Rafael y Pilar se correlacionan o tienen ninguna relación modera.

Tabla 14: Calculo de la recomendación de sitio.

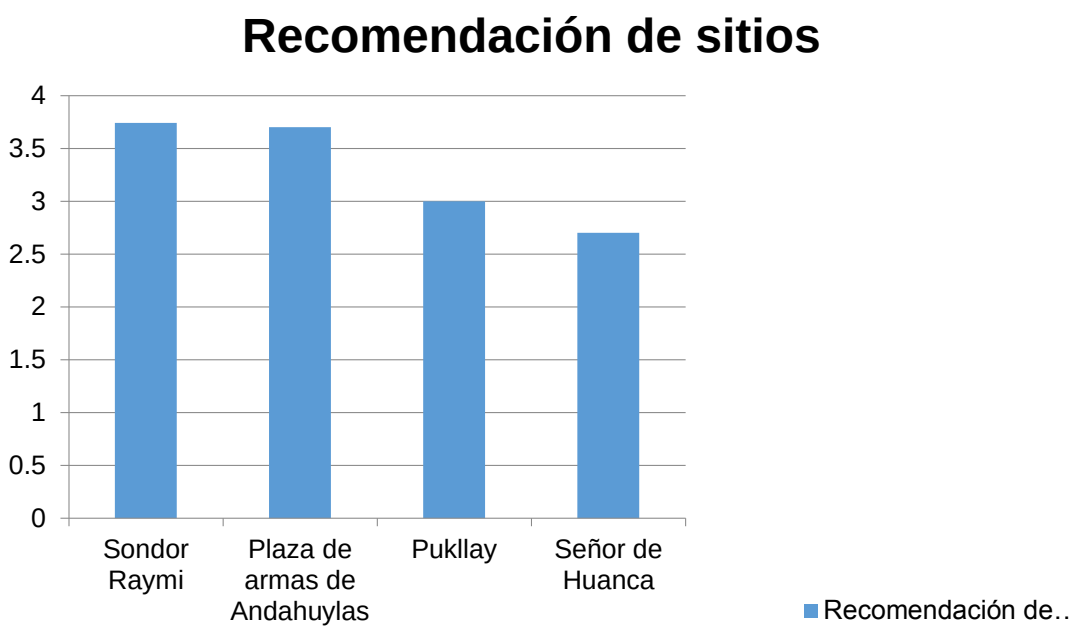
	Pearson (P)	Puntuación - Sondor Raymi	P* P-Sondor Raymi	Puntuación-Señor de Huanca	P*P-Señor de Huanca	Puntuación - Pukllay	P*P-Puntuación de Pukllay	Puntuación de Plza de Armas de Andahuylas	P*P-Plaza de Armas de Andahuylas
Maria	0.00	2	0.00	3	0.00	1	0.00	2	0.00
Reynaldo	1.00	4	4.00		0.00		0.00		0.00
Andrea	1.00	5	5.00	3	3.00	3	3.00	4	4.00
Flora	0.00		0.00	2	0.00		0.00	2	0.00
Julio	0.00	4	0.00	4	0.00	4	0.00		0.00
Jose	0.87	3	2.60		0.00		0.00		0.00
Macarena	0.50	2	1.00	3	1.50	3	1.50	4	2.00
Pilar	0.65		0.00	2	1.31		0.00	3	1.96
Total			12.60		5.81		4.50		7.96
Sum.Pearson			3.37		2.15		1.50		2.15
Total/Sum. Pearson			3.74		2.70		3.00		3.70

Fuente: Elaboración Propia.

1. Se colocan los resultados del cálculo del Coeficiente de Correlación de Pearson.
2. Calificación de cada atractivo turístico por cada usuario.
3. Este resultado se obtiene al multiplicar el Coeficiente de Correlación de Pearson con la calificación de cada atractivo turístico por usuario.
4. Sumatoria de producto del punto 3.
5. Sumatoria del coeficiente de correlación de Pearson, pero solo de los que registran calificación.
6. División entre los resultados de punto 4 y el punto 5, este resultado es que indica cual sitio es el q será recomendado.

Según el cálculo realizado los atractivos recomendados para el Usuario Rafael serán mostrados en el siguiente orden:

Grafico 22: Reprecentación estadistica de resultados obtenidos.



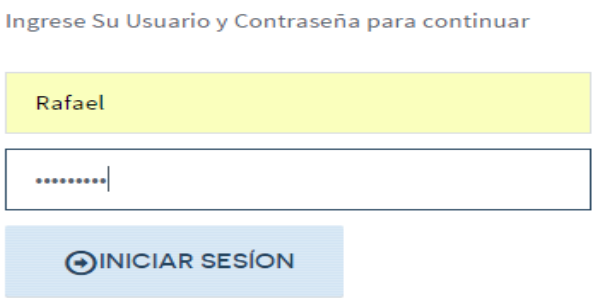
Fuente: Elaboración Propia.

Seguidamente se realizara la verificación con el módulo de recomendación del sitio web para comprobar la correcta funcionalidad del algoritmo.

Paso N° 1: Inicio de sesión:

- Solo los usuarios que tengan registradas una cuenta en el sitio web y hayan iniciado sesión podrán ser recomendados.

Grafico 23: Incio de sesión del sitio web.



Fuente: Elaboración Propia.

Paso N° 2: Calificación de Sitios.

- Las calificaciones solo la pueden realizar los usuarios que han iniciado sesión en el sitio web.

Grafico 24: Calificación de sitio turistico.

Fuente: Elaboración Propia.

Grafico 25: Calificación de sitio turistico.

Fuente: Elaboración Propia.

Grafico 26: Calificación de sitio turistico.

Fuente: Elaboración Propia.

Paso N° 3: Resultados obtenidos

- Estos resultados solo serán emitidos a usuarios que hayan calificado a por lo menos 2 sitios, caso contrario solo se le mostraran sugerencias de otros sitios aleatoriamente.

Grafico 27: Resultados obtenidos en el sitio web.

Se realizó una captura de pantalla a los resultados obtenidos en el sitio web, como muestra de que los resultados coinciden con los calculados realizados en la hoja de cálculo de Excel.

En primera instancia se muestra la plaza de armas de Andahuaylas que según el cálculo fue la más recomendada por los usuarios más cercanos.

Segundo se muestran en orden correlativo las otros atractivos turísticos que obtuvieron puntuaciones similares.

Fuente: Elaboración Propia.

5.3 Resultado 3:

Según el tercer objetivo “Desarrollar el módulo interacción con el turista para compartir y consultar información de diversos atractivos turísticos y planificar visitas a dichos atractivos.

Se desarrolló he implemento el módulo de interacción con el turista, este módulo estaba basado en los conocimientos de inteligencia colectiva, esto quiere decir, que todos sabemos algo que el resto desconoce.

El usuario tiene la capacidad de compartir información sobre atractivos turísticos, en este caso de la provincia de Andahuaylas, y sentir que es una pieza fundamental en la difusión del turismo de su provincia.

Los usuarios también pueden planificar visitas, a destinos o eventos a los cuales no hayan tenido la oportunidad de visitar o asistir.

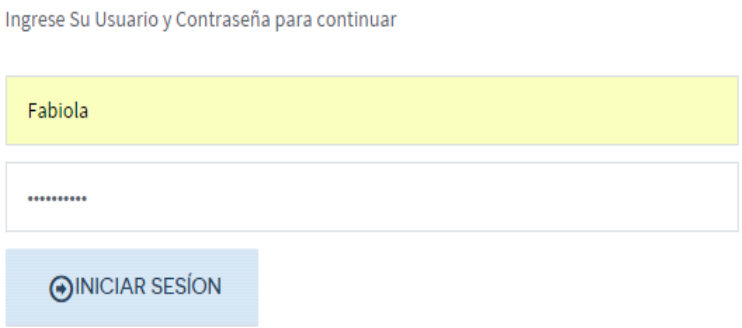
Todo lo mencionado anteriormente contribuye a que el usuario interesado interactúe con más frecuencia con el sitio web es decir, crear en el usuario un interés respecto al sitio web para que pueda seguir siendo parte de él.

Seguidamente se muestra el modulo desarrollado

Paso N° 1: Resultados obtenidos

- Para tener acceso a este módulo, el usuario a debido de registrar una cuenta y haber iniciado sesión en el sitio web.

Grafico 28: Inicio de sesión al sitio web.



Fuente: Elaboración Propia.

Paso N° 2: Ingrese al panel de control.

- El usuario deberá ingresar al panel de control tal como se muestra en el grafico siguiente para acceder al módulo de interacción con el usuario.

Grafico 29: Ingreso al panel de control.

Fuente: Elaboración Propia.

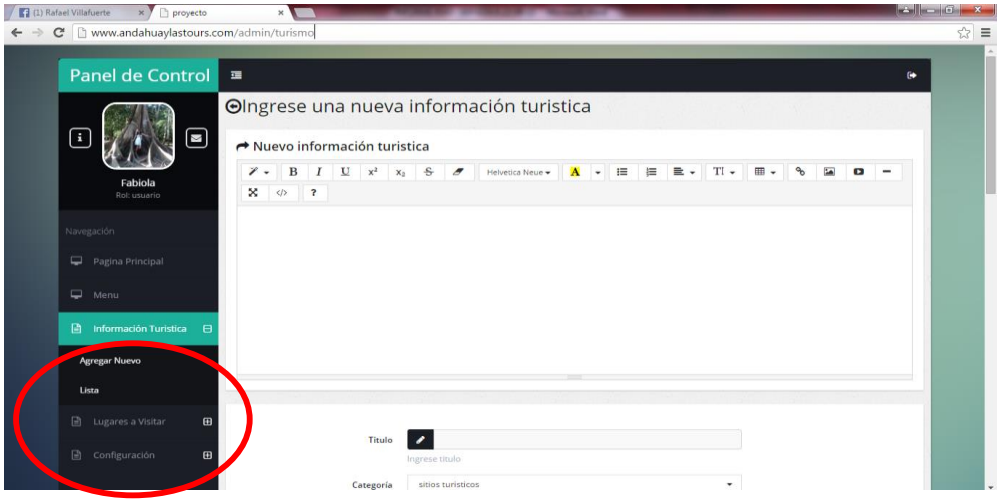
Paso N° 3: Panel de control

- Se muestran varias opciones para el usuario sin embargo las más importantes son el ítem de Información Turística (Agregar nuevo, lista y también el módulo de Planificación de visitas (mis lugares)

Paso N° 4: Panel de control

- Se muestra que el usuario tiene la opción de ingresar información que será publicada después de una previa revisión por los moderadores o Administradores.

Grafico 30: Opcion de usuario-Iingresar información.

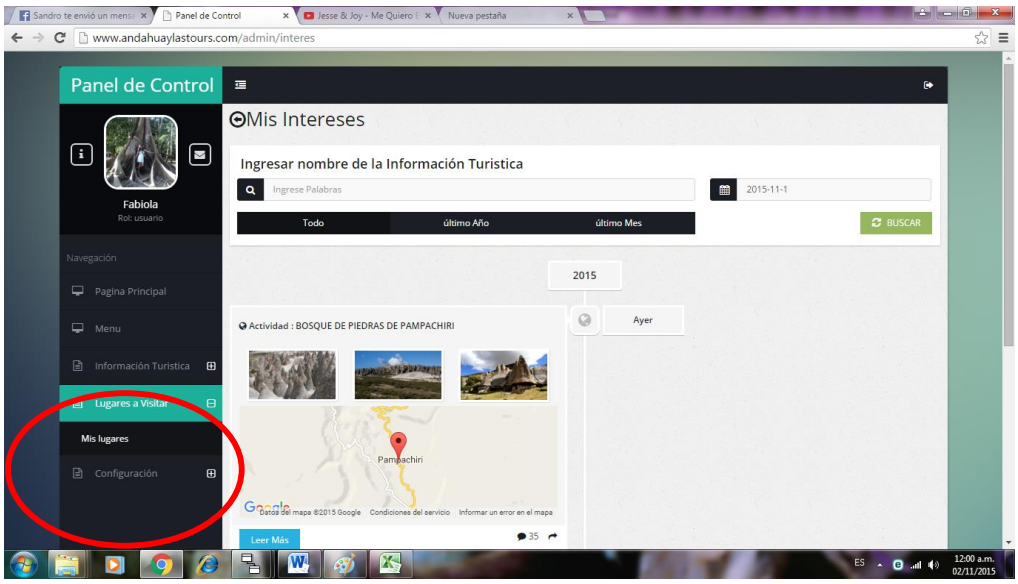


Fuente: Elaboración Propia.

Lo importante de este módulo es que le perite al usuario compartir información, dándole al usuario control sobre qué información desea compartir.

- Se muestra al usuario la opción de Lugares a visitar en donde le permite al usuario observar el registro de visitas que tenía pensado realizar.

Grafico 31: Opcion de usuario-Lugares a visitar



Fuente: Elaboración Propia.

En el Anexo 4 se encuentra de forma mas especifica todas las bondades que se presenta en la aplicación.

La visualización de todo el interfaz se puede apreciar mejor en la siguiente dirección web www.andahuaylastours.com.

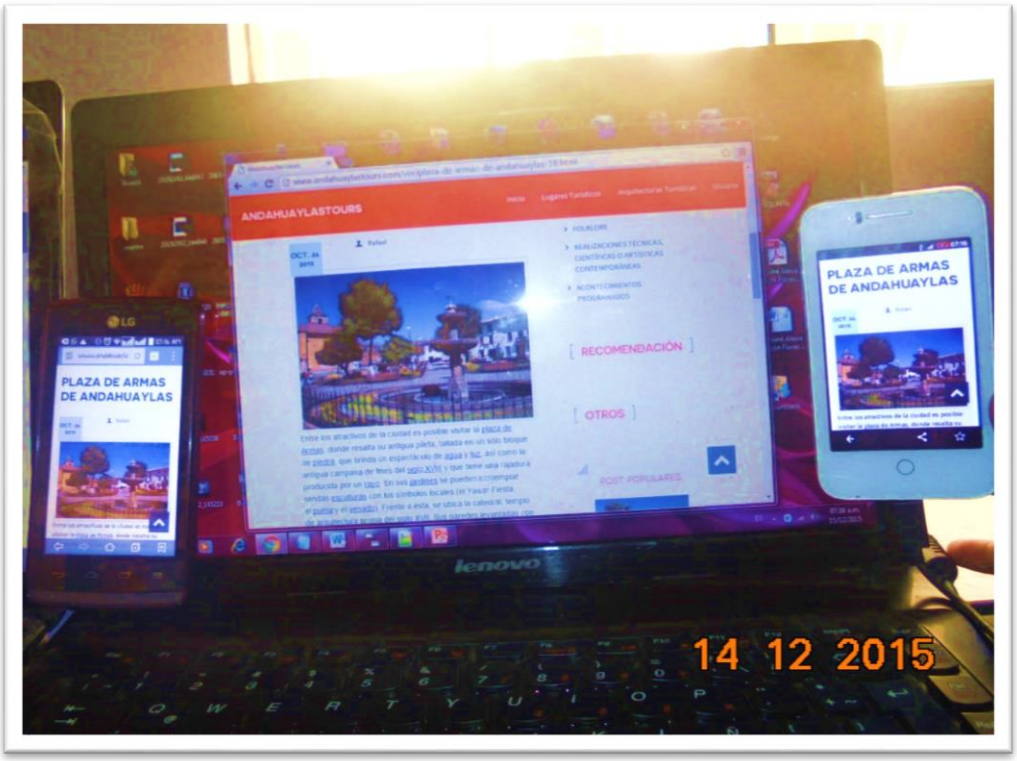
5.4 Resultado 4:

Según el cuarto objetivo “Realizar una prueba piloto de la aplicación para determinar su adaptabilidad.”.

Se ha subido la aplicación a la web para los usuarios puedan acceder desde cualquier dispositivos móvil y comprobar que el sitio web es adaptable a cualquier dispositivo.

- En el siguiente grafico observamos que el sitio web e adapta al tamaño de la pantalla del dispositivo.

Grafico 32: Adaptavilidad de dispositivos moviles.



Fuente: Elaboración Propia.

- En del siguiente grafico se observa que se ha probado la adaptabilidad en dispositivos móviles diferentes.

Grafico 33: Adaptavilidad según tamaño de pantalla.

- Dispositivo móvi LG Joy
- Sistema Operativo: Android
- Naavegador: Google Grome

- Dispositivo móvi Alcatel onetouch
- Sistema Operativo: Firefox OS 1.3.0.0
- Naavegador: Firefox.

Fuente: Elaboración Propia.

En el Anexo 5 se muestran 2 equipos diferentes en las cuales la aplicación web demuestra que tiene un diseño web adaptable.

CAPÍTULO VI: CONCLUSIONES

6.1 PRIMERO:

La base de datos de la aplicación “Responsive Web Design Para Planificación y Recomendación Turística Aplicando Inteligencia Colectiva” Es una base de datos relacional muy concisa y precisa, debido a que para su diseño fue sometida a un conjunto de reglas de análisis y trasformación de las estructuras de datos que minimizaron la redundancia y maximizaron la estabilidad.

6.2 SEGUNDO:

Las teorías de inteligencia colectivas aplicadas al desarrollo de aplicaciones web permiten tener en actividad la base de datos con ayuda de todos los usuarios, basándonos en el principio que “todos sabemos sobre algo que el resto desconoce”.

6.3 TERCERO

Las aplicaciones que hacen uso de algoritmos de filtrado colaborativo como parte de su desarrollo, hacen que el sitio web sea más interactivo, dándole al usuario no solo la oportunidad de interactuar con el sitio web en sí, sino la oportunidad de interactuar con otros usuarios que pertenecen al mismo sitio web, en este caso el usuario tiene el poder de transmitirle a otro usuario sus experiencias en el módulo de recomendación, y planificar nuevas visitas según las sugerencias de otros usuarios.

6.4 CUARTO

En la prueba piloto realizada con el fin de desmostar la portabilidad y/o adaptabilidad de la página web, se demostró que este tiene la capacidad de adaptarse o cambiar ciertos contenidos dependiendo del dispositivo móvil que lo accede, en donde el usuario puede leer la información de forma cómoda, sin tener que hacer muchos movimientos de Scroll (desplazamiento, rollo o voluta al movimiento en 2D).

CAPÍTULO VII: RECOMENDACIONES

7.1 PRIMERO

Es recomendable tener un buen diseño de base de datos, que contenga todos los datos necesarios requeridos, ya que será la base de todo el desarrollo de la aplicación web, en donde se almacenara toda la data ingresada por el usuario, y de la cual se extraerá data para ser mostrada y/o en ocasiones ser procesada para un nuevo almacenamiento

7.2 SEGUNDO

Es recomendable que se sigan aplicando las teorías de Inteligencia colectiva en el desarrollo de páginas web, debido a que permite crear relaciones redituales entre el usuario y la página web desarrollada.

7.3 TERCERO

Es recomendable que en futuros estudios se apliquen y prueben otros algoritmos de filtrado colaborativo, para ser puestos a prueba y aplicarlos en el diseño de páginas web.

7.4 CUARTO

Es recomendable que si desea desarrollar páginas web que tengan la posibilidad de adaptarse a cualquier dispositivo, usemos la opción del Diseño Web Adaptable, ya que como se menciona a lo largo del informe, tiene muchas ventajas que facilitan la navegación del usuario en la página web.

REFERENCIAS BIBLIOGRÁFICAS

Aguilar Domínguez, A. (2009). *Base de Datos con interfaz web para el departamento de multimedios y aplicaciones interactivas del museo Universum.* Mexico D. F.: Centro de Publicaciones de la Universidad Nacional Autónoma de México.

Betarte, L., Machado, R., & Molina, V. (2006). *PGMúsica Sistema de Recomendación de Música.* Uruguay.

Bootstrap. (2015). *Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web.* Obtenido de Learn about the project's history, meet the maintaining teams, and find out how to use the Bootstrap brand.: http://getbootstrap.com/

Campos Paré, R. (2002). *Base de datos.* Editorial UOC.

Cegarra Sánchez, J. (2011, pag. 42). *Metodología de la investigación científica y técnologica.* Ediciones Día de Santos.

Cobo, Á. (2005). *PHP y MySQL: Tecnología para el desarollo de aplicaiones Web.* Santander: Ediciones Días de SAntos.

DIRCETUR. (2014). *Apurimác destino turistico de deportes y aventura*. Obtenido de Turismo : http://dirceturapurimac.gob.pe/web/turismo/funciones/

Eslava Muñoz, V. J. (2013). *El nuevo PHP. Conceptos avanzados.*

Formoso Lopéz, V. (2013). *Técnicas eficientes para la recomendación de productos basadas en filtrado colaborativo.* Coruña.

Goge Rito, J. L. (2012). *Elaboración de un Sitio Web para el Programa Nacional de Asistencia al Turista en el Instituto Guatemalteco de Turismo.* Guatemala: Servicios de Publicaciones de la Universidad de Guatemala.

Guiza Ezkauriatza, M. (2011). *Trabajo colaborativo en la web:.* Palma de Mallorca.

Hobbs, L. (1999). *Diseñar su Propia Web.* Barcelona: Marcombo S.A.

Ísmodes, E. (2006). *Paises sin futuro: ¿que puede hacer la universidad?* Lima: Fondo Editorial PUCP.

Knafou, R., & Stock, M. (2003). *El turismo, factor de cambio territorial: Evolución de los lugares, actores y prácticas a lo largo del tiempo.* En la Costa de Aragues: Prensas Universitarias de Zaragoza.

Labrada Martinez, E., & Salgado Ceballos, C. (1 de Enero de 2013). *Diseño web Adpatativo o Responsivo*. Obtenido de http://www.revista.unam.mx/vol.14/num1/art07/art07.pdf

Lenguajes y Sistemas Informáticos. (s.f.). Obtenido de ¿Que es un framework web?: http://www.lsi.us.es/~javierj/investigacion_ficheros

León Osorio, R. (2008). *Base de datos relacionales Teoría y Práctica.* Mellin, Colombia: ITM.

Leyva Cortés, E., Prieto Tinoco, J. I., Samplo de la Torre, M. d., & Garzón Villar, M. L. (2006). *Sistemas y Ampliaciones Informáticas.* España: Mad, S.L.

Mariño Campos, R. (2005). *Diseño de páginas web y diseño gráfico: metodologias y técnicas para la implementación de sitios web y para el diseño gráfico.* España: IdeasPropias Editorial S. L.

Nevado Cabello, M. V. (s.f.). *Introducción a Las Bases de Datos Relacionales .* Madrid: Editorial Visión Libros.

Pardo Kuklinsk, H., & Cobo Romaní, C. (2007). *Planeta Web 2.0. Inteligencia colectiva o medios fast food.* Barcelona: LMI.

Pressman, R. S. (s.f.). *Ingeniería de Software: Un enfoque práctico.*

Ramos Martín, A., & Ramos Martín, M. J. (2014). *Web, Aplicaciones.* Madrid: Paraninfo, S.A.

Schenone, M. H. (2004). *Diseño de una Metodología Ágil de Desarrollo de Software.* Buenos Aires: Servicios de Publicacion de la Universidad de Buenos Aires.

Sommerville, I., & Alfonso Galipienso, M. I. (2005). *Ingenieria de Software.* Madrid: Pearson S.A.

Such Climent, M. P. (2008). Hacia el Desarrollo Sostenible del Turismo: Iniciativas y Significado. En F. Cebrián Abellan, *Turismo Rural y Desarrolo* (págs. 40-40). La mancha: Servicios de Publicación de la Universidad de la Castilla.

Vásquez, I. M. (2012). *Definición de un Framework para aplicciones Web con navegación sensible a Concerns.* Buenos Aires: Servicios de publicación de la Universidad Nacional de la Plata.

ANEXOS 1

BASE DE DATO NORMALIZADA.

En el siguiente grafico se muestra que nuestra base de datos cumple con los principios de normalización hasta su tercera forma normal, cuyos requisitos para que se encuentre en esta forma es cumplir los siguientes requisitos:

Una tabla está en Primera Forma Normal si:

- Todos los atributos son atómicos. Un atributo es atómico si los elementos del dominio son simples e indivisibles.
- La tabla contiene una clave primaria única.
- La clave primaria no contiene atributos nulos.
- No debe existir variación en el número de columnas.
- Los Campos no clave deben identificarse por la clave (Dependencia Funcional)
- Debe Existir una independencia del orden tanto de las filas como de las columnas, es decir, si los datos cambian de orden no deben cambiar sus significados.

Una tabla está en Segunda Forma Normal si:

- Si está en Primera Forma Normal (1FN).
- Los atributos que no forman parte de ninguna clave dependen de forma completa de la clave principal.

Una tabla está en Tercera Forma Normal si:

- La tabla está en la segunda forma normal (2NF)
- Ningún atributo no-primario de la tabla es dependiente transitivamente de una clave primaria.

Gráfico 34: Normalización de base de datos

RWD_CUENTADEUSUARIO

Id_Usuario	PrimerNom_Usuario	NomUsuario-Usuario	Contraseña_Usuario	Id_Rol
1	Erika	Erika.Samora	**********	1
2	Adelaida	Adelaida_12	**********	2

CUENTADEUSUARIO_USUARIODATO

Id_Usuariodato	Nombre_Usuariodato	Apellido_Usuariodato	Rutafoto_Usuariodato	Id_Usuario
7	Fabiola	Villegas	www.rutafotofabiola.com	26
8	Rafael	Villafuerte	www.rutafotorafael.com	27

CUENTADEUSUARIO_ROL

Id_Rol	Nombre_Rol
1	Turista
2	Moderador

RWD_INFORMACIÓNTURISTICA

Id_Info Turismo	Nombre_Info Turismo	Descripción_InfoTurismo	Id_Categoria	Id_Usuario	Id_Distrito	Estado	Latitud	Longitud
1	Laguna de Pacucha	La laguna de Pacucha es una de las más hermosas y misteriosas de la sierra central....	1	1	1	1		
2	Puckllay	Este es un evento nacional e internacional, en el cual se congregan una gran cantidad de comparsas que demuestran	2	2	2	2		

DEPARTAMENTO

Id_Departamento	Nombre_Departamento
7	Apurímac
8	Ayacucho

PROVINCIA

Id_Provincia	Id_Departamento	Nombre_Provincia
1	7	Andahuaylas
2	7	Abancay

DISTRITO

Id_Distrito	Id_Provincia	Nombre_Distrito
1	1	Pacucha
2	1	Andahuaylas

INFOTURISMO _ IMAGEN

ID_ImaTurismo	Id_InfoTurismo	Ruta_ImaTurismo
1	7	www.imagen1.php.com
2	6	www.imagen2.php.com

INFOTURISMO_CATEGORIA

Id_Categoria	Nombre_Categoria
1	Sitios turísticos
2	Fiestas costumbristas

INFOTURISMO_ESTADO

Id_Estado	Nombre_Estado
1	Publicado
2	Des publicado

RWD_PROGRAMACIÓNDEVISITAS

ID_Cronograma	Id_Usuario	Lugar_Lugar	Id_Fecha
1	1	1	20-12-2015
2	1	2	15-11-2105

RWD_PUNTUACIONES

Id_Puntuaciones	Id_Usuario	Id_InfoTurismo	Nro_Puntución
1	2	3	5
2	1	5	3

Fuente: Elaboración Propia

ANEXOS 2

ANÁLISIS DE LA DOCUMENTACIÓN RESPONSIVE WEB DESIGN PARA PLANIFICACIÓN Y RECOMENDACION TURÍSTICA APLICANDO INTELIGENCIA COLECTIVA

1. Requerimientos del Software

1.1. Requerimientos Funcionales

Los requerimientos funcionales de la aplicación se muestran en la siguiente tabla.

Tabla 15: Lista de requerimientos funcionales

REQUERIMIENTO DEL SISTEMA	DESCRIPCIÓN
Registrar cuenta de usuario	Un turista podrá registrase y poseer una cuenta de usuario en el sitio web. El administrador del sitio podrá asignarle al Usuario-Turista el rol de Usuario-Moderador según estime por conveniente, sabiéndose que una vez que se le asigne este último rol, el usuario tendrá mayores privilegios para con el sistema.
Registrar sugerencias sobre la diversidad turística y cultural.	El Usuario- Turista, podrá recomendar lugares o festividades en base a puntuaciones del 1 al 5 en donde: 1 = Malo 2 = Regular 3 = Bueno 4 = Muy Bueno 5 = Excelente Según sus experiencias vividas, esta información es sencillamente subjetiva, pero de mucha importancia.

Registrar información sobre diversidad turística y cultural	El Usuario-Turista, tendrá la opción de compartir (insertar y/o modificar) cualquier información referida a lugares turísticos, fiestas costumbristas entre otros. El Usuario- Moderador, tendrá la opción de validación de la información subida por el Usuario Turista, ya que es el quien decide si la información es publicada o denegada, este también asu vez puede contribuir con reforzar la información compartida por el usuario en el sitio web.
Reportes	El administrador es el único que puede acceder al módulo de reportes en el cual se mostraran - Número de visitas al sitio. - Numero de usuario registrados y cuáles son sus roles. - Cantidad de información contenida en el sitio. - Cantidad de información compartida por cada usuario. - Numero de aprobaciones o denegaciones realizadas por cada Usuario-Moderador, y el total de ellas.

Fuente: Elaboración Propia

1.2. Requerimientos no funcionales

Los requerimientos no funcionales de la aplicación se muestran en la siguiente tabla.

Tabla 16: Lista de requerimientos funcionales

REQUERIMIENTO DEL SISTEMA	DESCRIPCIÓN
Funcionalidad	- El sistema deberá soportar un aproximado de 20 usuarios simultáneos. - El sistema deberá prever un tiempo de acceso a los datos en menos de 20 segundos de empezado el acceso (sujeto a varianza de ancho de banda). - El sistema deberá completar el 100% de 1 transacción en 1 minuto.
Usabilidad	- El usuario deberá ser capaz de utilizar cualquier función del sistema, utilizando solamente los elementos de ayuda del sistema (manual de usuario) entre 5 a 20 minutos dependiendo del radio de conocimiento del usuario.
Portabilidad	- El software deberá poder ser visualizado desde cualquier dispositivo, quiere decir que la forma del software deberá adecuarse al tamaño de la pantalla del dispositivo por el

	cual se ha ingresado al sitio web.
Rendimiento	La carga de las páginas en el navegador: menor a 30 segundos. - Memoria RAM (en Mb): mayor a 15 Mb y menor de 100Mb (depende de la carga) - Base de datos: Depende de la cantidad de transacciones (datos).
Restricciones de diseño	Restricciones del entorno del sistema: - El software será elaborado en el lenguaje de programación PHP utilizando la herramienta NetBeans, utilizará el administrador de base de datos MySql. - Restricciones de Arquitectura - El sistema deberá ser desarrollado en arquitectura del Modelo Vista Controlador

Fuente: Elaboración Propia

2. Casos de Uso del Sistema

En esta sección se presentan los diagramas de casos de uso del sistema, obtenidos durante el proceso de especificación de requisitos, los cuales muestran en alto nivel las funcionalidades que el sistema realizara.

Inicialmente, se indica el diagrama de actores que interactúan con el sistema y posteriormente y la descripción de cada uno de los paquetes con sus respectivos diagramas de casos de uso.

2.1. Diagrama de actores del sistema:

A continuación en el grafico N° 35, se muestra el diagrama de actores del sistema.

Grafico 30: Diagrama de actores

Fuente: Elaboración Propia

2.1.1. Definición de los Actores

- Administrador

Tabla 117: Definición del actor Administrador

ACT -01	ADMINISTRADOR DEL SISTEMA
DESCRIPCIÓN	Es la persona encargada de administrar todo el sistema, se encarga de verificar que el sistema cumpla con la función para la cual fue creada.
COMENTARIO	Ninguno

Fuente: Elaboración Propia

- Moderador

Tabla 18: Definición del actor Usuario- Moderador

ACT -02	USUARIO - MODERADOR
DESCRIPCIÓN	Es la persona que se encarga de colaborar con el Administrador en el filtro de publicaciones que se harán en el sitio web, informaciones que

	son compartidas por el Usuario-Turista.
COMENTARIO	Ninguno

Fuente: Elaboración Propia

- Turista

Tabla 19: Definición del actor Usuario- Turista

ACT -01	**USUARIO – TURISTA**
DESCRIPCIÓN	Es la persona que colabora en incrementar la información existente sobre la diversidad turística y cultural, y realiza sugerencias mediante votaciones.
COMENTARIO	Ninguno

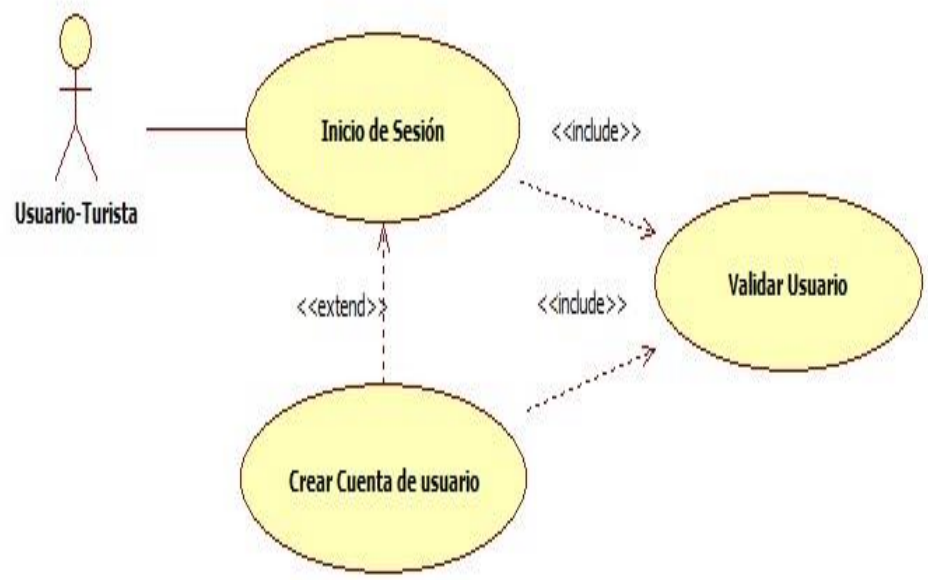
Fuente: Elaboración Propia

2.2. Diagramas de Casos de Uso

A continuación se muestran los diagramas de casos de uso del sistema.

2.2.1. Inicio y registro de Sesión Usuario-Turista

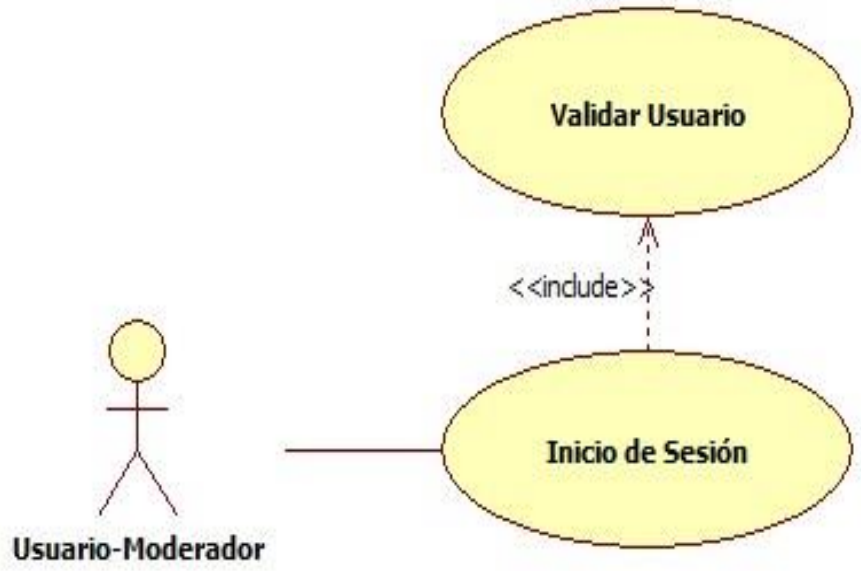
Grafico 36: Diagrama de caso de uso- Inicio de sesión-turista.



Fuente: Elaboración Propia

2.2.2. Inicio de Sesión Usuario-Moderador

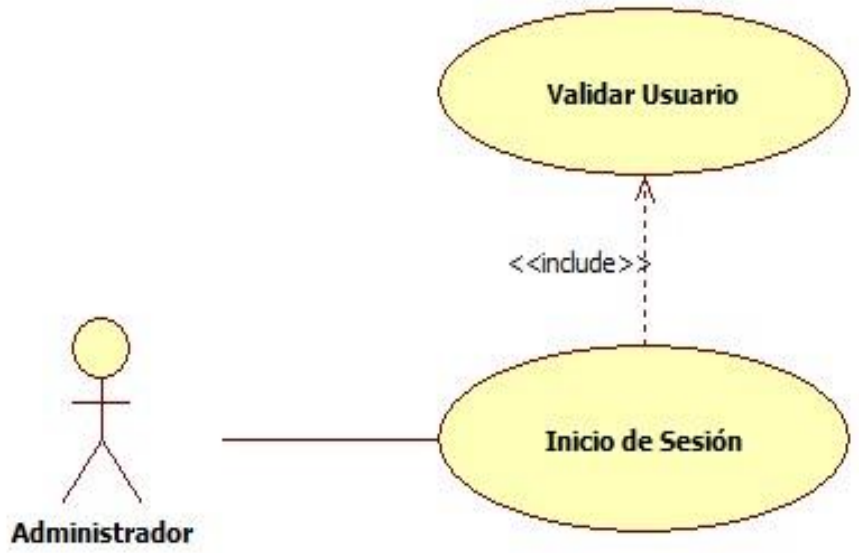
Grafico 37: Diagrama de caso de uso-Inicio de sesión-Moderador.



Fuente: Elaboración Propia

2.2.3. Inicio de Sesión Administrador

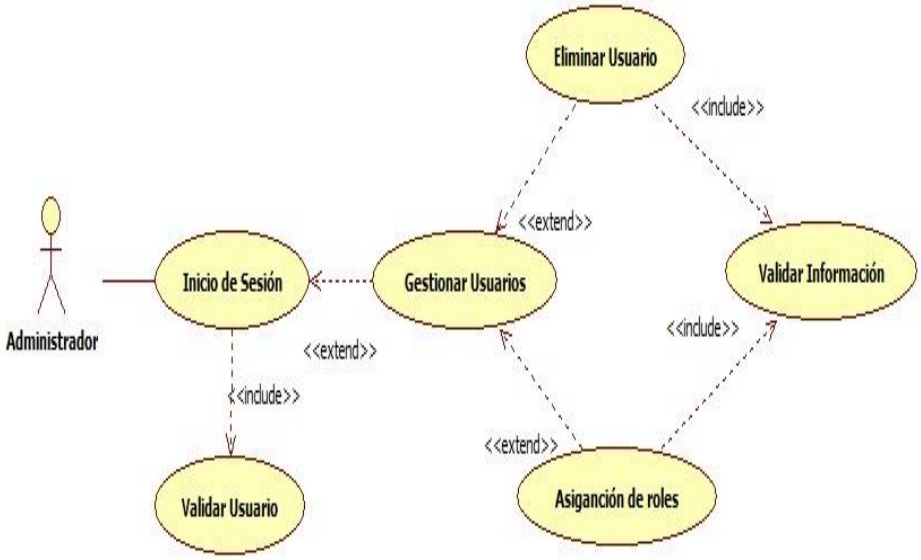
Grafico 38: Diagrama de caso de uso-Inicio de sesión-Administrador.



Fuente: Elaboración Propia

2.2.4. Gestión de Usuario

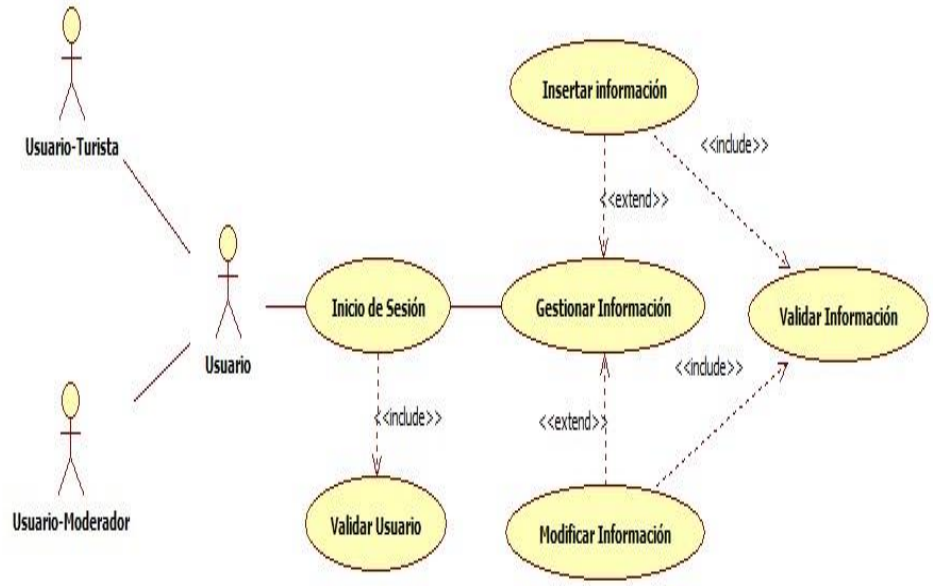
Grafico 39: Diagrama de caso de uso-Gestión de usuario.



Fuente: Elaboración Propia

2.2.5. Gestión de Información compartida por el Usuario-Turista-Moderador

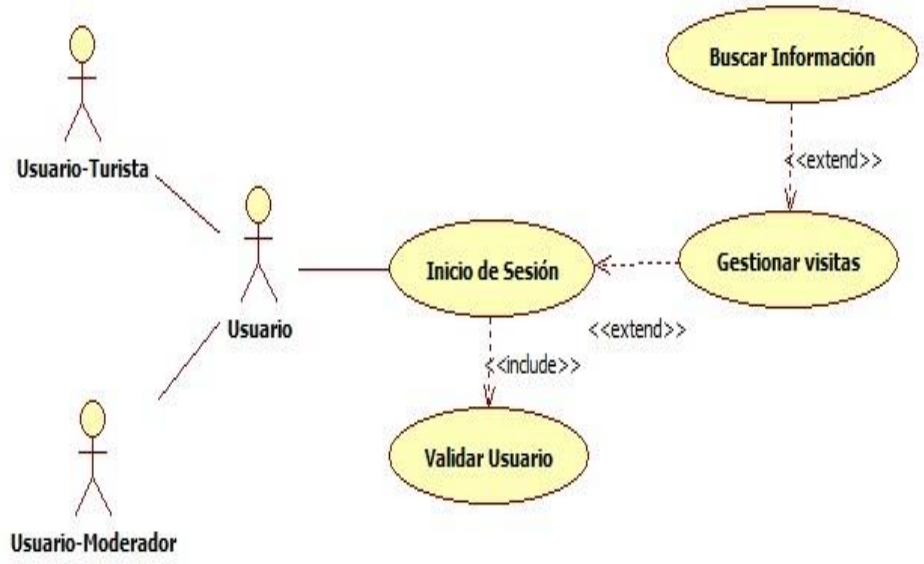
Grafico 40: Diagrama de caso de uso-Gestión de la información.



Fuente: Elaboración Propia

2.2.6. Gestionar Visitas

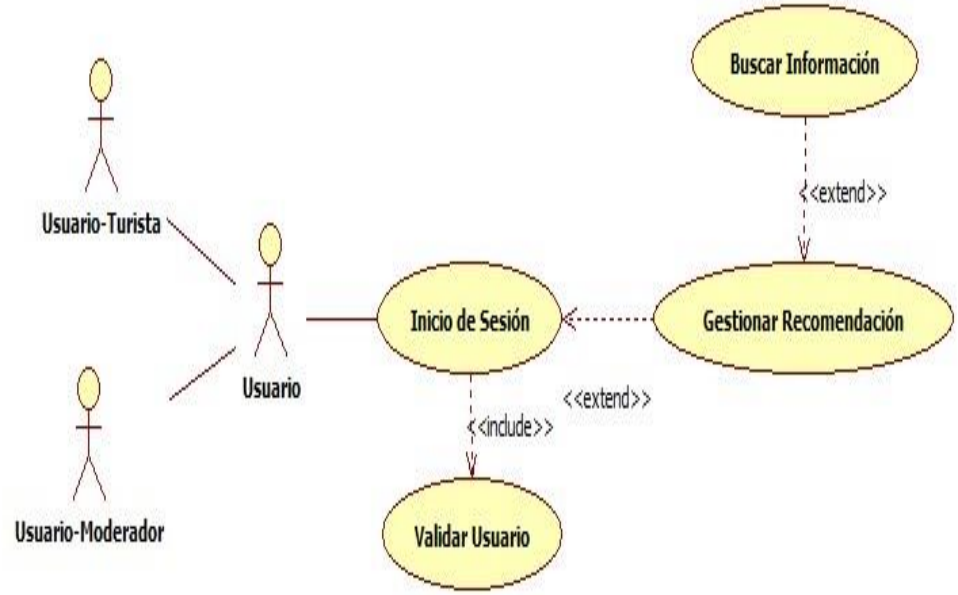
Grafico 41: Diagrama de caso de uso-Gestión de visitas.



Fuente: Elaboración Propia

2.2.7. Gestionar Recomendación

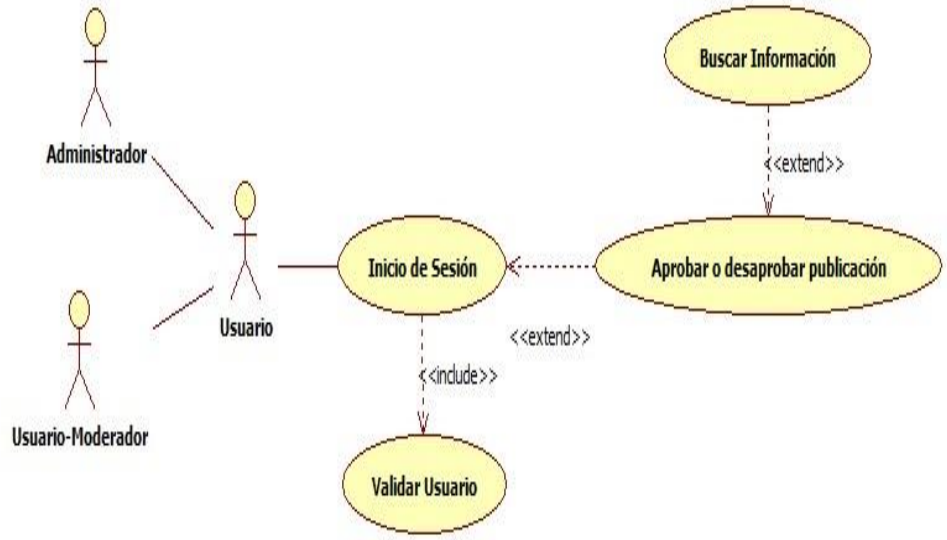
Grafico 42 Diagrama de caso de uso-Gestión de recomendaciones.



Fuente: Elaboración Propia

2.2.8. Aprobación o denegación de información compartida por el usuario.

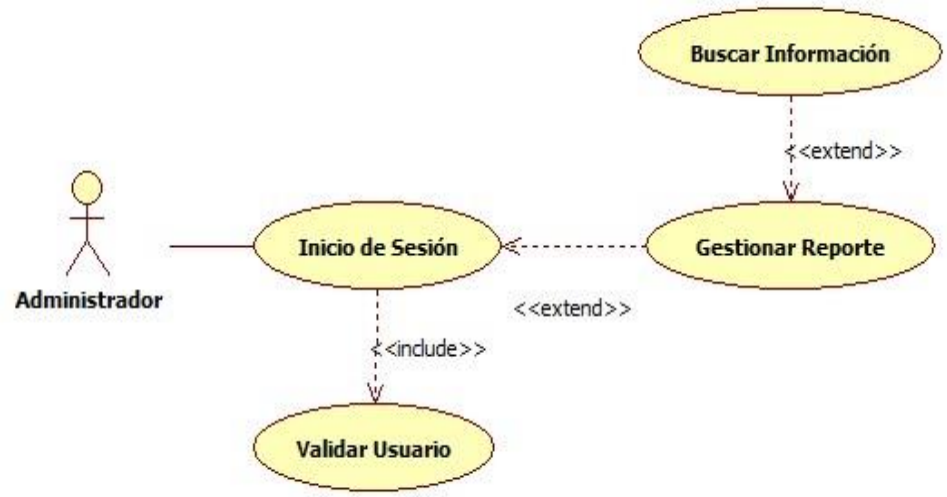
Grafico 43: Diagrama de caso de uso-Aprobar-Denegar-Información.



Fuente: Elaboración Propia

2.2.9. Gestionar reporte

Grafico 44: Diagrama de caso de uso -Gestión de reportes

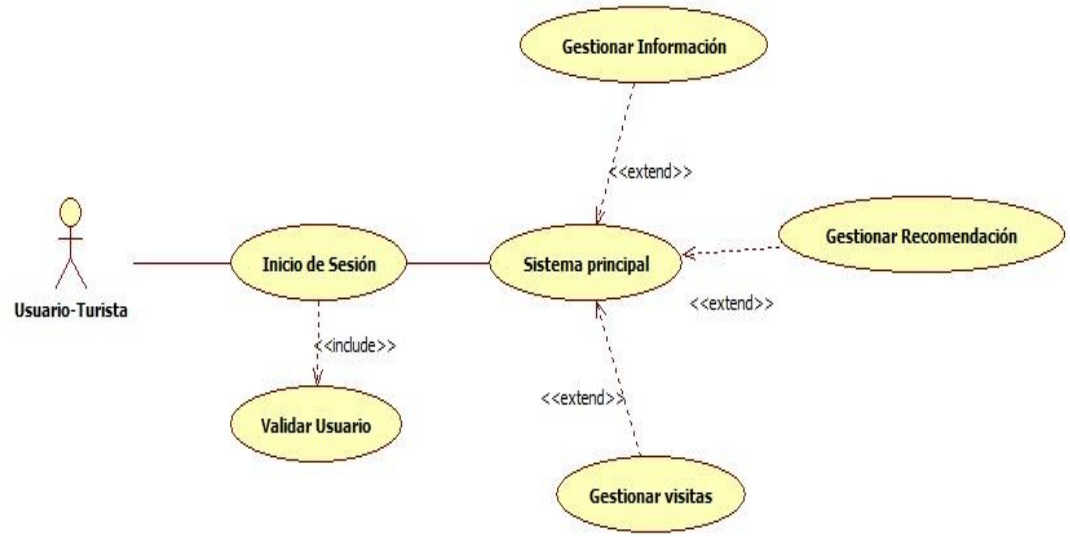


Fuente: Elaboración Propia

2.3. Diagrama de caso de uso general de opciones de usuario-turista

En el siguiente Grafico se muestra un diagrama general de opciones del Turista respecto al Sistema.

Grafico 45: Diagrama de caso de uso general-Opciones-Turista.

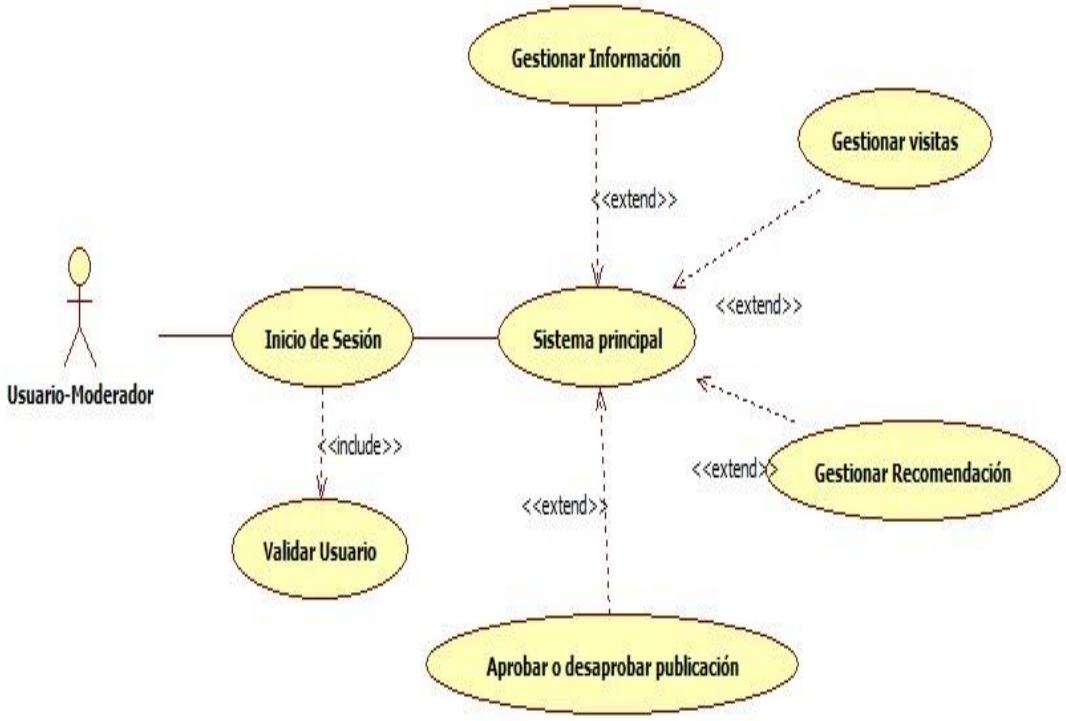


Fuente: Elaboración Propia

2.4. Diagramas de caso de uso general de opciones de usuario-moderador

En el siguiente Grafico se muestra un diagrama general de opciones del Moderador respecto al Sistema.

Grafico 46: Diagrama de caso de uso general-Opciones-Moderador.

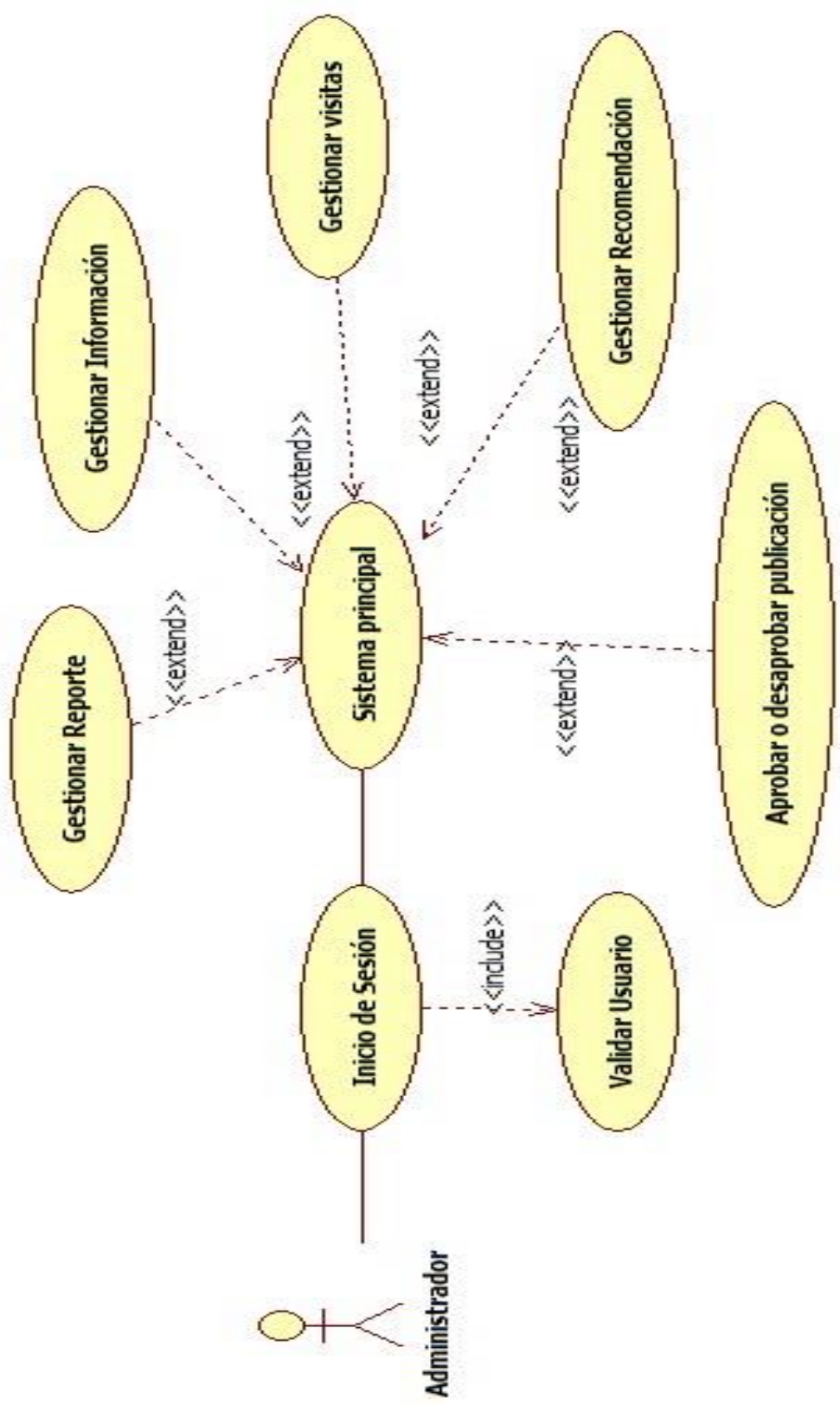


Fuente: Elaboración Propia

2.5. Diagramas general de opciones del administrador

En el siguiente Grafico se muestra un diagrama general de opciones del Moderador respecto al Sistema.

Grafico 47: Diagrama de caso de uso general-Opciones-Administrador

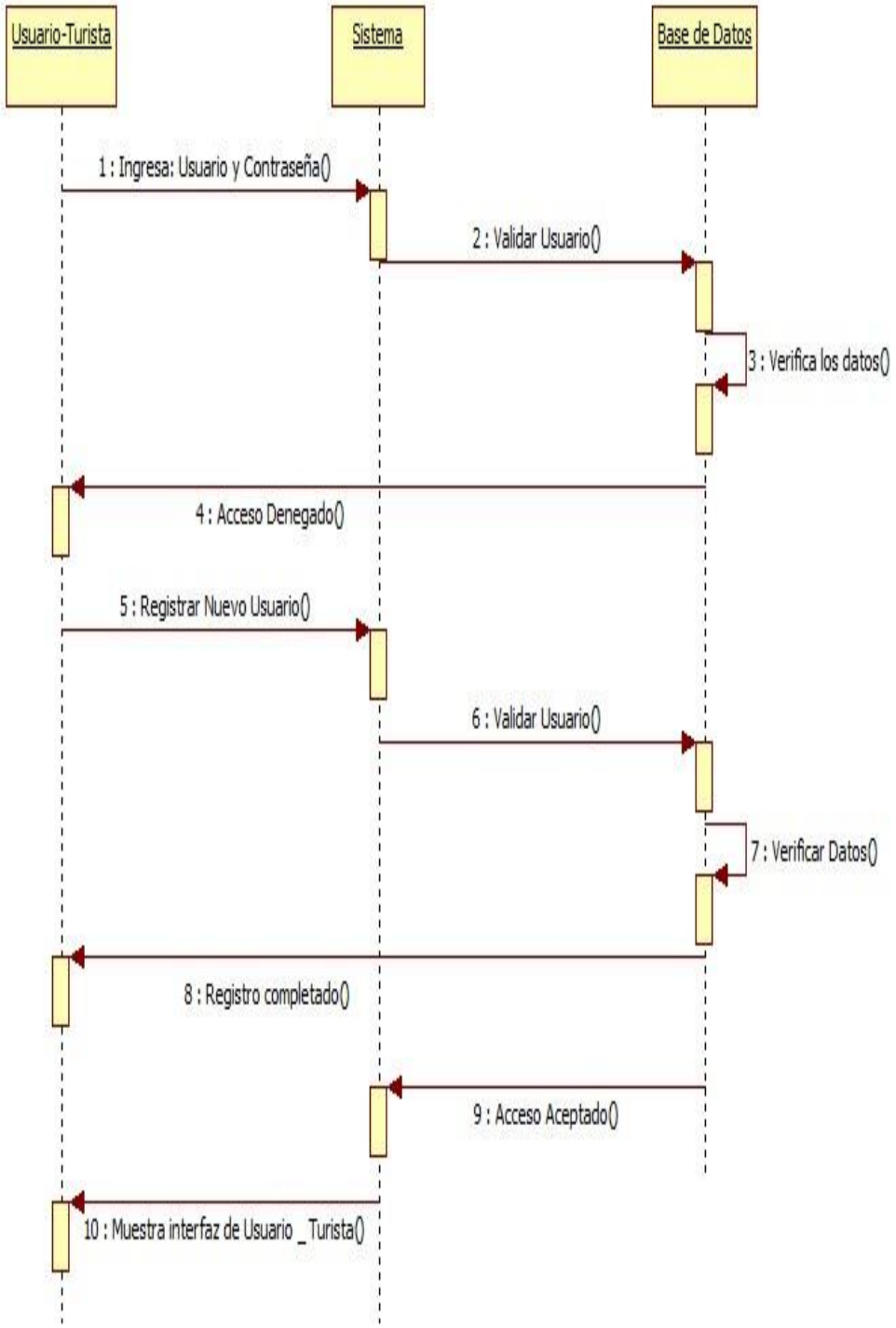


Fuente: Elaboración Propia

2.6. Diagramas de secuencia de casos de uso

2.6.1. Diagrama de Secuencia de Ingreso al Sistema

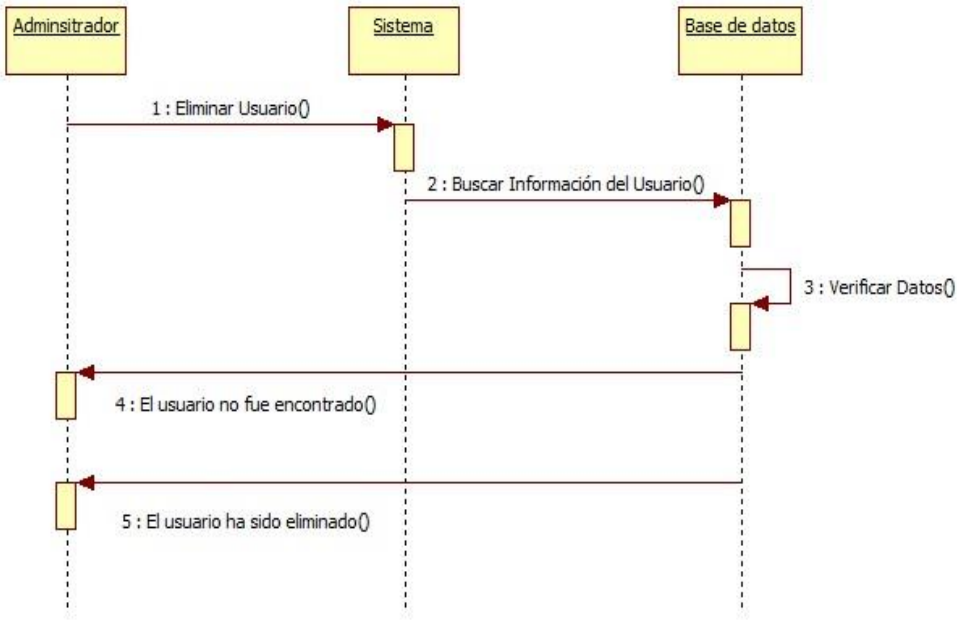
Grafico 48: Diagrama de secuencia de ingreso al sistema - Turista



Fuente: Elaboración Propia

2.6.2. Diagrama de Secuencia de Ingreso al Sistema

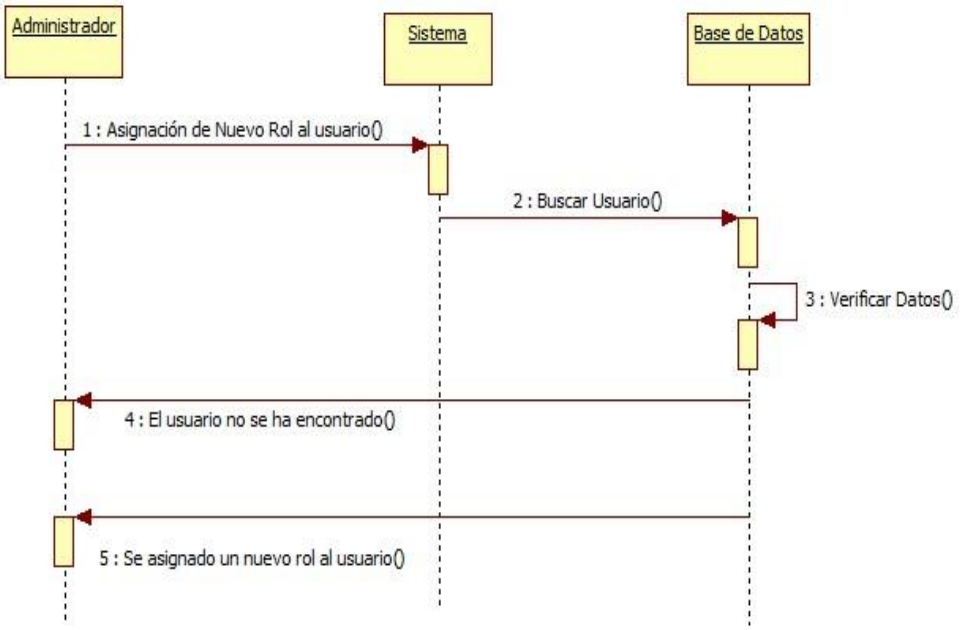
Grafico 49: Diagrama de secuencia de gestión de usuario



Fuente: Elaboración Propia

2.6.3. Diagrama de Secuencia de Asignación de Roles

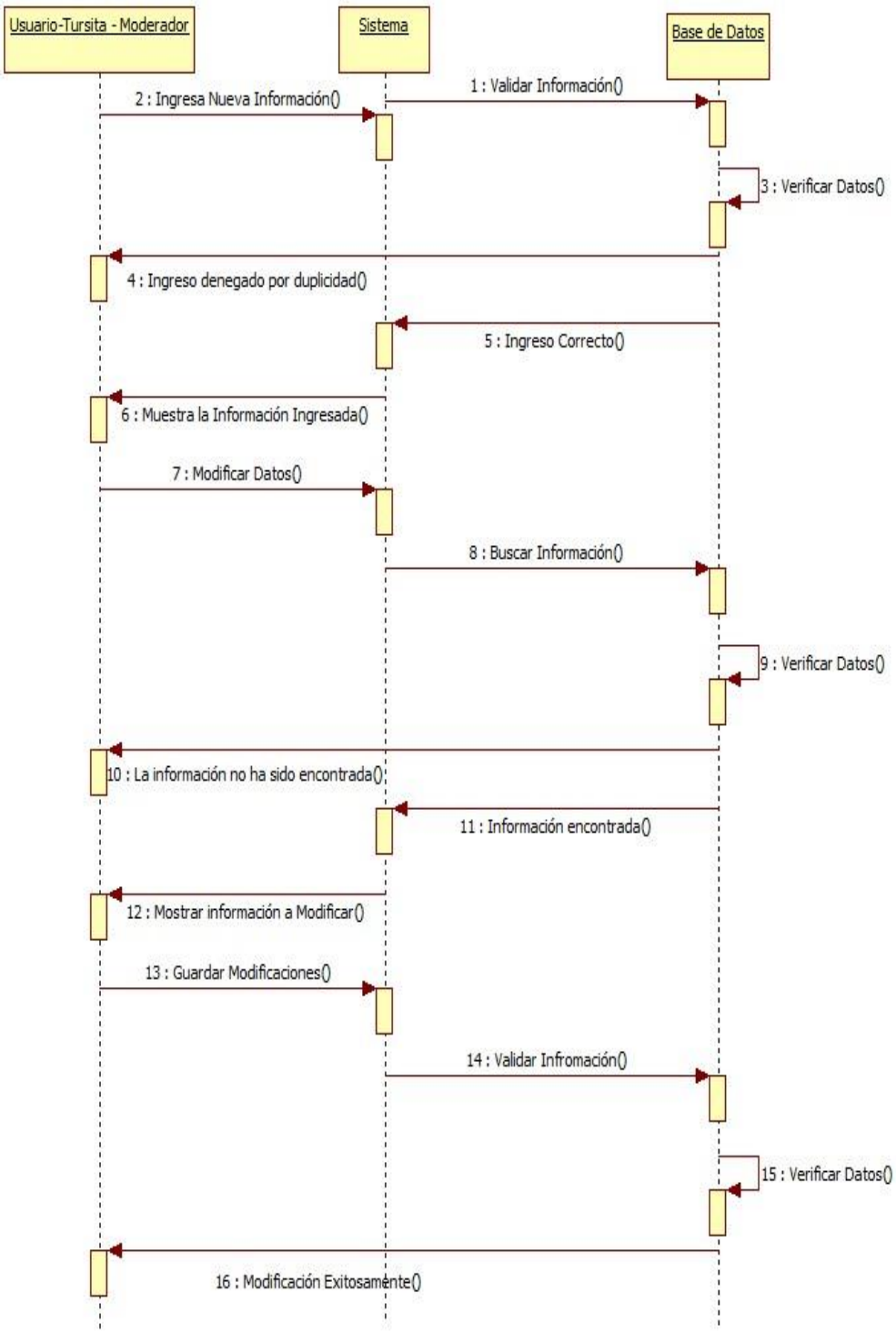
Grafico 50: Diagrama de secuencia de asignación de roles.



Fuente: Elaboración Propia

2.6.4. Diagrama de secuencia de gestión de información.

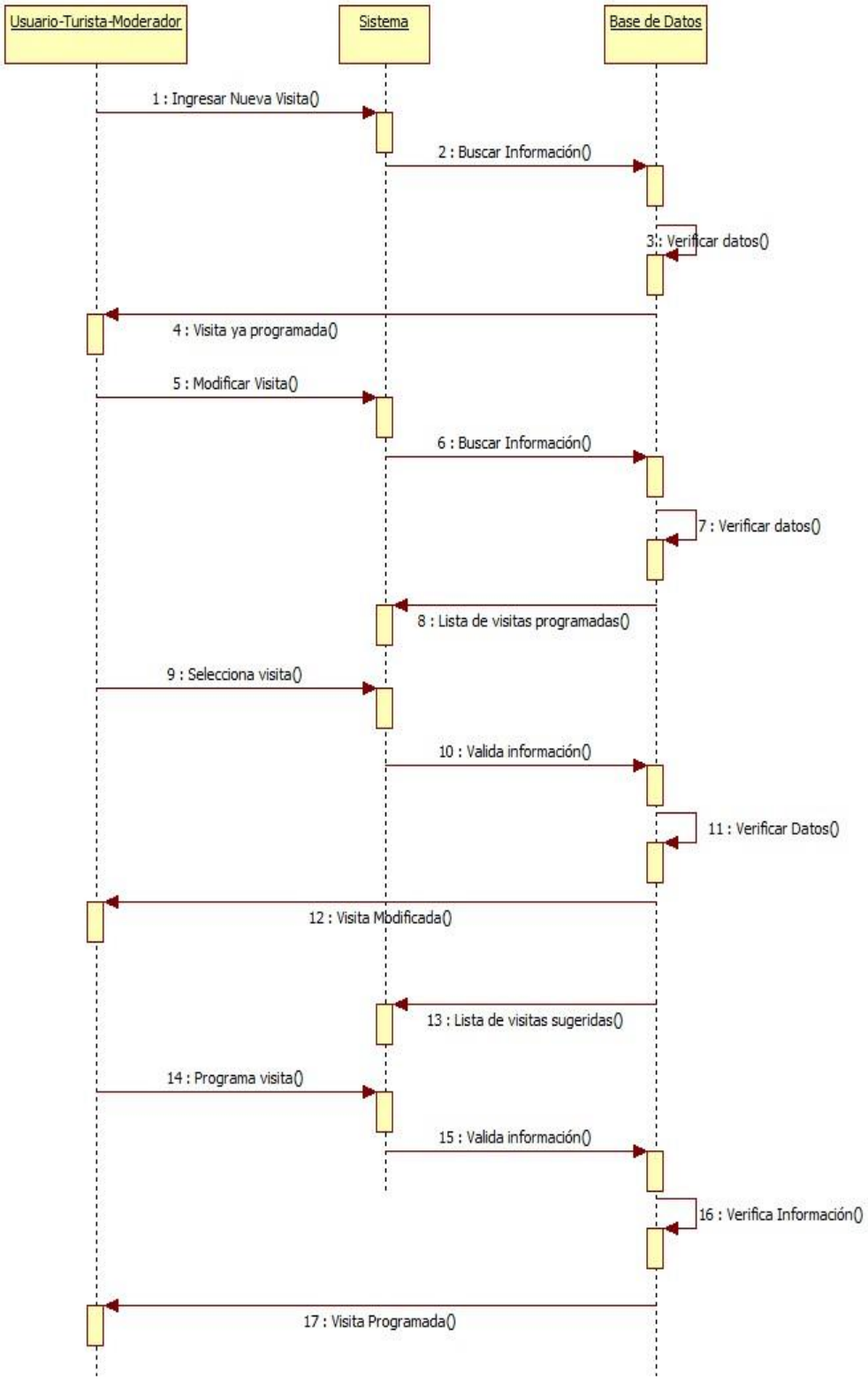
Grafico 51: Diagrama de secuencia de gestión de la información.



Fuente: Elaboración Propia

2.6.5. Diagrama de secuencia de gestión de visitas.

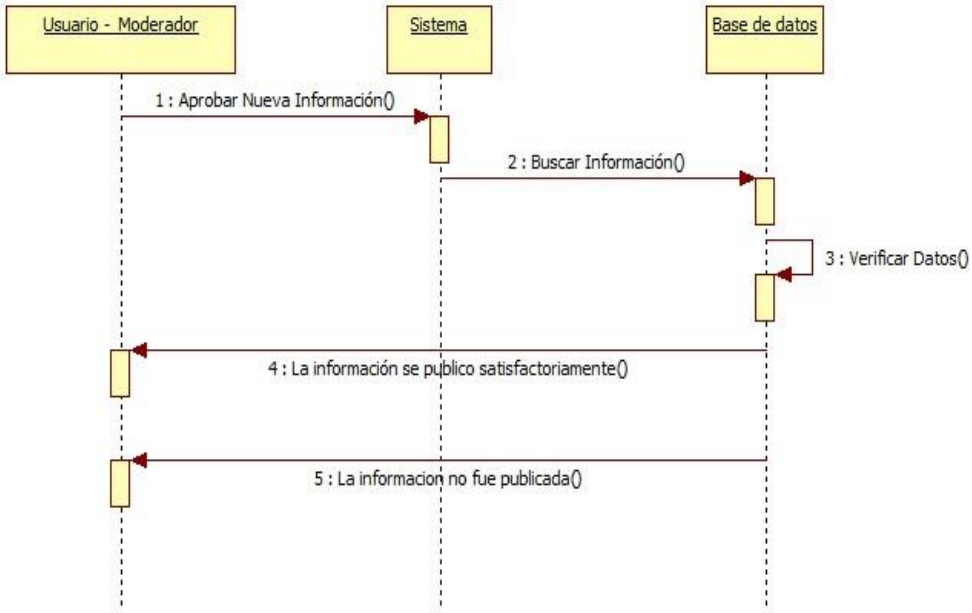
Grafico 52: Diagrama de secuencia de gestión de visitas



Fuente: Elaboración Propia

2.6.6. Diagrama de secuencia de aprobación y denegación de publicación de información.

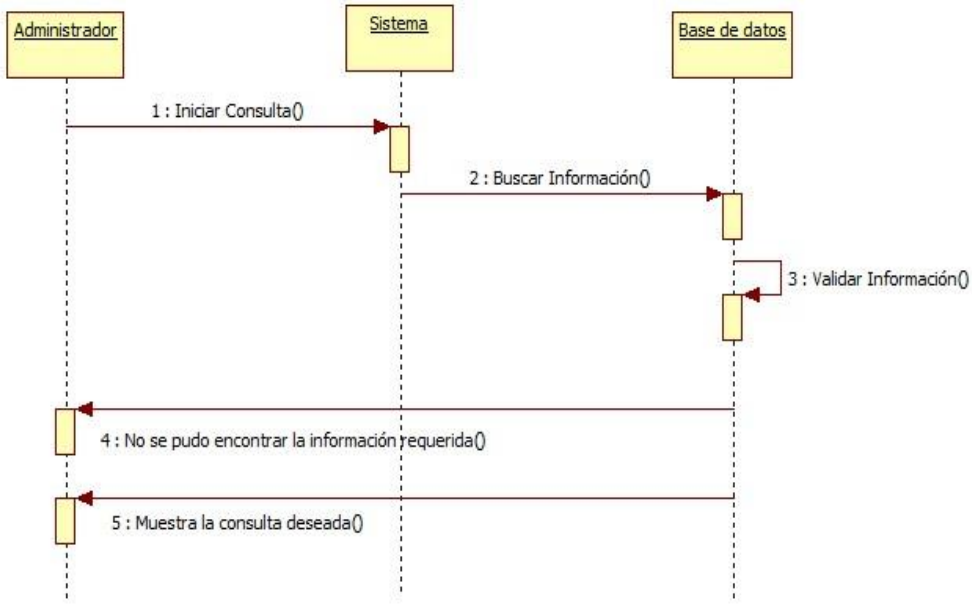
Grafico 53: Diagrama de secuencia-Aprobar-Desaprobar-Información.



Fuente: Elaboración Propia

2.6.7. Diagrama de secuencia de gestión de reportes.

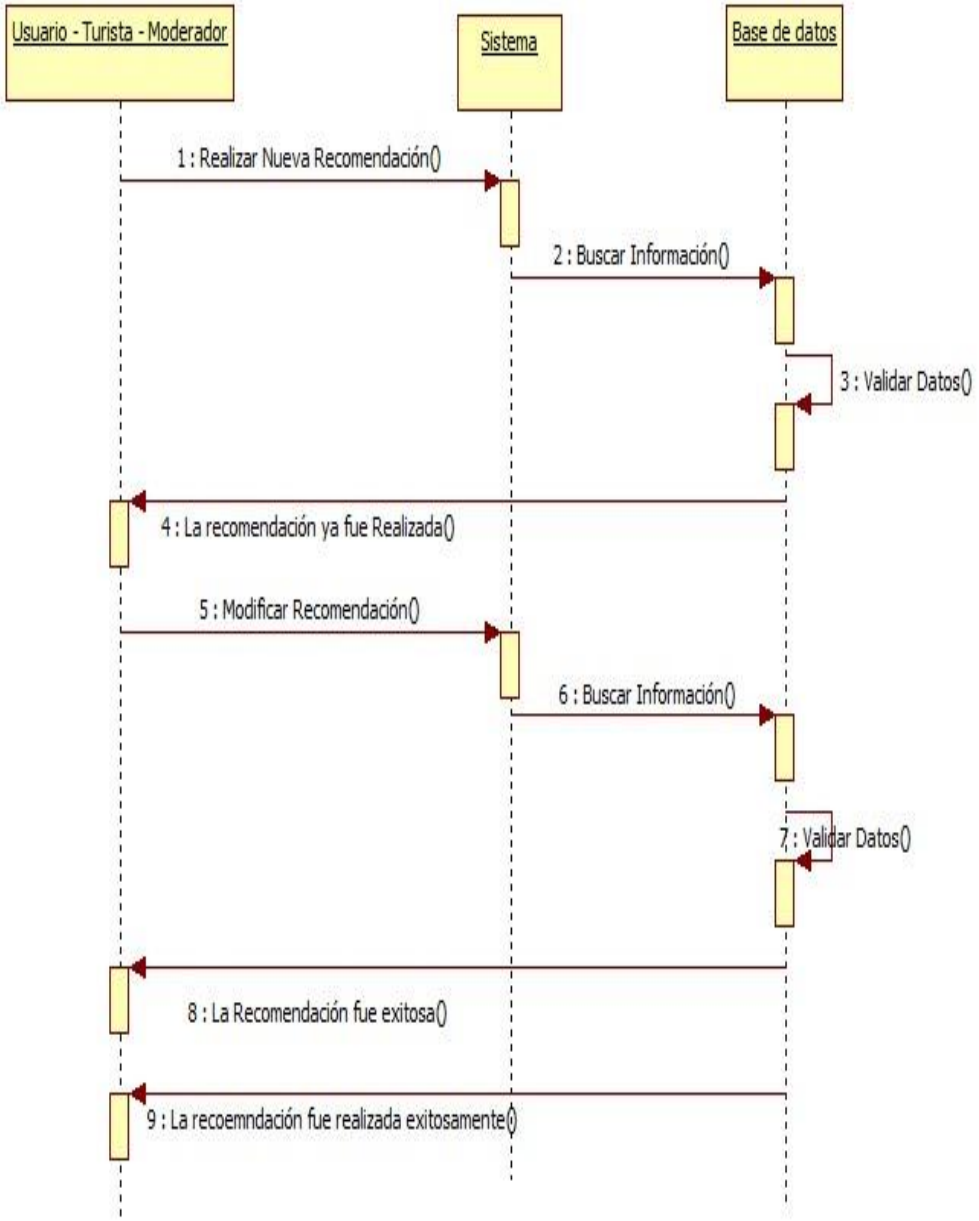
Grafico 54: Diagrama de secuencia de aprobar y desaprobar información.



Fuente: Elaboración Propia

2.6.8. Diagrama de secuencia de gestión de recomendaciones.

Grafico 55: Diagrama de secuencia de Aprobar-Desaprobar-Información.



Fuente: Elaboración Propia

2.7. Especificación de los casos de uso del sistema.

2.7.1. Especificación de caso de uso Inicio de sesión del Usuario-Turista.

Tabla 20: Especificación de caso de uso Inicio de sesión del Usuario-Turista.

Caso de uso	**01**
Nombre	Inicio de sesión-Usuario turista
Descripción	Se muestra en pantalla un formulario donde el usuario puede ingresar su usuario y contraseña si los tuviera o registrarse en el caso fuera necesario (si no posee usuario y contraseña).
Estado	Completo
Actores	Usuario Turista
Pre-Condición	- Si el usuario no posee cuenta de usuario y contraseña, deberá registrarse para poder obtener una. - Para registrarse como un usuario, antes deberá verificarse en la base de datos si el usuario no se ha registrado antes.
Flujo Principal	
Acción del Actor	Respuesta del software
1. Ingresar nombre de usuario	2. Verifica que le nombre de usuario se encuentre registrado.
3. Ingresa contraseña.	4. Verifica que la contraseña coincida con

	el usuario en la base de datos. 5. El sistema da acceso a la aplicación. 6. Caso de uso termina.
Excepción 1: Nombre de usuario incorrecto.	
7. Ingresar nombre de usuario	8. Muestra en pantalla un mensaje de error “Nombre de usuario no se encuentra registrado, intente nuevamente”. 9. Regresa al paso número uno.
Excepción 2: Contraseña incorrecta.	
10. Ingresar contraseña.	11. Muestra en pantalla un mensaje de error “Contraseña no coincide con el usuario, intente nuevamente”. 12. Regresa al paso número 10.
Excepción 3: Registrar nueva cuenta	
13. El usuario solicita el registro de una nueva cuenta.	14. La aplicación le muestra un formulario y le solicita que ingrese un usuario y una contraseña.
15. El usuario ingresa los datos requeridos.	16. El sistema verifica que todos los campos requeridos no se encuentren vacíos.

	17. El sistema verifica los datos en la base de datos. 18. Si todo esta correcto se almacena la información 19. El usuario deberá regresar al paso 1. 20. Si es incorrecto se le muestra un mensaje de error “el usuario ya se encuentra registrado” 21. El usuario deberá volver al paso 15 y volver a empezar. 22. Fin del flujo.
CU_ Relacionados	Ninguno
Post – condición	El sistema permite acceso a la aplicación y muestra todas sus bondades necesarias.

Fuente: Elaboración Propia

2.7.2. Especificación de caso de uso Inicio de sesión del Usuario-Moderador.

Tabla 21: Especificación de caso de uso Inicio de sesión del Usuario-Moderador.

Caso de uso	**02**
Nombre	Inicio de sesión-Usuario moderador
Descripción	Se muestra un formulario de inicio de sesión, en donde el moderador ingresa su usuario y contraseña y accede a la aplicación con mayores privilegios que el usuario turista

Estado	Completo
Actores	Usuario Moderador
Pre-Condición	- El turista debe poseer cuenta ya registrada como usuario-turista y que el administrador le haya asignado el rol de moderador.
Flujo Principal	
Acción del Actor	Respuesta del software
1. Ingresar nombre de usuario	2. Verifica que le nombre de usuario se encuentre registrado.
3. Ingresa contraseña.	4. Verifica que la contraseña coincida con el usuario en la base de datos. 5. El sistema da acceso a la aplicación. 6. Caso de uso termina.
Excepción 1: Nombre de usuario incorrecto.	
7. Ingresar nombre de usuario	8. Muestra en pantalla un mensaje de error “Nombre de usuario no se encuentra registrado, intente nuevamente”. 9. Regresa al paso número uno.
Excepción 2: Contraseña incorrecta.	
10. Ingresar contraseña.	11. Muestra en pantalla un mensaje de error “Contraseña no coincide

	con el usuario, intente nuevamente". 12. Regresa al paso número 10.
CU_ Relacionados	Ninguno
Post – condición	El sistema permite acceso a la aplicación y muestra todas sus bondades necesarias.

Fuente: Elaboración Propia

2.7.3. Especificación de caso de uso Inicio de sesión del Administrador

Tabla 22: Especificación de caso de uso Inicio de sesión del Administrador

Caso de uso	**03**
Nombre	Inicio de sesión-Usuario Administrador.
Descripción	Se muestra un formulario de inicio de sesión, en donde el administrador debe de ingresar el nombre de usuario y contraseña, este es el actor que posee todos los privilegios del sistema.
Estado	Completo
Actores	Administrador
Pre-Condición	- El Administrador debe de tener el nombre del usuario y contraseña para acceder al sistema.
Flujo Principal	
Acción del Actor	Respuesta del software

1. Ingresar nombre de usuario	2. Verifica que le nombre de usuario se encuentre registrado.
3. Ingresa contraseña.	4. Verifica que la contraseña coincida con el usuario en la base de datos. 5. El sistema da acceso a la aplicación. 6. Caso de uso termina.
Excepción 1: Nombre de usuario incorrecto.	
7. Ingresar nombre de usuario	8. Muestra en pantalla un mensaje de error "Nombre de usuario no se encuentra registrado, intente nuevamente". 9. Regresa al paso número uno.
Excepción 2: Contraseña incorrecta.	
10. Ingresar contraseña.	11. Muestra en pantalla un mensaje de error "Contraseña no coincide con el usuario, intente nuevamente". 12. Regresa al paso número 10.
CU_ Relacionados	Ninguno
Post – condición	El sistema permite acceso a la aplicación y muestra todas sus bondades necesarias.

Fuente: Elaboración Propia

2.7.4. Especificación de caso de uso Gestión de usuario.

Tabla 23: Especificación de caso de uso gestión de usuario

Caso de uso	**04**
Nombre	Gestión de usuario
Descripción	Esta opción es, la que se le atribuye al Administrador, este tiene la posibilidad de asignarle el rol de moderador al Usuario-Turista.
Estado	Completo
Actores	Administrador
Pre-Condición	- El usuario debe poseer cuenta ya registrada como usuario-turista para que el administrador pueda gestionarlos.
Flujo Principal	
Acción del Actor	Respuesta del software
1. Selecciona la opción de asignar nuevo rol al usuario-turista.	2. Muestra el formulario de asignación de roles. 3. Presenta un motor de búsqueda de Usuario-Turista
4. Busca la Usuario que desea asignarle un nuevo rol	5. Muestra la opción seleccionada con las opciones que se les fueron asignadas.
6. Selecciona la opción asignar rol de moderador.	7. Valida los datos, y muestra un aviso “El rol se ha asignado

	satisfactoriamente al usuario seleccionado”
Excepción 1: Eliminar usuario	
8. Selecciona la opción de eliminar usuario-turista.	9. Muestra el formulario de eliminación. 10. Presenta un motor de búsqueda de usuario.
11. Selecciona el usuario a eliminar.	12. Muestra la opción seleccionada con las opciones que se les fueron asignadas.
13. Elimina a usuario seleccionado.	14. Valida los datos, y muestra un aviso “El usuario ha sido eliminado satisfactoriamente” 15. Fin del flujo.
CU_ Relacionados	CU-03 Inicio de sesión Administrador.
Post – condición	El sistema permite la asignación de roles y la eliminación de Usuarios – Turistas y/o Usuarios-Moderadores.

Fuente: Elaboración Propia

2.7.5. Especificación de caso de uso Gestión de la información

Tabla 24: Especificación de caso de uso gestión de la información.

Caso de uso	**05**
Nombre	Gestión de la información
Descripción	El usuario Turista-Moderador-Administrador, comparten información en el sitio web, información que será almacenada en la base de datos de la aplicación.
Estado	Completo
Actores	Usuario-Turista, Usuario-Moderador, Administrador.
Pre-Condición	- Todos los usuarios deben de haber accedido con su cuenta de usuario a la aplicación.
Flujo Principal	
Acción del Actor	Respuesta del software
1. Ingresa al módulo que permite el ingreso y edición de la información compartida	2. Muestra el módulo de ingreso y edición de la información compartida 3. El usuario puede publicar información. 4. El usuario puedo editar la información publicada.
5. Ingresa la nueva información al sistema.	6. Verifica que los campos requeridos no se encuentren vacíos.

	7. Muestra aviso “Desea guardar la información ingresada”
8. Pulsa la opción guardar para almacenar los datos ingresados.	9. Guarda la información ingresada en la base de datos.
Excepción 1: Editar información	
10. Ingresa a modificar información	11. Muestra el formulario de edición de información compartida. 12. Muestra motor de búsqueda de información requerida para editar.
13. Selecciona la información que desea editar.	14. Muestra la información y las opciones de edición que son necesarias para efectuar la edición.
15. Realiza las modificaciones necesarias.	16. Valida la información con la base de datos. 17. Muestra un aviso “la edición se realizó satisfactoriamente” 18. Fin del flujo.
CU_ Relacionados	CU_01 Inicio de sesión del Usuario turista. CU_02 Inicio de sesión del Usuario moderador
Post – condición	El sistema permite el ingreso y modificación de información

	sobre la diversidad cultural y turística de la aplicación web.

Fuente: Elaboración Propia

2.7.6. Especificación de caso de uso gestión de visitas

Tabla 25: Especificación de caso de uso gestión de visitas.

Caso de uso	**06**
Nombre	Gestión de visitas
Descripción	El usuario Turista-Moderador, programan sus visitas en las fechas q estimen por conveniente.
Estado	Completo
Actores	Usuario-Turista, Usuario-Moderador.
Pre-Condición	- El usuario Turista y usuario moderador deben de haber antes ingresado a la aplicación web.
Flujo Principal	
Acción del Actor	Respuesta del software
1. Ingresa al módulo de gestión de visitas.	2. Muestra formulario de gestión de visitas.
3. Busca sitio o evento al cual quiere visitar o asistir respectivamente	4. Muestra la información disponible.
5. Programa una visita.	6. Valida los datos en la base de datos,

	verificando que los campos requeridos no estén en blanco. 7. Muestra un aviso “la visita se guardó satisfactoriamente”. 8. Fin del flujo.
CU_ Relacionados	CU_01 Inicio de sesión del usuario-turista. CU_02 Inicio de sesión del usuario-moderador.
Post – condición	El sistema muestra todas las vistas programadas.

Fuente: Elaboración Propia

2.7.7. Especificación de caso de uso gestión de recomendaciones.

Tabla 26: Especificación de caso de uso gestión de recomendaciones.

Caso de uso	**07**
Nombre	Gestión de recomendaciones
Descripción	El usuario Turista o Moderador realizan sus sugerencias a través de puntuaciones según sus experiencias vividas.
Estado	Completo
Actores	Usuario-Turista, Usuario-Moderador.
Pre-Condición	- Todos los usuarios deben de haber accedido con su cuenta de usuario a la aplicación.

Flujo Principal	
Acción del Actor	Respuesta del software
1. Ingresa al módulo de recomendaciones	2. Muestra los formularios requeridos por el usuario. 3. Muestra motor de búsqueda de información.
4. Selecciona la información que desea recomendar.	5. Muestra la información requerida por las opciones necesarias.
6. Realiza la votación a través de puntuaciones.	7. Valida los datos. 8. Verifica que los campos requeridos no se encuentren vacíos. 9. Muestra un aviso “la puntuación se realizó satisfactoriamente.” 10. Fin del flujo.
CU_ Relacionados	CU_01 Inicio de sesión del Usuario-turista. CU_02 Inicio de sesión del Usuario moderador.
Post – condición	El sistema muestra sugerencias de diversidad turística y cultural, según sea su semejanza con otro usuario.

Fuente: Elaboración Propia

2.7.8. Especificación de caso de uso aprobar o desaprobar la información compartida por el usuario-turista.

Tabla 27: Especificación de caso de uso aprobar o desaprobar la información compartida por el usuario turista.

Caso de uso	**08**
Nombre	Aprobar y desaprobar la información compartida por el usuario-turista..
Descripción	El Moderador y/o el administrador son los únicos responsables de aprobar o desaprobar la información compartida.
Estado	Completo
Actores	Usuario-Moderador, Administrador.
Pre-Condición	- Todos los usuarios deben de haber accedido con su cuenta de usuario a la aplicación.
Flujo Principal	
Acción del Actor	Respuesta del software
1. Ingresa a la sección de aprobación y denegación de publicaciones compartidas por el usuario-turista	2. Muestra el formulario de la sección de aprobación y denegación.

3. Busca las publicaciones recientes	4. Muestra la información requerida por el usuario.
5. Realiza la aprobación o desaprobación de la información compartida por el usuario.	6. Valida los datos con la base de datos. 7. Si es Aprobado, envía un aviso "la información fue publicada satisfactoriamente" 8. Si es desaprobado, se muestra un aviso "la información se ha denegado" 9. Actualiza la lista. 10. Fin del flujo
CU_ Relacionados	CU_02 Inicio de sesión del usuario moderador CU_03 Inicio de sesión del Administrador.
Post – condición	El sistema muestra las publicaciones aprobadas por el Usuario-Moderador o el Administrador.

Fuente: Elaboración Propia

2.7.9. Especificación de caso de uso gestión de reportes

Tabla 28: Especificación de caso de uso gestión de reportes

Caso de uso	**09**
Nombre	Gestión de reportes
Descripción	Se muestran los reportes solicitados por el administrador de la aplicación web.

Estado	Completo
Actores	Administrador.
Pre-Condición	- El Administrador debe haber iniciado sesión.
Flujo Principal	
Acción del Actor	Respuesta del software
1. Ingresa a l módulo de gestión de reportes	2. Muestra formulario del módulo de gestión de reportes
3. Consulta información	4. Muestra la información solicitada por el usuario 5. Fin del flujo
CU_ Relacionados	CU_03 Inicio de sesión del Administrador
Post – condición	El sistema muestra los reportes solicitados por el Administrador

Fuente: Elaboración Propia

ANEXOS 3

CONSTRUCCIÓN DE LA APLICACIÓN

1. Controlador IndexController

En esta sección se encuentra todo el proceso que se muestra en el interfaz principal.

```
1. <?php
2.
3. class IndexController extends Zend_Controller_Action
4. {
5.
6.     public function init()
7.     {
8.         $this->view->baseUrl=$this->getRequest()-
   >getBaseUrl();
9.         //db-table
10.           $this->db_usuariocalificacion=new
   Application_Model_DbTable_Usuariocalificacion();
11.           $this->db_usuario=new
   Application_Model_DbTable_Usuario();
12.           $this->db_bdturismo=new
   Application_Model_DbTable_Bdtturismo();
13.           $this->db_turismoimagenes=new
   Application_Model_DbTable_Turismoimagenes();
14.           $this->db_catturismo=new
   Application_Model_DbTable_Catturismo();
15.           $this->session = new
   Zend_Session_Namespace('usuario');
16.               $this->db_usuariodato=new
   Application_Model_DbTable_Usuariodato();
17.
18.           //modelos
19.           $this->usuario=new Application_Model_Usuario();
20.           $this->usuarioturismo=new
   Application_Model_Usuariocalificacion();
21.           if(isset($this->session->cod_usuario)){
22.               $this->view->codUsuario=$this->session-
   >cod_usuario;
23.               $this->view->nombreUser=$this->session-
   >usu_nombre;
24.           }
25.       }
26.
27.       public function indexAction()
28.       {
29.           //desabilitando layout
30.           $this->_helper->layout()->disableLayout();
31.
32.           $turismo=$this->db_bdturismo->TodoTurismo(50);
33.           $categorias=$this->db_catturismo->all();
34.           $i=0;
35.
36.           foreach($turismo as $row){
37.               $groupImg=$this->db_turismoimagenes-
   >CodTurismo($row['cod_turismo']);
38.               foreach($groupImg as $row2){
39.                   $imagen=$row2['tuimag_ruta'];
40.               }
41.               $turismo[$i]['img']=$imagen;
42.               $turismo[$i]['titulo2']=str_replace(' ', '-',
   $this-
```

```
>_sanear_string(strtolower(utf8_decode($row['turi_nombre'])))
);
                $i++;
            }
            $this->view->turismo=$turismo;
            $this->view->categoria=$categorias;

        }
        //funcion que crea un lista
array[informacionturistica]=calificacion
        public function _crearlista($CalificacionUser)
        {
            $Array=array();
            $i=0;
            foreach($CalificacionUser as $row2){

$Array[$row2['cod_turismo']]=$row2['turi_calificacion'];
            }
             return $Array;
        }
        //funcion array almacena informacion turistica que
califico el usuario
        public function _turismoUser($User)
        {
            $Array=array();
            $i=0;
            foreach($User as $row2){
                  $Array[$i]=$row2['cod_turismo'];
                  $i++;
            }
             return $Array;
        }
        //funcion array devuelve calificaciones de información
turistica en comun de user y user nuevo
        public function
_turismoIntersect($Data,$ArrayIntersect)
        {
            $Array=array();
            $i=0;
            foreach($ArrayIntersect as $Turismo){
                foreach($Data as $indice1 => $fil1){
                    if($Turismo==$indice1){
                        $Array[$i]=$fil1;
                        $i++;
                    }
                }
            }
             return $Array;
        }
        //funcion array almacena informacion turistica que
califico el usuario
        public function _pearson($x,$y)
        {
            $length= count($x);
            $mean1=array_sum($x) / $length;
            $mean2=array_sum($y) / $length;

            $a=0;
            $b=0;
            $axb=0;
            $a2=0;
```

```
96.           $b2=0;
97.
98.           for($i=0;$i<$length;$i++)
99.           {
100.              $a=$x[$i]-$mean1;
101.              $b=$y[$i]-$mean2;
102.              $axb=$axb+($a*$b);
103.              $a2=$a2+ pow($a,2);
104.              $b2=$b2+ pow($b,2);
105.          }
106.          $corr= $axb / sqrt($a2*$b2);
107.          return $corr;
108.      }
109.      public function _algoritmoAction()
110.      {
111.            // action body
112.          //traemos todos los usuarios del sistema menos el
   usuario en session
113.          $AllUser=$this->db_usuario->TodoUsuario($this-
   >session->cod_usuario);
114.          //calificacion del usuario nuebo
115.
116.          $Nuevo=$this->db_usuariocalificacion-
   >CalificacionUsuario($this->session->cod_usuario);
117.          //si exiten calificacion del nuevo usuario
   procedemos con el algoritmo
118.          if(count($Nuevo)>0){
119.          $Datos1=$this->_crearlista($Nuevo);
120.          $TurismoUserNuevo=$this->_turismoUser($Nuevo);
121.
122.
123.          $NuevoAllUser=array();
124.          $arrayTurismo=array();
125.          $i=0;
126.          foreach($AllUser as $row){
127.              $CalificacionUser=$this-
   >db_usuariocalificacion-
   >CalificacionUsuario($row['cod_usuario']);
128.              $Datos=$this->_crearlista($CalificacionUser);
129.              $turismoUser =$this-
   >_turismoUser($CalificacionUser);
130.             //intersecando que exista ciudades en comun
   entre las dos personas
131.               $result = array_intersect($turismoUser,
   $TurismoUserNuevo);
132.
133.              // si existe alguna ciudad en comun
134.              if(count($result)>0){
135.                  $Valor=0;
136.                  //datos de la persona que entra en bucle
137.                  //indice1 ciudad que voto fil1 el valor
138.                  //datos1 usuario nuevo
139.                  //indice2 ciudad que voto fil2 el valor
140.
141.                  //DISTANCIA EUCLIDEANA
142.                  foreach ($Datos as $indice1 => $fil1){
143.                      foreach ($Datos1 as $indice2 => $fil2){
144.                          if($indice1==$indice2){
145.                              $Valor=$Valor + pow(($fil2-
   $fil1), 2);
146.                          }
```

```
147.                     }
148.                     //arrayturismo guarda todas las
   ciudades calificadas;
149.                     $arrayTurismo[$i]=$indice1;
150.                     $i++;
151.                 }
152.                 $Distancia = (1/(1+sqrt($Valor)));
153.
154.                 //ALGORITMO DE PEARSON
155.                 $InterUser=$this-
   >_turismoIntersect($Datos,$result);
156.                 $InterUserNuevo=$this-
   >_turismoIntersect($Datos1,$result);
157.                  $Distancia=$this-
   >_pearson($InterUser,$InterUserNuevo);
158.  //           echo '<pre>';
159.  //           print_r($Datos);
160.  //           print_r($turismoUser);
161.  //           print_r($TurismoUserNuevo);
162.  //           print_r($result);
163.  //           print_r($InterUser);
164.  //           print_r($InterUserNuevo);
165.  //           print_r($Distancia);
166.  //           echo '<pre>';exit;
167.
168.                     array_push($NuevoAllUser,
169.                                         array(
170.                                             "Usuario"=>
   $row['cod_usuario'],
171.                                             "Ciudad" => $Datos,
172.                                            // "Distanciae" =>
   1/(1+sqrt($Valor)),
173.                                             "Distanciae" =>
   $Distancia,
174.                                         ));
175.             }
176.          }
177.
178.
179.          //ciudad unica elimina redundancia
180.          //$CiudadNuevoUser ciudades calificadas por el
   nuevo usuario
181.          $CiudadUnica=array_unique($arrayTurismo);
182.          foreach($CiudadUnica as $indice => $row){
183.                foreach($TurismoUserNuevo as $row2){
184.                    if($row==$row2){
185.                     unset ($CiudadUnica[$indice]);
186.                    }
187.                }
188.          }
189.          //ciudadunica solo contiene ciudades que no voto
   el nuevo usuario
190.          //comparando ciudaddes
191.          //$NuevoAllUser contiene las distancias de cada
   usuario con respecto al nuevo usuario
192.          $Recomendacion=array();
193.          foreach($CiudadUnica as $row){
194.              $suma=0;
195.              $SumDE=0;
196.             foreach($NuevoAllUser as $row2){
```

```
197.                 foreach($row2['Ciudad'] as $indice =>
   $valor){
198.                 //usuario [i] califico el sitio $row
199.                     if($indice == $row){
200.                         $Multi=$row2['Distanciae']*$valor;
201.                         $suma=$suma+$Multi;
202.                         $SumDE=$SumDE +
   $row2['Distanciae'];
203.                     }
204.                 }
205.
206.             }
207.             $Recomendacion[$row]=$suma/$SumDE;
208.         }
209.         return $Recomendacion;
210. //           echo '<pre>';
211. //           print_r($CiudadUnica);
212. //           print_r($NuevoAllUser);
213. //
214. //           print_r($DistanciaE);
215. //          print_r($Recomendacion);
216. //           echo '<pre>';exit;
217.         }
218.
219.     }
220.
221.     public function loginAction()
222.     {
223.         $this->_helper->layout()->disableLayout();
224.           //recogemos la información de los inputs
225.           $Correo=$this->getRequest()-
   >getParam("txtAlias");
226.             $Contra=$this->getRequest()-
   >getParam("txtPassword");
227.             $Alert=$this->getRequest()->getParam("alert");
228.             if(isset($Correo) and isset($Contra)){
229.             $authAdapter = new Zend_Auth_Adapter_DbTable();
230.             $authAdapter
231.                 ->setTableName('usuario')
232.                 ->setIdentityColumn('usu_nombre')
233.                 ->setCredentialColumn('usu_contrasena');
234.
235.             $authAdapter
236.                 ->setIdentity($Correo)
237.                 ->setCredential($Contra);
238.
239.             $auth = Zend_Auth::getInstance();
240.
241.             $result = $auth->authenticate($authAdapter);
242.
243.             if(  $result->isValid() ){
244.                 $datos=$authAdapter->getResultRowObject();
245.                 $session = new
   Zend_Session_Namespace('usuario');
246.                 $session->cod_usuario=$datos->cod_usuario;
247.                 $session->usu_nombre=$datos->usu_nombre;
248.                 $Personal=$this->db_usuariodato-
   >UsuarioDatos($datos->cod_usuario);
249.                 $session->foto=$Personal[0]['usua_imagen'];
250.                 $rol=$this->db_usuario->UsuarioRol($datos-
   >cod_usuario);
```

```
                    $session->rol=$rol[0]['rol_nombre'];
                     $this->_redirect('index');
                }else{
                    $this->_redirect('index/login/alert/error');
                }
                }
                if($Alert=="error"){
                    $this->view->alert="El usuario o la Contraseña son incorrectos";
                    $this->view->classalert="alert-danger";
                }
                if($Alert=="reg"){
                    $this->view->alert="El registro se completo correctamente, ahora puede ingresar";
                    $this->view->classalert="alert-success";
                }
        }

        public function registroAction()
        {
            $txtEmail=$this->getRequest()->getParam("txtEmail");
            $txtLogin=$this->getRequest()->getParam("txtLogin");
            $txtPass=$this->getRequest()->getParam("txtPass");
            if(isset($txtEmail)){
                $this->usuario->registrar(0,$txtLogin,$txtEmail,$txtPass);
                $this->_redirect('index/login/alert/reg');
            }
        }
        public function ratingAction()
        {
            $CodTurismo=$this->getRequest()->getParam("codt");
            $CodCalificacion=$this->getRequest()->getParam("codc");
            $TuriCalificacion=$this->getRequest()->getParam("val");
            $url=$this->getRequest()->getParam("url");
            if(isset($CodTurismo)){
                $this->usuarioturismo->guardar($CodCalificacion,$this->session->cod_usuario,$CodTurismo,$TuriCalificacion);
                $this->_redirect('ver/'.$url);
            }
        }
        public function _sanear_string($string)
        {

         $string = utf8_encode(trim($string));

        $string = str_replace(
            array('á', 'à', 'ä', 'â', 'ª', 'Á', 'À', 'Â', 'Ä'),
            array('a', 'a', 'a', 'a', 'a', 'A', 'A', 'A', 'A'),
            $string
        );

        $string = str_replace(
            array('é', 'è', 'ë', 'ê', 'É', 'È', 'Ê', 'Ë'),
            array('e', 'e', 'e', 'e', 'E', 'E', 'E', 'E'),
            $string
```

```
303.      );
304.
305.      $string = str_replace(
306.          array('í', 'ì', 'ï', 'î', 'Í', 'Ì', 'Ï', 'Î'),
307.          array('i', 'i', 'i', 'i', 'I', 'I', 'I', 'I'),
308.          $string
309.      );
310.
311.      $string = str_replace(
312.          array('ó', 'ò', 'ö', 'ô', 'Ó', 'Ò', 'Ö', 'Ô'),
313.          array('o', 'o', 'o', 'o', 'O', 'O', 'O', 'O'),
314.          $string
315.      );
316.
317.      $string = str_replace(
318.          array('ú', 'ù', 'ü', 'û', 'Ú', 'Ù', 'Û', 'Ü'),
319.          array('u', 'u', 'u', 'u', 'U', 'U', 'U', 'U'),
320.          $string
321.      );
322.
323.      $string = str_replace(
324.          array('ñ', 'Ñ', 'ç', 'Ç'),
325.          array('n', 'N', 'c', 'C',),
326.          $string
327.      );
328.
329.      //Esta parte se encarga de eliminar cualquier caracter
    extraño
330.      $string = str_replace(
331.          array( "¨", "°", "-", "~",
332.                "#", "@", "|", "!", "\"",
333.                "·", "$", "%", "&", "/",
334.                "(", ")", "?", "'", "¡",
335.                "¿", "[", "^", "`", "]",
336.                "+", "}", "{", "¨", "´",
337.                ">", "< ", ";", ",", ":",
338.                "."),
339.          '',
340.          $string
341.      );
342.
343.
344.      return $string;
345.      }
346.
347.
348.
349.  }
```

2. Controlador AdminController

En esta sección se encuentra todo el proceso de mantenimiento para el administrador, moderador y usuario.

```
1.    <?php
2.    /**
3.     * clase controlador que contiene todo el proceso
      mantenimiento para el
4.     * administrador, moderador y usuario.
5.     */
```

```
6.
7.
8.
9.      class AdminController extends Zend_Controller_Action
10.     {
11.
12.         /**
13.          * inicializando variables objeto globales
14.          */
15.         public function init()
16.         {
17.
18.             //codigo base
19.             $this->view->baseUrl=$this->getRequest()-
        >getBaseUrl();
20.             $this->_helper->layout()->disableLayout();
21.             //tb-table
22.             $this->db_catactividad=new
        Application_Model_DbTable_Catactividad();
23.             $this->db_catturismo=new
        Application_Model_DbTable_Catturismo();
24.             $this->db_bdtactividades=new
        Application_Model_DbTable_Bdtactividades();
25.             $this->db_bdturismo=new
        Application_Model_DbTable_Bdtturismo();
26.             $this->db_departamento=new
        Application_Model_DbTable_Departamento();
27.             $this->db_provincia=new
        Application_Model_DbTable_Provincia();
28.             $this->db_distrito=new
        Application_Model_DbTable_Distrito();
29.             $this->db_actividadimagenes=new
        Application_Model_DbTable_Actividadimagenes();
30.             $this->db_turismoimagenes=new
        Application_Model_DbTable_Turismoimagenes();
31.             $this->db_platostipicos=new
        Application_Model_DbTable_Platostipicos();
32.             $this->db_platosimagenes=new
        Application_Model_DbTable_Platosimagen();
33.             $this->db_fauna=new
        Application_Model_DbTable_Fauna();
34.             $this->db_flora=new
        Application_Model_DbTable_Flora();
35.             $this->db_faunaimagenes=new
        Application_Model_DbTable_Faunaimagenes();
36.             $this->db_floraimagenes=new
        Application_Model_DbTable_Floraimagenes();
37.             $this->db_usuario=new
        Application_Model_DbTable_Usuario();
38.             $this->db_usuariodato=new
        Application_Model_DbTable_Usuariodato();
39.             $this->db_inscripcionactividad=new
        Application_Model_DbTable_Inscactividad();
40.             $this->db_usuarioturismo=new
        Application_Model_DbTable_Usuarioturismo();
41.             $this->db_turimopermiso=new
        Application_Model_DbTable_Turismopermiso();
42.             $this->db_usuariocalificacion=new
        Application_Model_DbTable_Usuariocalificacion();
43.             //modelos
```

```
44.           $this->dbtactividades=new
      Application_Model_Dbtactividades();
45.           $this->dbtturismo=new
      Application_Model_Dbtturismo();
46.           $this->actividadimagenes=new
      Application_Model_Actividadimagenes();
47.           $this->turismoimagenes=new
      Application_Model_Turismoimagenes();
48.           $this->platostipicos=new
      Application_Model_Platostipicos();
49.           $this->platosimagenes=new
      Application_Model_Platosimagen();
50.           $this->fauna=new Application_Model_Fauna();
51.           $this->flora=new Application_Model_Flora();
52.           $this->faunaimagenes=new
      Application_Model_Faunaimagenes();
53.           $this->floraimagenes=new
      Application_Model_Floraimagenes();
54.           $this->usuario=new Application_Model_Usuario();
55.           $this->usuarioDato=new
      Application_Model_Usuariodato();
56.           $this->turismopermiso=new
      Application_Model_Turismopermiso();
57.
58.           //sessiones
59.           $this->session2 = new
      Zend_Session_Namespace('imagen');
60.           //si no existe session usuario la creamos
61.
62.            $this->session = new
      Zend_Session_Namespace('usuario');
63.            if(isset($this->session->cod_usuario)){
64.                $this->view->codUsuario=$this->session-
      >cod_usuario;
65.                $this->view->nombreUser=$this->session-
      >usu_nombre;
66.                $this->view->foto=$this->session->foto;
67.           }
68.
69.           //si el rol de la persona autentificada es
      superadmin
70.           if($this->session->rol=='superadmin'){
71.               $this->view->rol='superadmin';
72.           }
73.           //si el rol de la persona autentificada es admin
74.           if($this->session->rol=='moderador'){
75.               $this->view->rol='moderador';
76.           }
77.           //si el rol de la persona autentificada es usuario
78.           if($this->session->rol=='usuario'){
79.               $this->view->rol='usuario';
80.           }
81.
82.       }
83.
84.       /**
85.        * acción principal del controlador que muestra la
      interfaz
86.        * de autentificación al sistema
87.        */
88.       public function indexAction()
```

```
89.       {
90.           // action body
91.       }
92.
93.       /**
94.        * acción para autentificarse en el sistema
95.        */
96.       public function logueoAction()
97.       {
98.             //recogemos la información de los inputs
99.             $Correo=$this->getRequest()-
      >getParam("txtAlias");
100.              $Contra=$this->getRequest()-
      >getParam("txtPassword");
101.              if(isset($Correo) and isset($Contra)){
102.              $authAdapter = new Zend_Auth_Adapter_DbTable();
103.              $authAdapter
104.                  ->setTableName('usuario')
105.                  ->setIdentityColumn('usu_nombre')
106.                  ->setCredentialColumn('usu_contrasena');
107.
108.              $authAdapter
109.                  ->setIdentity($Correo)
110.                  ->setCredential($Contra);
111.
112.              $auth = Zend_Auth::getInstance();
113.
114.              $result = $auth->authenticate($authAdapter);
115.
116.              if(  $result->isValid() ){
117.                  $datos=$authAdapter->getResultRowObject();
118.                  $session = new
      Zend_Session_Namespace('usuario');
119.                  $session->cod_usuario=$datos->cod_usuario;
120.                  $Personal=$this->db_usuariodato-
      >UsuarioDatos($datos->cod_usuario);
121.
122.                  $session->foto=$Personal[0]['usua_imagen'];
123.                  $rol=$this->db_usuario->UsuarioRol($datos-
      >cod_usuario);
124.                  $session->rol=$rol[0]['rol_nombre'];
125.                   $this->_redirect('admin/portada');
126.              }else{
127.                  $this->_redirect('admin/index');
128.              }
129.              }else{
130.                  $this->_redirect('admin/index');
131.              }
132.      }
133.
134.      /**
135.       * acción para salir del sistema
136.       */
137.      public function logoutAction()
138.      {
139.          unset($this->session->cod_usuario);
140.          $this->_redirect('index');
141.      }
142.
143.      /**
144.       * IACTIVIDAD
```

```
145.       * acción para ingresar una nueva actividad
146.       */
147.      public function actividadAction()
148.      {
149.           //existe usuario logueado de lo contrario error
150.           if(isset($this->session->cod_usuario)){
151.               $Titulo=$this->getRequest()-
    >getParam("txttitulo");
152.               //enviando categorias de actividad a la vista
153.               $Categoria=$this->db_catactividad->all();
154.               $this->view->Categoria=$Categoria;
155.               //enviando actividades a la vista
156.               $Actividades=$this->db_bdtactividades-
    >TodoActividad(100);
157.               $this->view->actividades=$Actividades;
158.               //enviando departamentos a la vista
159.               $Departamento=$this->db_departamento-
    >departamentos();
160.               $this->view->departamento=$Departamento;
161.
162.                   if(isset ($Titulo)){
163.                       //variables del formulario por post
164.                       $Categoria=$this->getRequest()-
    >getParam("selcategoria");
165.                       $Distrito=$this->getRequest()-
    >getParam("seldistrito");
166.                       $Hora=$this->getRequest()-
    >getParam("txthora");
167.                       $Fecha=$this->getRequest()-
    >getParam("txtfecha");
168.                       $Costo=$this->getRequest()-
    >getParam("txtcosto");
169.                       $Descripcion=$this->getRequest()-
    >getParam("txtcontenido");
170.
171.                       //insertamos los datos a la base de
    datos
172.                       $CodActividad=$this->dbtactividades-
    >guardar($id, $Titulo, $Descripcion, $Fecha, $Costo, $this-
    >session->cod_usuario, $Categoria, $Distrito);
173.                       //insertamos las imagenes en la tabla
    de actividad imagenes
174.                       foreach($this->session2->imagen as
    $rutaverdadera =>$rutaficticia ){
175.                           $this->actividadimagenes-
    >guardar($id,  'uploads/'.$rutaverdadera, $CodActividad);
176.                       }
177.                       //eliminamos la session que contiene
    las imagenes
178.                       unset($this->session2->imagen);
179.                       //enviamos alertas a la vista
180.                        $this->view->alerta="Se ha ingresado
    la actividad correctamente";
181.                        $this->view->tipoalerta=1;
182.                   }else{
183.                        unset($this->session2->imagen);
184.                   }
185.           }else{
186.               $this->_redirect('admin/index');
187.           }
188.      }
```

```
189.
190.      /**
191.       * acción para subir multiples imagenes y almacenarlos
    en una session
192.       */
193.      public function multiuploadAction()
194.      {
195.          //verificamos si el usuario esta logueado
196.          if(isset($this->session->cod_usuario)){
197.          $delete=$this->getRequest()->getParam('delete');
198.          if($delete==1)
199.        {
200.                  //capturamos el nombre de la imagen a
    eliminar
201.              $name = $this->getRequest()-
    >getParam('filename');
202.                  //buscamos el nombre de la imagen en el
    array session
203.                  $dato=array_search($name,$this->session2-
    >imagen);
204.                  //preguntamos si existe
205.              if(file_exists('uploads/'.$dato))
206.              {
207.                      //eliminamos la imagen y el array que
    lo contiene
208.                      unlink('uploads/'.$dato);
209.                      unset($this->session2->imagen[$dato]);
210.                      //enviamos la respuesta ala vista
    mediante json
211.                      echo json_encode(array("res" => true));
212.              }
213.              else
214.              {
215.                      echo json_encode(array("res" =>
    false));
216.              }
217.        }
218.          //cantidad de caracteres del nuevo nombre de la
    imagen
219.          $caracteres=10;
220.          //creamos el nombre aleatorio para la imagen
221.          $random_pass = substr(md5(rand()),0,$caracteres);
222.          $numero = $random_pass;
223.          //declaramos carpeta contenedora
224.          $uploaddir = 'uploads/';
225.          //subimos la imagen cambiando la extencion a jpg
    por ser de menos peso
226.          $uploadfile = $uploaddir.
    basename($_FILES['file']['name']);
227.          if (move_uploaded_file($_FILES['file']['tmp_name'],
    $uploadfile)) {
228.                      $extension_file =
    explode(".",strtolower($uploadfile));
229.                      $num = count($extension_file)-1;
230.                      if($extension_file[$num] == "gif"){
231.                          $imagenxd =
    imagecreatefromgif($uploadfile);
232.
    $patch_grabar='uploads/'.$numero.'.jpg';
233.                      }
234.                      if($extension_file[$num] == "png" ){
```

```
235.                         $imagenxd =
    imagecreatefrompng($uploadfile);
236.
    $patch_grabar='uploads/'.$numero.'.jpg';
237.                     }
238.                     if($extension_file[$num] == "jpg" ||
    $extension_file[$num] == "jpeg"){
239.                         $imagenxd =
    imagecreatefromjpeg($uploadfile);
240.
    $patch_grabar='uploads/'.$numero.'.jpg';
241.                     }
242.                     imagejpeg($imagenxd,$patch_grabar,100);
243.                     $imagen ='uploads/'.$numero.'.jpg';
244.
245.                     unlink($uploadfile);
246.                     //almacenamos la imagen subida en un
    array
247.                     $this->session2-
    >imagen[$numero.'.jpg']=$_FILES['file']['name'];
248.                 }
249.         }
250.     }
251.
252.     /**
253.      * acción para selecionar departamento - provincia -
    distrito
254.      */
255.     public function ubicacionAction()
256.     {
257.         $parametro=$this->getRequest()->getParam("p");
258.         if($parametro=="p"){
259.             $cod_departamento=$this->getRequest()-
    >getParam("departamento");
260.             $provincias=$this->db_provincia-
    >provincias($cod_departamento);
261.             $this->view->provincia=$provincias;
262.         }if($parametro=="d"){
263.             $cod_provincia=$this->getRequest()-
    >getParam("provincia");
264.             $distritos=$this->db_distrito-
    >distritos($cod_provincia);
265.             $this->view->distrito=$distritos;
266.         }
267.     }
268.
269.     /**
270.      * MACTIVIDAD
271.      * acción para modificar la actividad
272.      */
273.     public function modificarAction()
274.     {
275.         if(isset($this->session->cod_usuario)){
276.             //Capturando el Id de la Actividad
277.             $CodActividad=$this->getRequest()-
    >getParam("cod");
278.             $Titulo=$this->getRequest()-
    >getParam("txttitulo");
279.             //verificamos si existe un codigo de actividad
280.
281.             if(isset($CodActividad)){
```

```
282.                 //enviando categorias de actividad a la
    vista
283.                 $Categoria=$this->db_catactividad->all();
284.                 $this->view->Categoria=$Categoria;
285.                 //enviando ciudad a la vista
286.                 $Departamento=$this->db_departamento-
    >departamentos();
287.                 $this->view->departamentos=$Departamento;
288.                 //buscamos la actividad en la base de datos
289.                 $Actividad=$this->db_bdtactividades-
    >CodActividad($CodActividad);
290.                 if(count($Actividad)>0){
291.
292.                     //traemos todas las imagenes de
293.                     $Actividad_img=$this-
    >db_actividadimagenes->cod_actividad($CodActividad);
294.                     //enviamos a la vista la actividad y
    sus imagenes
295.                     $this->view->actividad=$Actividad;
296.                     $this->view-
    >actividadimg=$Actividad_img;
297.                     //capturamos el codigo de la actividad
    y lo enviamos a la vista
298.                     $this->view-
    >codactividad=$Actividad[0]['cod_actividades'];
299.                     //con el cod_distrito traemos la
    provincia y el departamento
300.                     $provincia=$this->db_distrito-
    >distritoid($Actividad[0]['cod_distrito']);
301.                     $departamento=$this->db_provincia-
    >provinciaid($provincia[0]['cod_provincia']);
302.
303.                     //obtenemos todos las provincias del
    departamento
304.                      $provincias=$this->db_provincia-
    >provincias($departamento[0]['cod_departamento']);
305.                     //obtenemos todos los distritos del la
    provincia
306.                      $distritos=$this->db_distrito-
    >distritos($provincia[0]['cod_provincia']);
307.
308.                      $this->view-
    >departamento=$departamento[0]['cod_departamento'];
309.                      $this->view-
    >provincia=$provincia[0]['cod_provincia'];
310.                      $this->view-
    >distrito=$Actividad[0]['cod_distrito'];
311.
312.                      $this->view->provincias=$provincias;
313.                      $this->view->distritos=$distritos;
314.
315.                      if(isset ($Titulo)){
316.                         //variables del formulario por post
317.                         $Categoria=$this->getRequest()-
    >getParam("selcategoria");
318.                         $Distrito=$this->getRequest()-
    >getParam("seldistrito");
319.                         $Hora=$this->getRequest()-
    >getParam("txthora");
320.                         $Fecha=$this->getRequest()-
    >getParam("txtfecha");
```

```
                            $Costo=$this->getRequest()->getParam("txtcosto");
                            $Descripcion=$this->getRequest()->getParam("txtcontenido");

                            //insertamos los datos a la base de datos
                            $CodActividad=$this->dbtactividades->guardar($CodActividad, $Titulo, $Descripcion, $Fecha, $Costo, $this->session->cod_usuario, $Categoria, $Distrito);
                            //insertamos las imagenes en la tabla de actividad imagenes
                            if(count($this->session2->imagen)>0){
                            foreach($this->session2->imagen as $rutaverdadera =>$rutaficticia ){
                                $this->actividadimagenes->guardar($id,  'uploads/'.$rutaverdadera, $CodActividad);
                            }
                            //eliminamos la session que contiene las imagenes
                            unset($this->session2->imagen);
                            }
                            //enviamos alertas a la vista
                             $this->view->alerta="Se ha modificado la actividad correctamente";
                             $this->view->tipoalerta=1;
                        }else{
                             unset($this->session2->imagen);
                        }
                    }else{
                        $this->_redirect('error');
                    }
                }else{
                    $this->_redirect('error');
                }



            }else{
                $this->_redirect('admin/index');
            }

        }

        /**
         * acción para la pagina principal del administrador
         */
        public function portadaAction()
        {

        }

        /**
         * ELIMINAR
         * acción para eliminar imágenes
         */
        public function eliminarimgAction()
        {
```

```
370.          //verificamos que exista un sesión
371.          if(isset($this->session->cod_usuario)){
372.              //obtenemos todas las imágenes seleccionadas
373.              $ArrayImg=$this->getRequest()-
    >getParam("checkimg");
374.              $Cod=$this->getRequest()->getParam("txtcod");
375.              //$valor con este string se eliminan las
    imágenes de actividad, turismo, platostipicos, flora, fauna
376.              $Valor=$this->getRequest()->getParam("valor");
377.              //eliminamos
378.              if($Valor=='actividad'){
379.                  if(count($ArrayImg)!=0){
380.                      foreach($ArrayImg as $row){
381.                          $this->actividadimagenes-
    >eliminar($row);
382.                      }
383.                  }
384.                  $this-
    >_redirect('admin/modificar/cod/'.$Cod);
385.              }
386.              if($Valor=='turismo'){
387.                  if(count($ArrayImg)!=0){
388.                      foreach($ArrayImg as $row){
389.                          $this->turismoimagenes-
    >eliminar($row);
390.                      }
391.                  }
392.                  $this-
    >_redirect('admin/modificartu/cod/'.$Cod);
393.              }
394.              if($Valor=='plato'){
395.                  if(count($ArrayImg)!=0){
396.                      foreach($ArrayImg as $row){
397.                          $this->platosimagenes-
    >eliminar($row);
398.                      }
399.                  }
400.                  $this-
    >_redirect('admin/modificarpla/cod/'.$Cod);
401.              }
402.              if($Valor=='fauna'){
403.                  if(count($ArrayImg)!=0){
404.                      foreach($ArrayImg as $row){
405.                          $this->faunaimagenes-
    >eliminar($row);
406.                      }
407.                  }
408.                  $this-
    >_redirect('admin/modificarfa/cod/'.$Cod);
409.              }
410.              if($Valor=='flora'){
411.                  if(count($ArrayImg)!=0){
412.                      foreach($ArrayImg as $row){
413.                          $this->floraimagenes-
    >eliminar($row);
414.                      }
415.                  }
416.                  $this-
    >_redirect('admin/modificarflo/cod/'.$Cod);
417.              }
418.
```

```
419.          }else{
420.              $this->_redirect('admin/index');
421.          }
422.      }
423.
424.      /**
425.       * ELIMINARIMG
426.       * acción para eliminar registros de base de datos
427.       */
428.      public function eliminarAction()
429.      {
430.          //verificamos que exista un sesión
431.          if(isset($this->session->cod_usuario)){
432.              //capturamos el codigo de la tupla a eliminar
433.              $CodGeneral=$this->getRequest()-
    >getParam("cod");
434.              $Valor=$this->getRequest()->getParam("valor");
435.              if($Valor=='actividad'){
436.                  //tabla hijo eliminamos las imagenes
437.                  $ActImg=$this->db_actividadimagenes-
    >cod_actividad($CodGeneral);
438.                  foreach($ActImg as $row){
439.                      $this->actividadimagenes-
    >eliminar($row['cod_actimagen']);
440.                  }
441.                  $this->dbtactividades-
    >eliminar($CodGeneral);
442.                  $this->_redirect('admin/actividad');
443.              }
444.              if($Valor=='turismo'){
445.
446.                  //tabla hijo eliminamos las imagenes
447.                  $TuriImg=$this->db_turismoimagenes-
    >CodTurismo($CodGeneral);
448.
449.                          $permiso=$this->db_turimopermiso-
    >TurismoPermisoUsuario($CodGeneral);
450.                          foreach($permiso as $fil){
451.                                $this->turismopermiso-
    >eliminar($fil['cod_turpermiso']);
452.                          }
453.                  foreach($TuriImg as $row){
454.                      $this->turismoimagenes->eliminar($row);
455.                  }
456.                  $this->dbtturismo->eliminar($CodGeneral);
457.                  $this->_redirect('admin/turismo');
458.              }
459.              if($Valor=='platos'){
460.                  //tabla hijo eliminamos las imagenes
461.                  $PlatoImg=$this->db_platosimagenes-
    >CodPlatosTipicos($CodGeneral);
462.                  foreach($PlatoImg as $row){
463.                      $this->platosimagenes->eliminar($row);
464.                  }
465.                  $this->platostipicos-
    >eliminar($CodGeneral);
466.                  $this->_redirect('admin/platostipicos');
467.              }
468.              if($Valor=='fauna'){
469.                  //tabla hijo eliminamos las imagenes
```

```
470.                 $FaunaImg=$this->db_faunaimagenes-
    >CodFauna($CodGeneral);
471.                 foreach($FaunaImg as $row){
472.                     $this->faunaimagenes->eliminar($row);
473.                 }
474.                 $this->fauna->eliminar($CodGeneral);
475.                 $this->_redirect('admin/fauna');
476.             }
477.             if($Valor=='flora'){
478.                 //tabla hijo eliminamos las imagenes
479.                 $FloraImg=$this->db_floraimagenes-
    >CodFlora($CodGeneral);
480.                 foreach($FloraImg as $row){
481.                     $this->floraimagenes->eliminar($row);
482.                 }
483.                 $this->flora->eliminar($CodGeneral);
484.                 $this->_redirect('admin/flora');
485.             }
486.         }else{
487.             $this->_redirect('admin/index');
488.         }
489.
490.     }
491.
492.     /**
493.      * ITURISMO
494.      * acción para el ingreso de nuevos sitios turisticos
495.      */
496.     public function turismoAction()
497.     {
498.          //existe usuario logueado de lo contrario error
499.          if(isset($this->session->cod_usuario)){
500.
501.              $TuriNombre=$this->getRequest()-
    >getParam("txttitulo");
502.              //enviando categorias del turismo a la vista
503.              $Categoria=$this->db_catturismo->all();
504.              $this->view->Categoria=$Categoria;
505.              //enviando los sistios turisticos a la vista
506.
507.
508.              //enviando departamentos a la vista
509.              $Departamento=$this->db_departamento-
    >departamentos();
510.              $this->view->departamento=$Departamento;
511.
512.
513.              $Turismo=$this->db_bdturismo-
    >TodoTurismoUser(100, $this->session->cod_usuario);
514.
515.              $this->view->turismo=$Turismo;
516.                  if(isset ($TuriNombre)){
517.                     //variables del formulario por post
518.                     $CodTucategoria=$this->getRequest()-
    >getParam("selcategoria");
519.                     $CodDistrito=$this->getRequest()-
    >getParam("seldistrito");
520.                     $TuriDescripcion=$this->getRequest()-
    >getParam("txtcontenido");
521.                     $TxtLatitud=$this->getRequest()-
    >getParam("txtlatitud");
```

```
522.                     $TxtLongitud=$this->getRequest()->getParam("txtlongitud");
523.
524.                     //insertamos los datos a la base de datos
525.                     $CodTurismo=$this->dbtturismo->guardar($CodTurismo, $TuriNombre, $TuriDescripcion, $CodTucategoria, $this->session->cod_usuario, $CodDistrito,$TxtLatitud,$TxtLongitud);
526.                     $this->turismopermiso->guardar(0,$CodTurismo,$this->session->cod_usuario,0);
527.                     //insertamos las imagenes en la tabla de turismo_imagenes
528.                     foreach($this->session2->imagen as $rutaverdadera =>$rutaficticia ){
529.                         $this->turismoimagenes->guardar($id,  'uploads/'.$rutaverdadera, $CodTurismo);
530.                     }
531.                     //eliminamos la session que contiene las imagenes
532.                     unset($this->session2->imagen);
533.                     //enviamos alertas a la vista
534.                      $this->view->alerta="Se ha ingresado el sitio turistico correctamente";
535.                      $this->view->tipoalerta=1;
536.                  }else{
537.                       unset($this->session2->imagen);
538.                  }
539.           }else{
540.               $this->_redirect('admin/index');
541.           }
542.      }
543.
544.      /**
545.       * MTURISMO
546.       * acción para modificar sitios turisticos
547.       */
548.      public function modificartuAction()
549.      {
550.         if(isset($this->session->cod_usuario)){
551.              //Capturando el Id del sitio turistico
552.              $CodTurismo=$this->getRequest()->getParam("cod");
553.              $TuriNombre=$this->getRequest()->getParam("txttitulo");
554.
555.              //verificamos si existe el sitio turistico
556.              if(isset($CodTurismo)){
557.                  //enviando categorias de sitio turistico a la vista
558.                  $Categoria=$this->db_catturismo->all();
559.                  $this->view->Categoria=$Categoria;
560.                  //enviando ciudad a la vista
561.                  $Departamento=$this->db_departamento->departamentos();
562.                  $this->view->departamentos=$Departamento;
563.                  //buscamos el sitio turistico  en la base de datos
564.                  $DatosTurismo=$this->db_bdturismo->CodTurismo($CodTurismo);
565.
```

```
566.
567.                 //si el rol de la persona autentificada es
    superadmin o admin le
568.                 //permite modificar cualquier tupla
569.                 if($this->session->rol=='superadmin' ||
    $this->session->rol=='moderador'){
570.                     $DatosTurismo=$this->db_bdturismo-
    >CodTurismo($CodTurismo);
571.                     $this->view->admin=true;
572.
573.                 }
574.                 //si el rol es usuario solo modificará sus
    tuplas que no estan publicadas
575.                 if($this->session->rol=='usuario'){
576.                     $DatosTurismo=$this->db_bdturismo-
    >CodTurismoUser($CodTurismo,$this->session->cod_usuario);
577.                     $this->view->admin=false;
578.                 }
579.
580.                 if(count($DatosTurismo)>0){
581.
582.                     //traemos todas las imagenes del sitio
    turistico
583.                     $TurismoImg=$this->db_turismoimagenes-
    >CodTurismo($CodTurismo);
584.                     //enviamos a la vista el sitio
    turistico y sus imágenes
585.                     $this->view->turismo=$DatosTurismo;
586.                     $this->view->turismoimg=$TurismoImg;
587.                     //capturamos el codigo de la
    información turistica y lo enviamos a la vista
588.                     $this->view->codturismo=$CodTurismo;
589.                     //echo
    $TurismoImg[0]['cod_turismo'];exit;
590.                     //con el cod_distrito traemos la
    provincia y el departamento
591.                     $provincia=$this->db_distrito-
    >distritoid($DatosTurismo[0]['cod_distrito']);
592.                     $departamento=$this->db_provincia-
    >provinciaid($provincia[0]['cod_provincia']);
593.
594.                     //obtenemos todos las provincias del
    departamento
595.                      $provincias=$this->db_provincia-
    >provincias($departamento[0]['cod_departamento']);
596.                     //obtenemos todos los distritos del la
    provincia
597.                      $distritos=$this->db_distrito-
    >distritos($provincia[0]['cod_provincia']);
598.
599.                      $this->view-
    >departamento=$departamento[0]['cod_departamento'];
600.                      $this->view-
    >provincia=$provincia[0]['cod_provincia'];
601.                      $this->view-
    >distrito=$DatosTurismo[0]['cod_distrito'];
602.
603.                      $this->view->provincias=$provincias;
604.                      $this->view->distritos=$distritos;
605.
606.                      if(isset ($TuriNombre)){
```

```
607.                            //variables del formulario por post
608.                            $CodTucategoria=$this-
    >getRequest()->getParam("selcategoria");
609.                            $CodDistrito=$this->getRequest()-
    >getParam("seldistrito");
610.                            $TuriDescripcion=$this-
    >getRequest()->getParam("txtcontenido");
611.                            $TxtLatitud=$this->getRequest()-
    >getParam("txtlatitud");
612.                            $TxtLongitud=$this->getRequest()-
    >getParam("txtlongitud");
613.
614.                            //insertamos los datos a la base de
    datos
615.                            $CodTurismo=$this->dbtturismo-
    >guardar($CodTurismo, $TuriNombre, $TuriDescripcion,
    $CodTucategoria, $this->session->cod_usuario,
    $CodDistrito,$TxtLatitud,$TxtLongitud);
616.                            //insertamos las imagenes en la
    tabla de actividad imagenes
617.                            if(count($this->session2-
    >imagen)>0){
618.                            foreach($this->session2->imagen as
    $rutaverdadera =>$rutaficticia ){
619.                                $this->turismoimagenes-
    >guardar($CodTuImagen, 'uploads/'.$rutaverdadera,
    $CodTurismo);
620.                            }
621.                            //eliminamos la session que
    contiene las imagenes
622.                            unset($this->session2->imagen);
623.                            }
624.                            //enviamos alertas a la vista
625.                             $this->view->alerta="Se ha
    modificado el sitio turístico correctamente";
626.                             $this->view->tipoalerta=1;
627.                        }else{
628.                             unset($this->session2->imagen);
629.                        }
630.                    }else{
631.                        $this->_redirect('error');
632.                    }
633.                }else{
634.                    $this->_redirect('admin/index');
635.                }
636.
637.
638.
639.
640.            }else{
641.                $this->_redirect('error');
642.            }
643.
644.        }
645.
646.        /**
647.         * IFLORA
648.         * acción para ingresar flora
649.         */
650.        public function floraAction()
651.        {
```

```
652.          //existe usuario logueado de lo contrario error
653.          if(isset($this->session->cod_usuario)){
654.               $FloNombre=$this->getRequest()-
     >getParam("txttitulo");
655.               //enviando sitios turisticos a la vista
656.               $SitiosTuristicos=$this->db_bdturismo-
     >TodoTurismo();
657.               $this->view->TodoTurismo=$SitiosTuristicos;
658.
659.               //si el rol de la persona autentificada es
     superadmin o admin le
660.               //muestra todas las tuplas de tabla flora
661.               if($this->session->rol=='superadmin' || $this-
     >session->rol=='moderador'){
662.                    $Flora=$this->db_flora-
     >TodoFlora($cantidad);
663.                    //enviamos a la vista la variable para que
     pueda modificar cualquier tupla
664.                    $this->view->admin=true;
665.               }
666.               //si el rol es usuario le mostrara sus propias
     publicaciones
667.
668.               if($this->session->rol=='usuario'){
669.                    $Flora=$this->db_flora->TodoFloraUser(100,
     $this->session->cod_usuario);
670.                    $this->view->admin=false;
671.
672.               }
673.
674.               $this->view->Flora=$Flora;
675.                    if(isset ($FloNombre)){
676.                       //variables del formulario por post
677.                       $CodTurismo=$this->getRequest()-
     >getParam("selcategoria");
678.                       $FloDescripcion=$this->getRequest()-
     >getParam("txtcontenido");
679.
680.                       //insertamos los datos a la base de
     datos
681.                       $CodFlora=$this->flora-
     >guardar($CodFlora, $FloNombre, $FloDescripcion, 0, $this-
     >session->cod_usuario, $CodTurismo);
682.                       //insertamos las imagenes en la tabla
     de flora_imagenes
683.
684.                       foreach($this->session2->imagen as
     $rutaverdadera =>$rutaficticia ){
685.                            $this->floraimagenes->guardar($id,
     'uploads/'.$rutaverdadera, $CodFlora);
686.
687.                       }
688.                       //eliminamos la session que contiene
     las imagenes
689.                       unset($this->session2->imagen);
690.                       //enviamos alertas a la vista
691.                        $this->view->alerta="Se ha ingresado
     la flora correctamente";
692.                        $this->view->tipoalerta=1;
693.                    }else{
694.                         unset($this->session2->imagen);
```

```
                    }
            }else{
                $this->_redirect('admin/index');
            }
        }

        /**
         * MFLORA
         * acción para modificar la flora
         */
        public function modificarfloAction()
        {
            //preguntamos si esta logueado en el sistema
            if(isset($this->session->cod_usuario)){
                //Capturando el Id de la flora
                $CodFlora=$this->getRequest()->getParam("cod");
                $FloNombre=$this->getRequest()->getParam("txttitulo");

                //verificamos si existe la flora
                if(isset($CodFlora)){
                    //enviando sitios turisticos a la vista
                    $SitiosTuristicos=$this->db_bdturismo->TodoTurismo();
                    $this->view->TodoTurismo=$SitiosTuristicos;

                    //si el rol de la persona autentificada es superadmin o admin le
                    //permite modificar cualquier tupla
                    if($this->session->rol=='superadmin' || $this->session->rol=='moderador'){
                        $DatosFlora=$this->db_flora->CodFlora($CodFlora);
                        $this->view->admin=true;
                        $FloPermiso=$this->getRequest()->getParam("iradio");
                    }
                    //si el rol es usuario solo modificará sus tuplas que no estan publicadas
                    if($this->session->rol=='usuario'){
                        $DatosFlora=$this->db_flora->CodFloraUser($CodFlora, $this->session->cod_usuario);
                        $this->view->admin=false;
                        $FloPermiso=0;
                    }

                    //exiten datos
                    if(count($DatosFlora)>0){

                        //traemos todas las imagenes de la flora
                        $FloraImg=$this->db_floraimagenes->CodFlora($CodFlora);
                        //enviamos a la vista la flora y sus imágenes
                        $this->view->Flora=$DatosFlora;
                        $this->view->FloraImg=$FloraImg;
                        //capturamos el codigo de la fauna y lo enviamos a la vista
                        $this->view->codflora=$FloraImg[0]['cod_flora'];
```

```
743.
744.                     //si existe un dato enviado por post se
    modifica la tupla
745.                     if(isset ($FloNombre)){
746.                         //variables del formulario por post
747.                         $CodTurismo=$this->getRequest()-
    >getParam("selcategoria");
748.                         $FloDescripcion=$this-
    >getRequest()->getParam("txtcontenido");
749.                         $CodUsuario=$this->getRequest()-
    >getParam("txtcodusuario");
750.
751.                         //insertamos los datos a la base de
    datos
752.                         $CodFlora=$this->flora-
    >guardar($CodFlora, $FloNombre,
    $FloDescripcion,$FloPermiso, $CodUsuario, $CodTurismo);
753.                         //insertamos las imagenes en la
    tabla de flora_imagenes
754.                         foreach($this->session2->imagen as
    $rutaverdadera =>$rutaficticia ){
755.                             $this->floraimagenes-
    >guardar($id, 'uploads/'.$rutaverdadera, $CodFlora);
756.
757.                         }
758.                         //eliminamos la session que
    contiene las imagenes
759.                         unset($this->session2->imagen);
760.                         //enviamos alertas a la vista
761.                          $this->view->alerta="Se ha
    ingresado la flora correctamente";
762.                          $this->view->tipoalerta=1;
763.                     }else{
764.                       unset($this->session2->imagen);
765.                     }
766.
767.                 }else{
768.                     $this->_redirect('error');
769.                 }
770.             }else{
771.                 $this->_redirect('admin/index');
772.             }
773.
774.         }else{
775.             $this->_redirect('error');
776.         }
777.     }
778.
779.     /**
780.      * MPERFIL
781.      * acción para modificar su perfil
782.      */
783.     public function perfilAction()
784.     {
785.         //existe usuario logueado de lo contrario error
786.
787.         if(isset($this->session->cod_usuario)){
788.
789.             //informacíon del usuario enviado a la vista
790.             $Datos=$this->db_usuario->UsuarioDatos($this-
    >session->cod_usuario);
```

```
791.              $this->view->UsuarioDatos=$Datos;
792.              $Datos2=$this->db_usuariodato-
    >UsuarioDatos($this->session->cod_usuario);
793.              $this->view->UsuarioPerfil=array("");
794.              $Departamento=$this->db_departamento-
    >departamentos();
795.              $this->view->departamentos=$Departamento;
796.
797.               if(count($Datos2)>0){
798.                    $this->view->UsuarioPerfil=$Datos2;
799.
800.                     }
801.              //variables del formulario por post
802.              $txtNombre=$this->getRequest()-
    >getParam("txtNombre");
803.              $txtIdIngreso=$this->getRequest()-
    >getParam("txtIdIngreso");
804.              $txtContrasena=$this->getRequest()-
    >getParam("txtContrasena");
805.              if(isset ($txtNombre)){
806.                  $txtApellido=$this->getRequest()-
    >getParam("txtApellido");
807.                  $txtCelular=$this->getRequest()-
    >getParam("txtCelular");
808.                  $selSexo=$this->getRequest()-
    >getParam("selsexo");
809.                  $selDistrito=$this->getRequest()-
    >getParam("seldistrito");
810.
811.                  //subiendo una imágen
812.                  $caracteres=10;
813.                  $random_pass =
    substr(md5(rand()),0,$caracteres);
814.                  $numero = $random_pass;
815.                  $uploaddir = 'uploads/';
816.                  $uploadfile = $uploaddir .
    basename($_FILES['filename']['name']);
817.                  if
    (move_uploaded_file($_FILES['filename']['tmp_name'],
    $uploadfile)) {
818.                              $extension_file =
    explode(".",strtolower($uploadfile));
819.                              $num = count($extension_file)-1;
820.                              if($extension_file[$num] ==
    "gif"){
821.                              $imagenxd =
    imagecreatefromgif($uploadfile);
822.
    $patch_grabar='uploads/'.$numero.'.jpg';
823.                              }
824.                              if($extension_file[$num] == "png"
    ){
825.                              $imagenxd =
    imagecreatefrompng($uploadfile);
826.
    $patch_grabar='uploads/'.$numero.'.jpg';
827.                              }
828.                              if($extension_file[$num] == "jpg"
    || $extension_file[$num] == "jpeg"){
829.                              $imagenxd =
    imagecreatefromjpeg($uploadfile);
```

```
830.
    $patch_grabar='uploads/'.$numero.'.jpg';
831.                             }
832.
    imagejpeg($imagenxd,$patch_grabar,100);
833.                             $imagen
    ='uploads/'.$numero.'.jpg';
834.
835.                             unlink($uploadfile);
836.
837.                           }else{
838.                             $imagen =
    $Datos2[0]['usua_imagen'];
839.                             }
840.                 $this->usuarioDato-
    >guardar($Datos2[0]['cod_usudato'],$this->session-
    >cod_usuario, $txtNombre, $txtApellido, $selSexo, $imagen);
841.                 //enviando alerta
842.                       $this->session->foto=$imagen;
843.                 $this->view->alerta="Se Modifico
    correctamente su Perfil";
844.                 $this->view->tipoalerta=1;
845.
846.             }
847.             if(isset($txtIdIngreso)){
848.                 $txtCorreo=$this->getRequest()-
    >getParam("txtCorreo");
849.                 $this->usuario->guardar($this->session-
    >cod_usuario, $txtIdIngreso, $txtCorreo);
850.                 $this->view->alerta="Se Modifico
    correctamente su Información de acceso";
851.                 $this->view->tipoalerta=1;
852.             }
853.             if(isset($txtContrasena)){
854.
    if($txtContrasena==$Datos[0]['usu_contrasena']){
855.                 $txtContNueva=$this->getRequest()-
    >getParam("txtContNueva");
856.                 $this->usuario->ModificarContrasena($this-
    >session->cod_usuario, $txtContNueva);
857.                 $this->view->alerta="Se Modifico
    correctamente su contraseña";
858.                 $this->view->tipoalerta=1;
859.                 }else{
860.                 $this->view->alerta="La contraseña antigua
    no es correcta";
861.                 $this->view->tipoalerta=0;
862.                 }
863.             }
864.         }else{
865.             $this->_redirect('admin/index');
866.         }
867.
868.     }
869.
870.     public function interesAction()
871.     {
872.         //existe usuario logueado de lo contrario error
873.         if(isset($this->session->cod_usuario)){
874.             //valores por post - buscar
```

```
875.              $txtTexto=$this->getRequest()-
      >getParam("txtTexto");
876.              $txtFecha=$this->getRequest()-
      >getParam("txtFecha");
877.              $txtVal=$this->getRequest()->getParam("val");
878.              $mostrartodo=true;
879.
880.              //fecha
881.              $date = new Zend_Date();
882.              $date->setTimezone('America/Lima');
883.              $this->view->fecha=$date->get("Y-M-d");
884.              if($txtVal=="anio"){
885.                  $Turismo=$this->db_usuarioturismo-
      >UsuarioVisitaAnio($this->session->cod_usuario,$date-
      >get("Y"));
886.                  $mostrartodo=false;
887.              }
888.              if($txtVal=="mes"){
889.                  $Turismo=$this->db_usuarioturismo-
      >UsuarioVisitaMes($this->session->cod_usuario,$date-
      >get("M"));
890.                  $mostrartodo=false;
891.              }
892.              if(isset($txtTexto) && $txtTexto!=""){
893.                  $Turismo=$this->db_usuarioturismo-
      >UsuarioVisitaNombre($this->session-
      >cod_usuario,$txtTexto);
894.
895.                  $mostrartodo=false;
896.              }
897.              if(isset($txtFecha)){
898.                  $Turismo=$this->db_usuarioturismo-
      >UsuarioVisitaFecha($this->session->cod_usuario,$txtFecha);
899.                  $mostrartodo=false;
900.              }
901.
902.              if($mostrartodo==true){
903.              //Array de sitios turisticos de interes del
      usuario
904.              $Turismo=$this->db_usuarioturismo-
      >UsuarioTurismo($this->session->cod_usuario, 100);
905.              }
906.
907.              //imágenes de cada sitio turistico
908.              $ImagenesTurismo=array();
909.              $i=0;
910.              foreach($Turismo as $row){
911.                  $imagenes=$this->db_turismoimagenes-
      >CodTurismo($row['cod_turismo']);
912.                  $ImagenesTurismo[$i]=$imagenes;
913.                  $i++;
914.              }
915.              $i=0;
916.              foreach($Turismo as $row2){
917.
918.                  $date2 = new
      Zend_Date($row2['usutur_fecha']);
919.                  $nameFecha=$date2->get("d").' '.$date2-
      >get(Zend_Date::MONTH_NAME_SHORT).' '.$date2->get("Y");
920.                  list($y,$m,$d)=  explode("-",
      $row2['usutur_fecha']);
```

```
921.                 if($d==$date->get("d")){
922.                    $nameFecha="Hoy";
923.                 }
924.                 if(($date->get("d")-1)==$d || ($date-
    >get("d")-1)==0){
925.                    $nameFecha="Ayer";
926.                 }
927.
928.                             $ArrayDatos[$i]=array(
929.                                       "fecha" => $nameFecha,
930.                                       "nombre" =>
    $row2['turi_nombre'],
931.                                       "categoria"=>
    $row2['cod_tucategoria'],
932.                                       "distrito"=>
    $row2['cod_distrito'],
933.                                       "turi_latitud"=>
    $row2['turi_latitud'],
934.                                       "turi_longitud"=>
    $row2['turi_longitud'],
935.                                       "imagenes"=>
    $ImagenesTurismo[$i],
936.                                  );
937.                             $i++;
938.             }
939. //         echo '<pre>';
940. //         print_r($ArrayDatos);
941. //         echo '<pre>';exit;
942.               $this->view->arraydatos=$ArrayDatos;
943.           }else{
944.               $this->_redirect('admin/index');
945.           }
946.     }
947.
948.     public function turispermisoAction()
949.     {
950.         if(isset($this->session->cod_usuario)){
951.             $Permiso=$this->getRequest()->getParam("per");
952.             $CodTurismo=$this->getRequest()-
    >getParam("cod");
953.             $CodTurpermiso=$this->getRequest()-
    >getParam("cod2");
954.             $CodUsuario=$this->session->cod_usuario;
955.             if(isset($Permiso)){
956.                 $this->turismopermiso-
    >guardar($CodTurpermiso ,$CodTurismo,$CodUsuario,$Permiso);
957.                 $this-
    >_redirect('admin/modificartu/cod/'.$CodTurismo);
958.             }
959.         }
960.     }
961.
962.     public function listaAction()
963.     {
964.       $Turismo=$this->db_bdturismo->TodoTurismoUser(100,
    $this->session->cod_usuario);
965.       $this->view->turismo=$Turismo;
966.     }
967.
968.     public function lista2Action()
969.     {
```

```
970.         $Turismo=$this->db_bdturismo->TodoTurismoAdmin(100,
     $this->session->cod_usuario);
971.         $this->view->turismo=$Turismo;
972.       }
973.
974.       public function rolusuarioAction()
975.       {
976.           $Usuarios=$this->db_usuario->TodoUsuarioRol();
977.           $this->view->usuarios=$Usuarios;
978.           $Radio=$this->getRequest()->getParam("radio1");
979.           $Alert=$this->getRequest()->getParam("alert");
980.           $Cancelar=$this->getRequest()->getParam("cancel");
981.           $Value=$this->getRequest()->getParam("cod");
982.
983.           if(isset($Value)){
984.             $i=1;
985.
986.             foreach($Usuarios as $row){
987.                   $Radio=$this->getRequest()-
     >getParam("radio".$i);
988.                   if($Radio!=""){
989.                   $this->usuario-
     >ModificarRol($row['cod_usuario'] ,$Radio);
990.                   }
991.                   $i++;
992.             }
993.            $this->_redirect('admin/rolusuario/alert/1');
994.           }
995.           if(isset($Alert)){
996.               $this->view->alerta="Se Modifico
     Correctamente";
997.               $this->view->tipoalerta=1;
998.           }
999.           if($Cancelar=="y"){
1000.              $this->view->alerta="Se Cancelo el proceso, no
     se realizo ninguna modificacíon";
1001.              $this->view->tipoalerta=2;
1002.          }
1003.      }
1004.
1005.      public function reporrolAction()
1006.      {
1007.        //portes de roles de cada usuario
1008.          $Usuarios=$this->db_usuario->TodoUsuarioRol();
1009.          $UsuariosTabla=$this->db_usuario-
     >TodoUsuarioRolReporte();
1010.          $ArrayGrafico=array();
1011.          $i=0;
1012.          foreach($UsuariosTabla as $row){
1013.
     $array=array($row['rol_nombre'],intval($row['cantidad']));
1014.              $ArrayGrafico[$i]=$array;
1015.              $i++;
1016.          }
1017.
1018.          $this->view->usuarios=$Usuarios;
1019.          $this->view->arraygrafico=$ArrayGrafico;
1020.      }
1021.
1022.      public function reportpostAction()
1023.      {
```

```
1024.       $Turismo=$this->db_bdturismo-
      >AllInfoturisticaReport();
1025.         $ArrayGrafico=array();
1026.         $i=0;
1027.         foreach($Turismo as $row){
1028.             if($i<10){
1029.
      $array=array($row['usu_nombre'],intval($row['cantidad']));
1030.             $ArrayGrafico[$i]=$array;
1031.             $i++;
1032.             }
1033.         }
1034.       $this->view->reporte=$Turismo;
1035.       $this->view->arraygrafico=$ArrayGrafico;
1036.
1037.     }
1038.
1039.     public function reportpostaAction()
1040.     {
1041.       $Turismo=$this->db_turimopermiso-
      >AllInfoturisticaReport();
1042.         $ArrayGrafico=array();
1043.         $i=0;
1044.         foreach($Turismo as $row){
1045.             if($i<10){
1046.
      $array=array($row['usu_nombre'],intval($row['cantidad']));
1047.             $ArrayGrafico[$i]=$array;
1048.             $i++;
1049.             }
1050.         }
1051.       $this->view->reporte=$Turismo;
1052.       $this->view->arraygrafico=$ArrayGrafico;
1053.
1054.     }
1055.
1056.     public function pearsonAction()
1057.     {
1058.             //recomendacion verificamos si existen datos en
      el algoritmo
1059.             $codUsuario=$this->getRequest()-
      >getParam("cod");
1060.             $AllUser=$this->db_usuario->TodoUsuario2();
1061.             $Algoritmo = array();
1062.             if(isset($codUsuario)){
1063.
1064.             $Algoritmo=$this-
      >_algoritmoAction($codUsuario);
1065.             $CiudadAll=$this->db_bdturismo-
      >TodoTurismo(50);
1066.             $Distancia=$Algoritmo[2];
1067.             $Algoritmo=$Algoritmo[0];
1068.
1069.             arsort($Algoritmo);
1070.             $recomenDatos=array();
1071.             $Tablas=array();
1072.             $h=0;
1073.
1074.             if(count($Algoritmo)>0){
1075.                 //CUADRO1
1076.
```

```
1077.                 $Array=array();
1078.                 $Array2=array();
1079.
1080.                 foreach($AllUser as $fil){
1081.                     $CalificacionUser=$this-
      >db_usuariocalificacion-
      >CalificacionUsuario($fil['cod_usuario']);
1082.                     $Array["usuario"]=$fil["usu_nombre"];
1083.
      $Array["codusuario"]=$fil["cod_usuario"];
1084.                     foreach($CiudadAll as $fil2){
1085.                         $Calificacion="x";
1086.                         foreach($CalificacionUser as
      $fil3){
1087.
      if($fil2['cod_turismo']==$fil3['cod_turismo']){
1088.
      $Calificacion=$fil3['turi_calificacion'];
1089.                             }
1090.                          }
1091.
      $Array[$fil2['cod_turismo']]=$Calificacion;
1092.                         }
1093.                          array_push($Array2,$Array);
1094.                     }
1095.
1096.
1097.                 // GRAFICOS 1
1098.                   $ArrayGrafico=array();
1099.                   $i=0;
1100.                   foreach($Distancia as $row){
1101.                       if($i<10){
1102.                       $array=array($row['UsuarioN'],
      floatval($row['Distanciae']));
1103.                       $ArrayGrafico[$i]=$array;
1104.                       $i++;
1105.                       }
1106.                   }
1107.                 // GRAFICOS 2
1108.                   $ArrayGrafico2=array();
1109.                   $j=0;
1110.                 foreach($Algoritmo as $indice => $row){
1111.                     $datos=$this->db_bdturismo-
      >CodTurismo($indice);
1112.
      $array=array($datos[0]['turi_nombre'],  floatval($row));
1113.                       $ArrayGrafico2[$j]=$array;
1114.                       $j++;
1115.
1116.                 }
1117.                 $this->view->arraygrafico2=$ArrayGrafico2;
1118.                 $this->view->arraygrafico=$ArrayGrafico;
1119.                 $this->view->usuario=$codUsuario;
1120.                 $this->view->recomendacion=$recomenDatos;
1121.                 $this->view->Ciudades=$CiudadAll;
1122.                 $this->view->tabla=$Array2;
1123.             }else{
1124.                 $this->view->alerta="No se encontraron
      patrones para la recomendación turística";
1125.                 $this->view->tipoalerta="0";
1126.                  $this->view->alerta2="0";
```

```
1127.             }
1128.             }else{
1129.                 $this->view->alerta2="0";
1130.             }
1131.             $this->view->usuarios=$AllUser;
1132.     }
1133.
1134.
1135.
1136.    //funcion que crea un lista
      array[informacionturistica]=calificacion
1137.     public function _crearlista($CalificacionUser)
1138.     {
1139.         $Array=array();
1140.         $i=0;
1141.         foreach($CalificacionUser as $row2){
1142.
      $Array[$row2['cod_turismo']]=$row2['turi_calificacion'];
1143.         }
1144.          return $Array;
1145.     }
1146.     //funcion array almacena informacion turistica que
      califico el usuario
1147.     public function _turismoUser($User)
1148.     {
1149.         $Array=array();
1150.         $i=0;
1151.         foreach($User as $row2){
1152.               $Array[$i]=$row2['cod_turismo'];
1153.               $i++;
1154.         }
1155.          return $Array;
1156.     }
1157.     //funcion array devuelve calificaciones de información
      turistica en comun de user y user nuevo
1158.     public function
      _turismoIntersect($Data,$ArrayIntersect)
1159.     {
1160.         $Array=array();
1161.         $i=0;
1162.         foreach($ArrayIntersect as $Turismo){
1163.             foreach($Data as $indice1 => $fil1){
1164.                 if($Turismo==$indice1){
1165.                     $Array[$i]=$fil1;
1166.                     $i++;
1167.                 }
1168.             }
1169.         }
1170.          return $Array;
1171.     }
1172.     //funcion array almacena informacion turistica que
      califico el usuario
1173.     public function _pearson($x,$y)
1174.     {
1175.         $length= count($x);
1176.         $mean1=array_sum($x) / $length;
1177.         $mean2=array_sum($y) / $length;
1178.
1179.         $a=0;
1180.         $b=0;
1181.         $axb=0;
```

```
1182.         $a2=0;
1183.         $b2=0;
1184.
1185.         for($i=0;$i<$length;$i++)
1186.         {
1187.             $a=$x[$i]-$mean1;
1188.             $b=$y[$i]-$mean2;
1189.             $axb=$axb+($a*$b);
1190.             $a2=$a2+ pow($a,2);
1191.             $b2=$b2+ pow($b,2);
1192.         }
1193.         $denominador=sqrt($a2*$b2);
1194.         //si el denominador es igual a 0
1195.         if($denominador!=0){
1196.             $corr= $axb /$denominador;
1197.         }else{
1198.             $corr=0;
1199.         }
1200.         //si el coeficiente de correlacion es negativo
1201.         if($corr<0){
1202.              return 0;
1203.         }else{
1204.            return $corr;
1205.         }
1206.
1207.     }
1208.     public function _algoritmoAction($codUsuario)
1209.     {
1210.           // action body
1211.         //traemos todos los usuarios del sistema menos el
      usuario en session
1212.         $AllUser=$this->db_usuario-
      >TodoUsuario($codUsuario);
1213.         //calificacion del usuario nuevo
1214.
1215.         $Nuevo=$this->db_usuariocalificacion-
      >CalificacionUsuario($codUsuario);
1216.         //si exiten calificacion del nuevo usuario
      procedemos con el algoritmo
1217.
1218.         if(count($Nuevo)>0){
1219.         $Datos1=$this->_crearlista($Nuevo);
1220.         $TurismoUserNuevo=$this->_turismoUser($Nuevo);
1221.
1222.
1223.         $NuevoAllUser=array();
1224.         $arrayTurismo=array();
1225.         $i=0;
1226.         foreach($AllUser as $row){
1227.             $CalificacionUser=$this-
      >db_usuariocalificacion-
      >CalificacionUsuario($row['cod_usuario']);
1228.             $Datos=$this->_crearlista($CalificacionUser);
1229.             $turismoUser =$this-
      >_turismoUser($CalificacionUser);
1230.            //intersecando que exista ciudades en comun
      entre las dos personas
1231.              $result = array_intersect($turismoUser,
      $TurismoUserNuevo);
1232.
1233.             // si existe alguna ciudad en comun
```

```
1234.             if(count($result)>1){
1235.                 $Valor=0;
1236.                 //datos de la persona que entra en bucle
1237.                 //indice1 ciudad que voto fil1 el valor
1238.                 //datos1 usuario nuevo
1239.                 //indice2 ciudad que voto fil2 el valor
1240.
1241.                 //DISTANCIA EUCLIDEANA
1242.                 foreach ($Datos as $indice1 => $fil1){
1243.                     foreach ($Datos1 as $indice2 => $fil2){
1244.                         if($indice1==$indice2){
1245.                             $Valor=$Valor + pow(($fil2-
      $fil1), 2);
1246.                         }
1247.                     }
1248.                     //arrayturismo guarda todas las
      ciudades calificadas;
1249.                     $arrayTurismo[$i]=$indice1;
1250.                     $i++;
1251.                 }
1252.                 $Distancia = (1/(1+sqrt($Valor)));
1253.
1254.                 //ALGORITMO DE PEARSON
1255.                 $InterUser=$this-
      >_turismoIntersect($Datos,$result);
1256.                 $InterUserNuevo=$this-
      >_turismoIntersect($Datos1,$result);
1257.                  $Distancia=$this-
      >_pearson($InterUser,$InterUserNuevo);
1258. //            echo '<pre>';
1259. //            print_r($Datos);
1260. //            print_r($turismoUser);
1261. //            print_r($TurismoUserNuevo);
1262. //            print_r($result);
1263. //            print_r($InterUser);
1264. //            print_r($InterUserNuevo);
1265. //            print_r($Distancia);
1266. //            echo '<pre>';exit;
1267.
1268.                   array_push($NuevoAllUser,
1269.                                       array(
1270.                                           "Usuario"=>
      $row['cod_usuario'],
1271.                                           "UsuarioN"=>
      $row['usu_nombre'],
1272.                                           "Ciudad" => $Datos,
1273.                                          // "Distanciae" =>
      1/(1+sqrt($Valor)),
1274.                                           "Distanciae" =>
      $Distancia,
1275.                                       ));
1276.             }
1277.         }
1278.
1279.
1280.         //ciudad unica elimina redundancia
1281.         //$CiudadNuevoUser ciudades calificadas por el
      nuevo usuario
1282.         $CiudadUnica=array_unique($arrayTurismo);
1283.         $CiudadAll=$CiudadUnica;
1284.         foreach($CiudadUnica as $indice => $row){
```

```
1285.                 foreach($TurismoUserNuevo as $row2){
1286.                     if($row==$row2){
1287.                      unset ($CiudadUnica[$indice]);
1288.                     }
1289.                 }
1290.           }
1291.           //ciudadunica solo contiene ciudades que no voto
      el nuevo usuario
1292.           //comparando ciudaddes
1293.           //$NuevoAllUser contiene las distancias de cada
      usuario con respecto al nuevo usuario
1294.           $Recomendacion=array();
1295.           foreach($CiudadUnica as $row){
1296.               $suma=0;
1297.               $SumDE=0;
1298.              foreach($NuevoAllUser as $row2){
1299.                  foreach($row2['Ciudad'] as $indice =>
      $valor){
1300.                  //usuario [i] califico el sitio $row
1301.                      if($indice == $row){
1302.                          $Multi=$row2['Distanciae']*$valor;
1303.                          $suma=$suma+$Multi;
1304.                          $SumDE=$SumDE +
      $row2['Distanciae'];
1305.                      }
1306.                  }
1307.
1308.              }
1309.              $Recomendacion[$row]=$suma/$SumDE;
1310.
1311.           }
1312.
1313.           return
      array($Recomendacion,$CiudadAll,$NuevoAllUser);
1314. //            echo '<pre>';
1315. //            print_r($CiudadUnica);
1316. //            print_r($NuevoAllUser);
1317. //
1318. //            print_r($DistanciaE);
1319. //           print_r($Recomendacion);
1320. //            echo '<pre>';exit;
1321.         }
1322.
1323.     }
1324. }
```

3. Controlador VerController

En esta sección se realizan todos los procesos para mostrar la información publicada por los moderadores y el administrador.

```
1. <?php

2. class VerController extends Zend_Controller_Action
3. {

4. public function init()
5. {
6. $this->view->baseUrl=$this->getRequest()->getBaseUrl();
```

```
7. //db-table
8. $this->db_bdturismo=new
   Application_Model_DbTable_Bdtturismo();
9. $this->db_turismoimagenes=new
   Application_Model_DbTable_Turismoimagenes();
10.   $this->db_catturismo=new
   Application_Model_DbTable_Catturismo();
11.   $this->db_usuariocalificacion=new
   Application_Model_DbTable_Usuariocalificacion();
12.   $this->db_usuarioturismo=new
   Application_Model_DbTable_Usuarioturismo();
13.   $this->db_usuario=new Application_Model_DbTable_Usuario();
14.   //activamos la opcion para sessiones en este controlador
15.   $this->session = new Zend_Session_Namespace('usuario');
16.   //existe usuario logueado
17.   if(isset($this->session->cod_usuario)){
18.   $this->view->codUsuario=$this->session->cod_usuario;
19.   $this->view->nombreUser=$this->session->usu_nombre;
20.   }
21.   //variable de galeria o normal
22.   $this->view->var="home home-four";
23.   $this->view->mapapost=1;
24.   }

25.   public function indexAction()
26.   {
27.   //noticia a buscar

28.   $id=$this->getRequest()->getParam("titulo");
29.   $temp=end(explode("-",$id));
30.   $idPost=str_replace('.html','',$temp);
31.   $sitioTuristico=$this->db_bdturismo->CodTurismo($idPost);
32.   $imagenes=$this->db_turismoimagenes->CodTurismo($idPost);

33.   if(isset($this->session->cod_usuario)){
34.   //recomendacion verificamos si existen datos en el
   algoritmo
35.   $Algoritmo = array();
36.   $Algoritmo=$this->_algoritmoAction();
37.   arsort($Algoritmo);
38.   $recomenDatos=array();
39.   $h=0;
40.   if(count($Algoritmo)>0){
41.   foreach($Algoritmo as $indice => $row){
42.   $recomenDatos[$h]=$this->db_bdturismo->CodTurismo($indice);

43.   $datos=$this->db_bdturismo->CodTurismo($indice);
44.   $j=0;
45.   foreach($datos as $row){
46.   $groupImg=$this->db_turismoimagenes-
   >CodTurismo($row['cod_turismo']);
47.   foreach($groupImg as $row2){
48.   $imagen=$row2['tuimag_ruta'];
49.   }
50.   $recomenDatos[$h][0]['img']=$imagen;
51.   $recomenDatos[$h][0]['titulo2']=str_replace(' ', '-',
   $this-
   >_sanear_string(strtolower(utf8_decode($row['turi_nombre']))))
   );
52.   $j++;
53.   }
```

```
54.   $h++;
55.   }

56.   $this->view->recomendacion=$recomenDatos;
57.   }

58.   //            echo '<pre>';
59.   //            print_r($Datos);
60.   //            print_r($turismoUser);
61.   //            print_r($TurismoUserNuevo);
62.   //            print_r($result);
63.   //            print_r($InterUser);
64.   //            print_r($InterUserNuevo);
65.   //            print_r($Distancia);
66.   //            echo '<pre>';exit;
67.   //verificamos si el sitio lo visitara el usuario
68.   $Planificacion=$this->db_usuarioturismo-
   >UsuarioVisita($this->session->cod_usuario,$idPost);

69.   if(count($Planificacion)>0){
70.   $this->view->planificacion=$Planificacion;
71.   }

72.   $Calificacion=$this->db_usuariocalificacion-
   >CalificacionUsuarioTurismo($this->session-
   >cod_usuario,$idPost);
73.   $this->view->calificacion=$Calificacion;
74.   $this->view->logueado=1;
75.   }

76.   $categorias=$this->db_catturismo->all();

77.   foreach($imagenes as $row2){
78.   $image=$row2['tuimag_ruta'];
79.   }

80.   $turismo=$this->db_bdturismo->TodoTurismo(50);
81.   $i=0;
82.   foreach($turismo as $row){
83.   $groupImg=$this->db_turismoimagenes-
   >CodTurismo($row['cod_turismo']);
84.   foreach($groupImg as $row2){
85.   $imagen=$row2['tuimag_ruta'];
86.   }
87.   $turismo[$i]['img']=$imagen;
88.   $turismo[$i]['titulo2']=str_replace(' ', '-', $this-
   >_sanear_string(strtolower(utf8_decode($row['turi_nombre']))))
   );
89.   $i++;
90.   }

91.   $url=str_replace(' ', '-', $this-
   >_sanear_string(strtolower(utf8_decode($sitioTuristico[0]['tu
   ri_nombre']))));
92.   $this->view->sitioturistico=$sitioTuristico;
93.   foreach($sitioTuristico as $row){
94.   $date2 = new Zend_Date($row['turi_fecha']);
95.   $nameFecha=$date2->get(Zend_Date::MONTH_NAME_SHORT).'
   '.$date2->get("d");
96.   $nameFecha2=$date2->get("Y");
97.   }
```

```
98.   $this->view->fecha=$nameFecha;
99.   $this->view->fecha2=$nameFecha2;
100.  $this->view->imagen=$image;
101.  $this->view->imagenes=$imagenes;
102.  $this->view->categorias=$categorias;
103.  $this->view->turismo=$turismo;

104.  $this->view->url=$url;


105.  }


106.  function _sanear_string($string)
107.  {

108.  $string = utf8_encode(trim($string));

109.  $string = str_replace(
110.  array('á', 'à', 'ä', 'â', 'ª', 'Á', 'À', 'Â', 'Ä'),
111.  array('a', 'a', 'a', 'a', 'a', 'A', 'A', 'A', 'A'),
112.  $string
113.  );

114.  $string = str_replace(
115.  array('é', 'è', 'ë', 'ê', 'É', 'È', 'Ê', 'Ë'),
116.  array('e', 'e', 'e', 'e', 'E', 'E', 'E', 'E'),
117.  $string
118.  );

119.  $string = str_replace(
120.  array('í', 'ì', 'ï', 'î', 'Í', 'Ì', 'Ï', 'Î'),
121.  array('i', 'i', 'i', 'i', 'I', 'I', 'I', 'I'),
122.  $string
123.  );

124.  $string = str_replace(
125.  array('ó', 'ò', 'ö', 'ô', 'Ó', 'Ò', 'Ö', 'Ô'),
126.  array('o', 'o', 'o', 'o', 'O', 'O', 'O', 'O'),
127.  $string
128.  );

129.  $string = str_replace(
130.  array('ú', 'ù', 'ü', 'û', 'Ú', 'Ù', 'Û', 'Ü'),
131.  array('u', 'u', 'u', 'u', 'U', 'U', 'U', 'U'),
132.  $string
133.  );

134.  $string = str_replace(
135.  array('ñ', 'Ñ', 'ç', 'Ç'),
136.  array('n', 'N', 'c', 'C',),
137.  $string
138.  );

139.  //Esta parte se encarga de eliminar cualquier caracter
      extraño
140.  $string = str_replace(
141.  array("\\", "¨", "º", "-", "~",
142.  "#", "@", "|", "!", "\"",
143.  "·", "$", "%", "&", "/",
144.  "(", ")", "?", "'", "¡",
```

```
145.  "¿", "[", "^", "`", "]",
146.  "+", "}", "{", "¨", "´",
147.  ">", "< ", ";", ",", ":",
148.  "."),
149.  '',
150.  $string
151.  );


152.  return $string;
153.  }
154.  //funcion que crea un lista
      array[informacionturistica]=calificacion
155.  public function _crearlista($CalificacionUser)
156.  {
157.  $Array=array();
158.  $i=0;
159.  foreach($CalificacionUser as $row2){
160.  $Array[$row2['cod_turismo']]=$row2['turi_calificacion'];
161.  }
162.  return $Array;
163.  }
164.  //funcion array almacena informacion turistica que califico
      el usuario
165.  public function _turismoUser($User)
166.  {
167.  $Array=array();
168.  $i=0;
169.  foreach($User as $row2){
170.  $Array[$i]=$row2['cod_turismo'];
171.  $i++;
172.  }
173.  return $Array;
174.  }
175.  //funcion array devuelve calificaciones de información
      turistica en comun de user y user nuevo
176.  public function _turismoIntersect($Data,$ArrayIntersect)
177.  {
178.  $Array=array();
179.  $i=0;
180.  foreach($ArrayIntersect as $Turismo){
181.  foreach($Data as $indice1 => $fil1){
182.  if($Turismo==$indice1){
183.  $Array[$i]=$fil1;
184.  $i++;
185.  }
186.  }
187.  }
188.  return $Array;
189.  }
190.  //funcion array almacena informacion turistica que califico
      el usuario
191.  public function _pearson($x,$y)
192.  {
193.  $length= count($x);
194.  $mean1=array_sum($x) / $length;
195.  $mean2=array_sum($y) / $length;

196.  $a=0;
197.  $b=0;
198.  $axb=0;
```

```
199.  $a2=0;
200.  $b2=0;

201.  for($i=0;$i<$length;$i++)
202.  {
203.  $a=$x[$i]-$mean1;
204.  $b=$y[$i]-$mean2;
205.  $axb=$axb+($a*$b);
206.  $a2=$a2+ pow($a,2);
207.  $b2=$b2+ pow($b,2);
208.  }
209.  $denominador=sqrt($a2*$b2);
210.  //si el denominador es igual a 0
211.  if($denominador!=0){
212.  $corr= $axb /$denominador;
213.  }else{
214.  $corr=0;
215.  }
216.  //si el coeficiente de correlacion es negativo
217.  if($corr<0){
218.  return 0;
219.  }else{
220.  return $corr;
221.  }

222.  }
223.  public function _algoritmoAction()
224.  {
225.  // action body
226.  //traemos todos los usuarios del sistema menos el usuario
     en session
227.  $AllUser=$this->db_usuario->TodoUsuario($this->session-
     >cod_usuario);
228.  //calificacion del usuario nuebo

229.  $Nuevo=$this->db_usuariocalificacion-
     >CalificacionUsuario($this->session->cod_usuario);
230.  //si exiten calificacion del nuevo usuario procedemos con
     el algoritmo
231.  if(count($Nuevo)>0){
232.  $Datos1=$this->_crearlista($Nuevo);
233.  $TurismoUserNuevo=$this->_turismoUser($Nuevo);


234.  $NuevoAllUser=array();
235.  $arrayTurismo=array();
236.  $i=0;
237.  foreach($AllUser as $row){
238.  $CalificacionUser=$this->db_usuariocalificacion-
     >CalificacionUsuario($row['cod_usuario']);
239.  $Datos=$this->_crearlista($CalificacionUser);
240.  $turismoUser =$this->_turismoUser($CalificacionUser);
241.  //intersecando que exista ciudades en comun entre las dos
     personas
242.  $result = array_intersect($turismoUser, $TurismoUserNuevo);

243.  // si existe alguna ciudad en comun
244.  if(count($result)>1){
245.  $Valor=0;
246.  //datos de la persona que entra en bucle
247.  //indice1 ciudad que voto fil1 el valor
```

```
248.  //datos1 usuario nuevo
249.  //indice2 ciudad que voto fil2 el valor

250.  //DISTANCIA EUCLIDEANA
251.  foreach ($Datos as $indice1 => $fil1){
252.  foreach ($Datos1 as $indice2 => $fil2){
253.  if($indice1==$indice2){
254.  $Valor=$Valor + pow(($fil2-$fil1), 2);
255.  }
256.  }
257.  //arrayturismo guarda todas las ciudades calificadas;
258.  $arrayTurismo[$i]=$indice1;
259.  $i++;
260.  }
261.  $Distancia = (1/(1+sqrt($Valor)));

262.  //ALGORITMO DE PEARSON
263.  $InterUser=$this->_turismoIntersect($Datos,$result);
264.  $InterUserNuevo=$this->_turismoIntersect($Datos1,$result);
265.  $Distancia=$this->_pearson($InterUser,$InterUserNuevo);
266.  //            echo '<pre>';
267.  //            print_r($Datos);
268.  //            print_r($turismoUser);
269.  //            print_r($TurismoUserNuevo);
270.  //            print_r($result);
271.  //            print_r($InterUser);
272.  //            print_r($InterUserNuevo);
273.  //            print_r($Distancia);
274.  //            echo '<pre>';exit;

275.  array_push($NuevoAllUser,
276.  array(
277.  "Usuario"=> $row['cod_usuario'],
278.  "Ciudad" => $Datos,
279.  // "Distanciae" => 1/(1+sqrt($Valor)),
280.  "Distanciae" => $Distancia,
281.  ));
282.  }
283.  }


284.  //ciudad unica elimina redundancia
285.  //$CiudadNuevoUser ciudades calificadas por el nuevo
      usuario
286.  $CiudadUnica=array_unique($arrayTurismo);
287.  foreach($CiudadUnica as $indice => $row){
288.  foreach($TurismoUserNuevo as $row2){
289.  if($row==$row2){
290.  unset ($CiudadUnica[$indice]);
291.  }
292.  }
293.  }
294.  //ciudadunica solo contiene ciudades que no voto el nuevo
      usuario
295.  //comparando ciudaddes
296.  //$NuevoAllUser contiene las distancias de cada usuario con
      respecto al nuevo usuario
297.  $Recomendacion=array();
298.  foreach($CiudadUnica as $row){
299.  $suma=0;
300.  $SumDE=0;
```

```
301.  foreach($NuevoAllUser as $row2){
302.  foreach($row2['Ciudad'] as $indice => $valor){
303.  //usuario [i] califico el sitio $row
304.  if($indice == $row){
305.  $Multi=$row2['Distanciae']*$valor;
306.  $suma=$suma+$Multi;
307.  $SumDE=$SumDE + $row2['Distanciae'];
308.  }
309.  }

310.  }
311.  $Recomendacion[$row]=$suma/$SumDE;

312.  }

313.  return $Recomendacion;
314.  //            echo '<pre>';
315.  //            print_r($CiudadUnica);
316.  //            print_r($NuevoAllUser);
317.  //
318.  //            print_r($DistanciaE);
319.  //           print_r($Recomendacion);
320.  //            echo '<pre>';exit;
321.  }

322.  }
323.  }
```

4. Controlador ErrorController

En este controlador viene incluido entre los controladores del framewoork Zend, en esta sección se encuentran todos los procesos para control de errores.

```
3.   <?php
4.
5.   class ErrorController extends Zend_Controller_Action
6.   {
7.
8.       public function errorAction()
9.       {
10.          $this->view->baseUrl=$this->getRequest()-
     >getBaseUrl();
11.          $this->_helper->layout()->disableLayout();
12.          $errors = $this->_getParam('error_handler');
13.
14.          if (!$errors || !$errors instanceof ArrayObject) {
15.              $this->view->message = 'You have reached the
     error page';
16.              return;
17.          }
18.
19.          switch ($errors->type) {
20.              case
     Zend_Controller_Plugin_ErrorHandler::EXCEPTION_NO_ROUTE:
21.              case
     Zend_Controller_Plugin_ErrorHandler::EXCEPTION_NO_CONTROLLER
     :
```

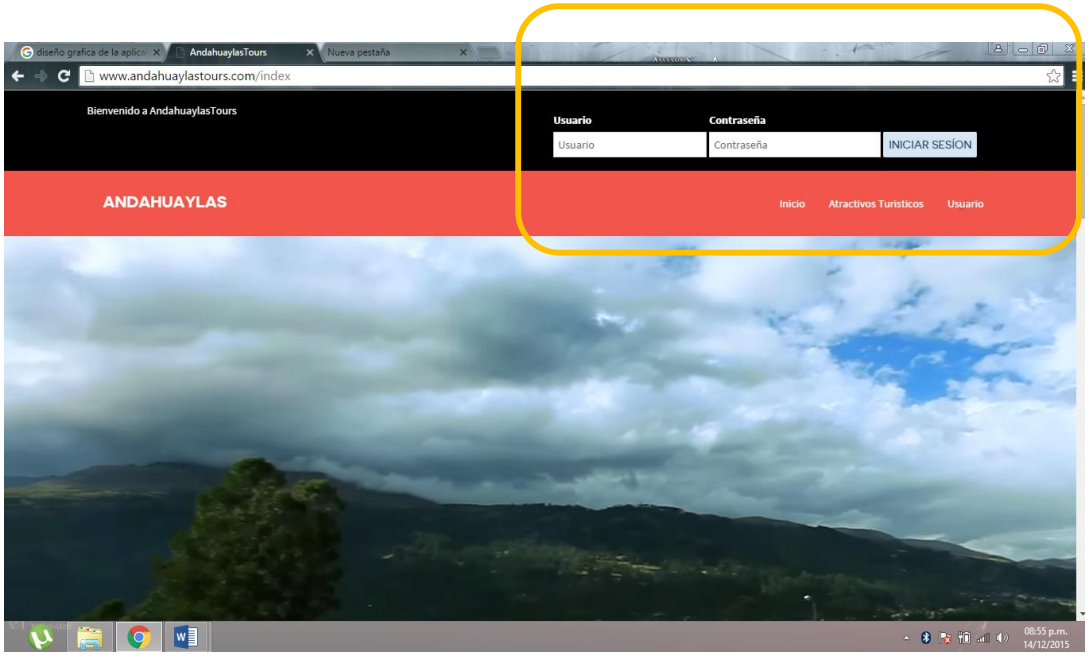
```
22.             case
    Zend_Controller_Plugin_ErrorHandler::EXCEPTION_NO_ACTION:
23.                 // 404 error -- controller or action not
    found
24.                 $this->getResponse()-
    >setHttpResponseCode(404);
25.                 $priority = Zend_Log::NOTICE;
26.                 $this->view->message = 'Page not found';
27.                 break;
28.             default:
29.                 // application error
30.                 $this->getResponse()-
    >setHttpResponseCode(500);
31.                 $priority = Zend_Log::CRIT;
32.                 $this->view->message = 'Application error';
33.                 break;
34.         }
35.
36.         // Log exception, if logger available
37.         if ($log = $this->getLog()) {
38.             $log->log($this->view->message, $priority,
    $errors->exception);
39.             $log->log('Request Parameters', $priority,
    $errors->request->getParams());
40.         }
41.
42.         // conditionally display exceptions
43.         if ($this->getInvokeArg('displayExceptions') ==
    true) {
44.             $this->view->exception = $errors->exception;
45.         }
46.
47.         $this->view->request   = $errors->request;
48.     }
49.
50.     public function getLog()
51.     {
52.         $bootstrap = $this->getInvokeArg('bootstrap');
53.         if (!$bootstrap->hasResource('Log')) {
54.             return false;
55.         }
56.         $log = $bootstrap->getResource('Log');
57.         return $log;
58.     }
59.
60.
61. }
```

ANEXOS 4

INTERFAZ GRAFICA DE LA APLICACIÓN

1. **Interfaz de inicio**

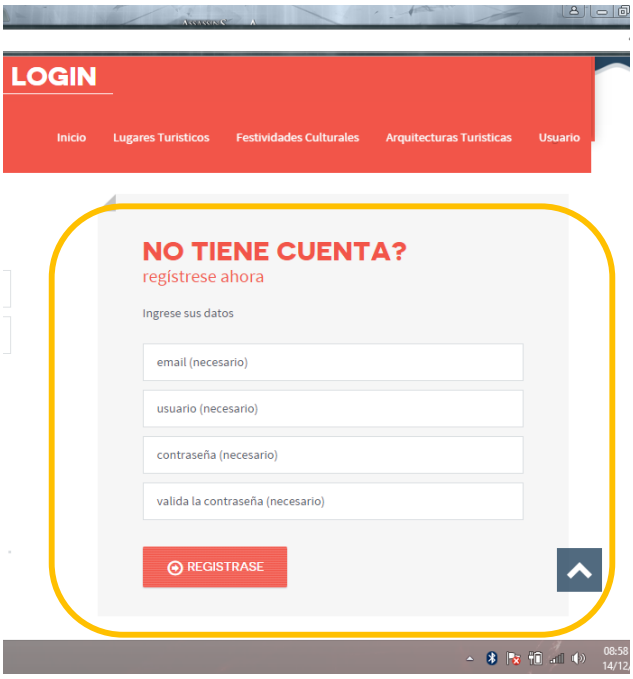
Aquí se muestra la interfaz gráfica que cualquier tipo de usuario (usuario, moderador y administrador), se observa en la parte marcada las opciones de inicio atractivos turísticos y usuario.



2. **Inicio de sesión**

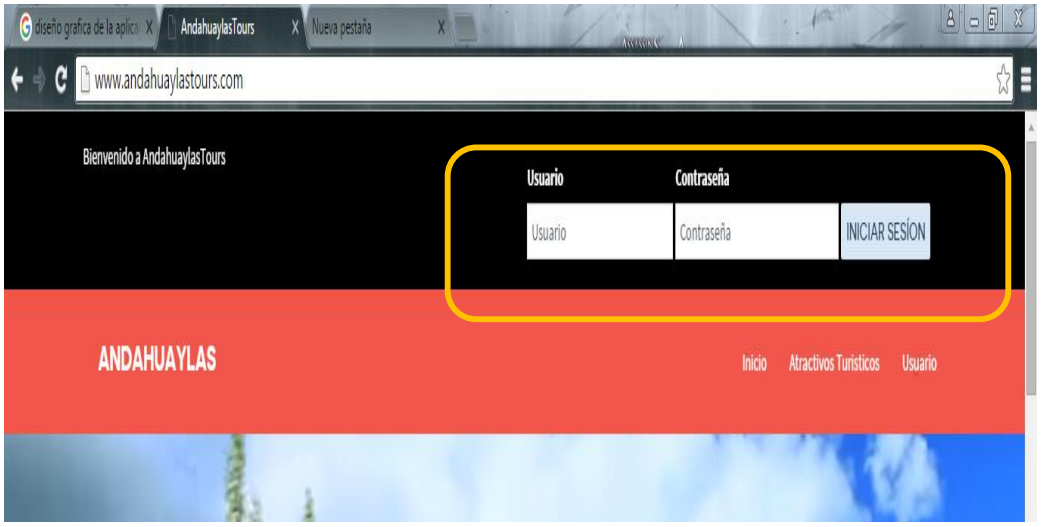
Si una persona desea interactuar con la aplicación primero deberá registrar una cuenta de usuario.

- Registro de cuenta de usuario.

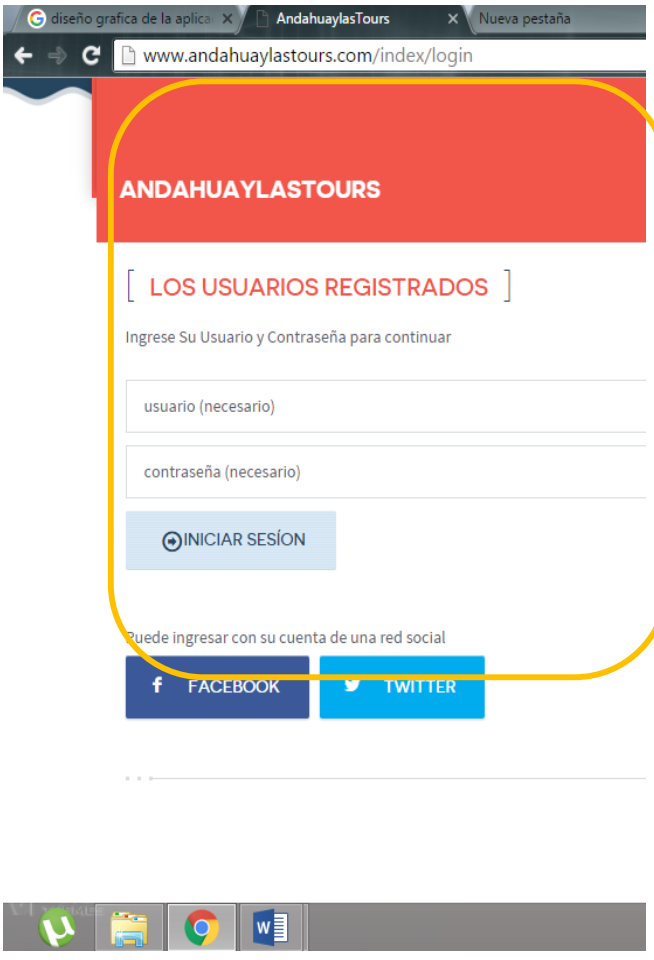


El inicio de sesión se realiza de 2 formas:

- Primera forma, es la que se ingresa de forma directa desde la página principal.



- Segunda forma, es la que se ingresa por la opción usuario.



3. Panel de control

En el panel de control se encuentran las diversas opciones a las que tienen acceso todos los usuarios según sean sus roles.

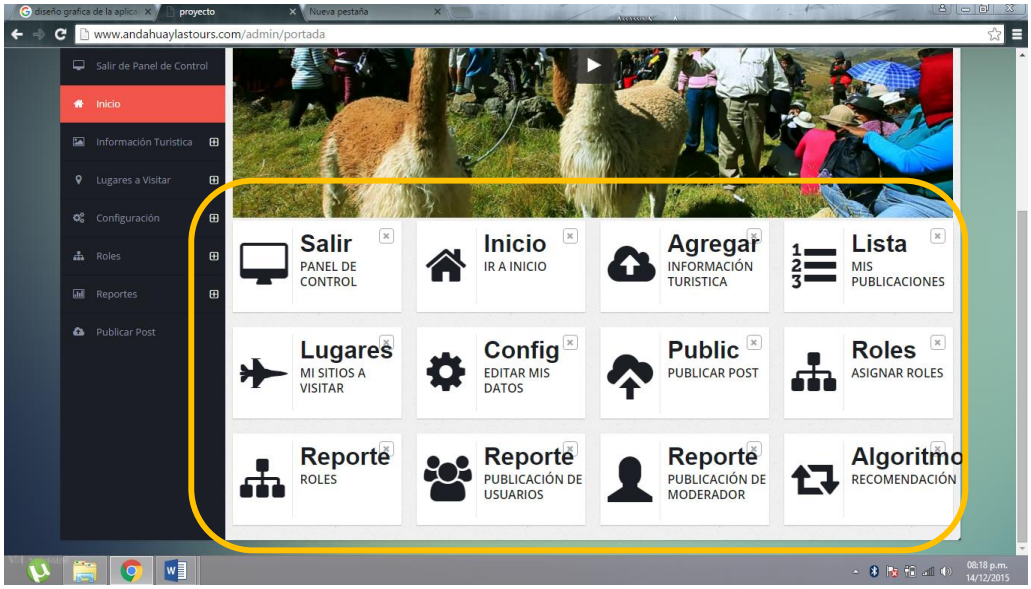
- En el siguiente grafico se muestra el entorno principal del panel de control algunas de las opciones son restringidas para los usuarios de acuerdo a sus roles.

Los usuarios con el rol de usuario, solo pueden acceder a opciones de agregar información, listar información, mis visitas, configuración de cuenta.

Los usuarios con el rol de moderador, acceden a todas las opciones del usuario con el rol usuario y a su vez tienen acceso a las publicaciones de los post en espera.

Los usuarios con el rol de administrador, pueden acceder

- En el siguiente grafico se muestran los accesos directos de las opciones que tiene cada usuario, en este caso se muestran las opciones del administrador.

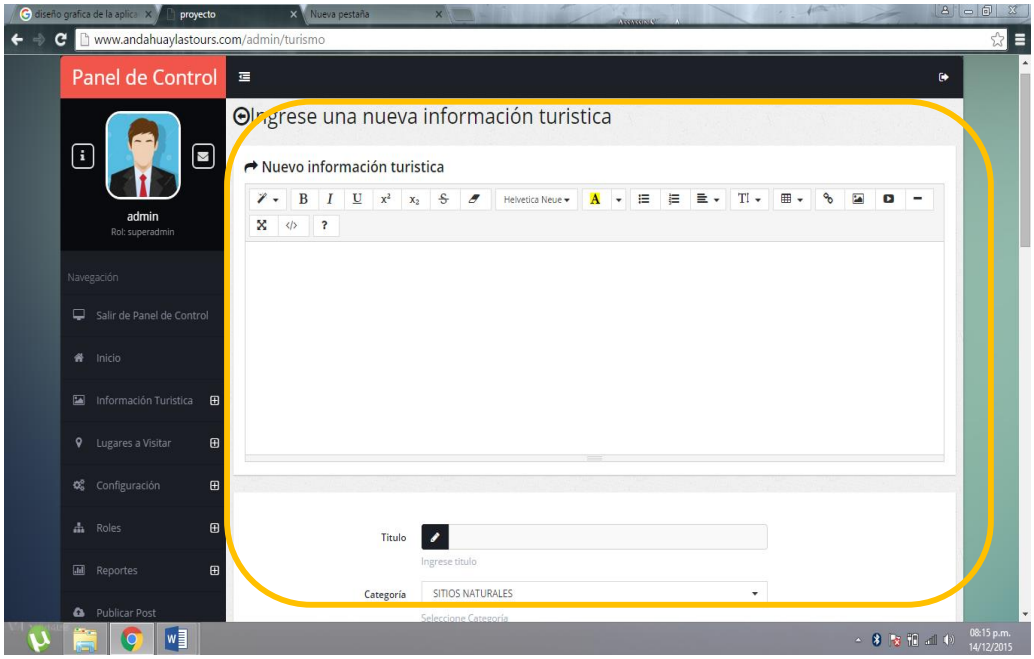


- Tenemos la primera opción que se denomina información turística, esta opción se le muestra a todos los usuarios registrados.

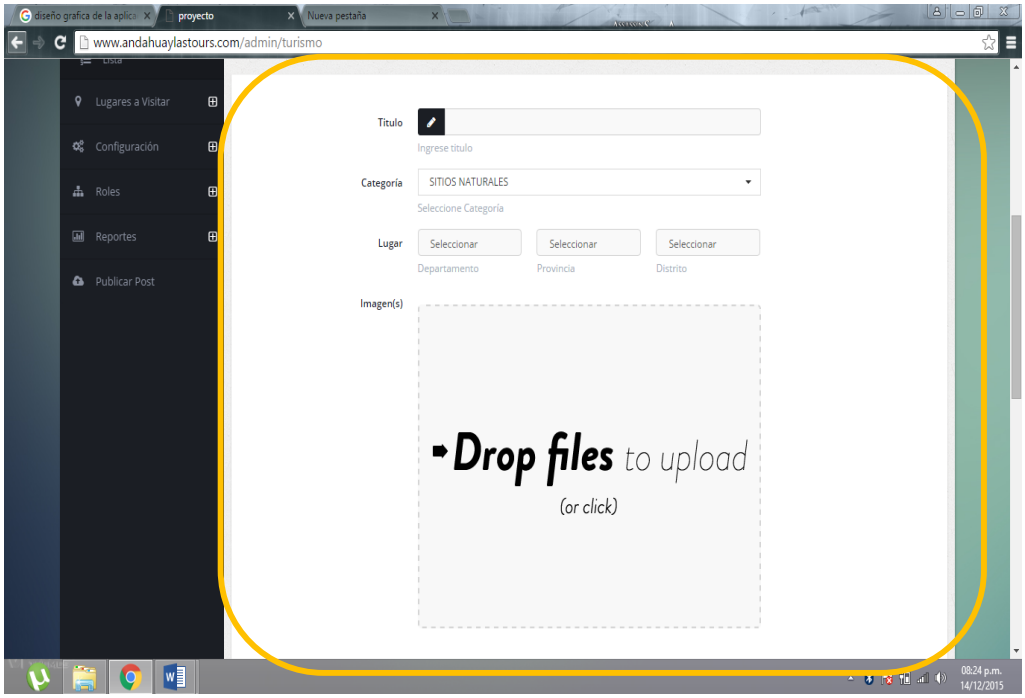
Dentro de la opción información turística existen 2 opciones más que son: Agregar información y listar la información agregada.

A continuación se muestran los datos que se deben ingresar en la opcion de agregar nueva información.

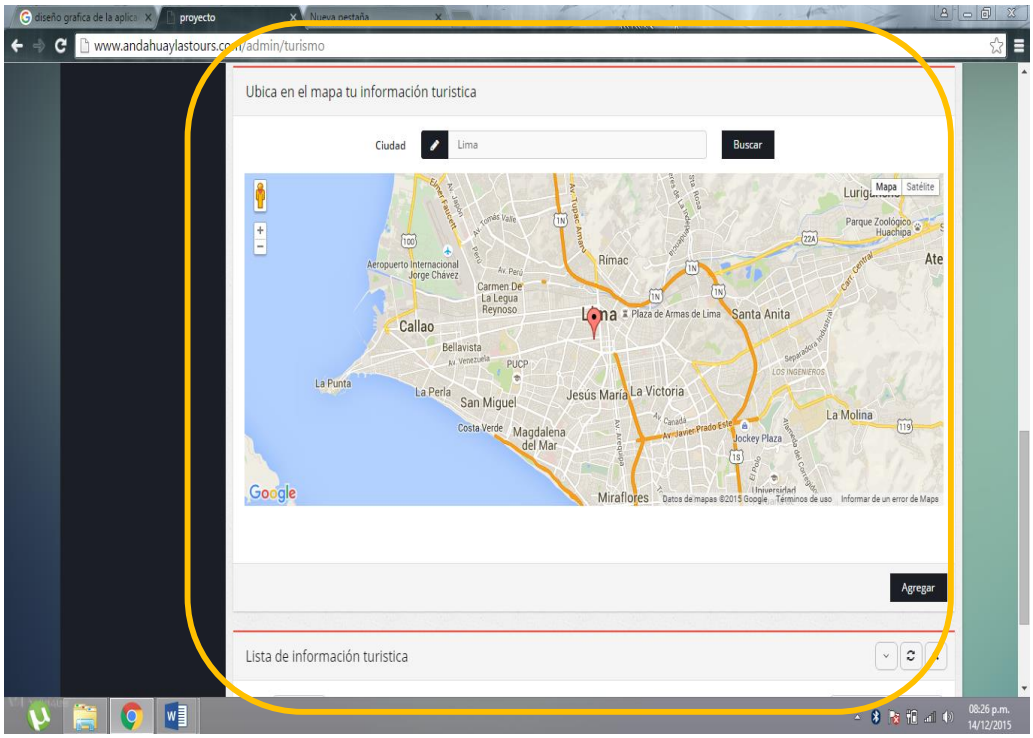
Nueva información turística: se ingresa todo el texto que describe el sitio o festividad turística.



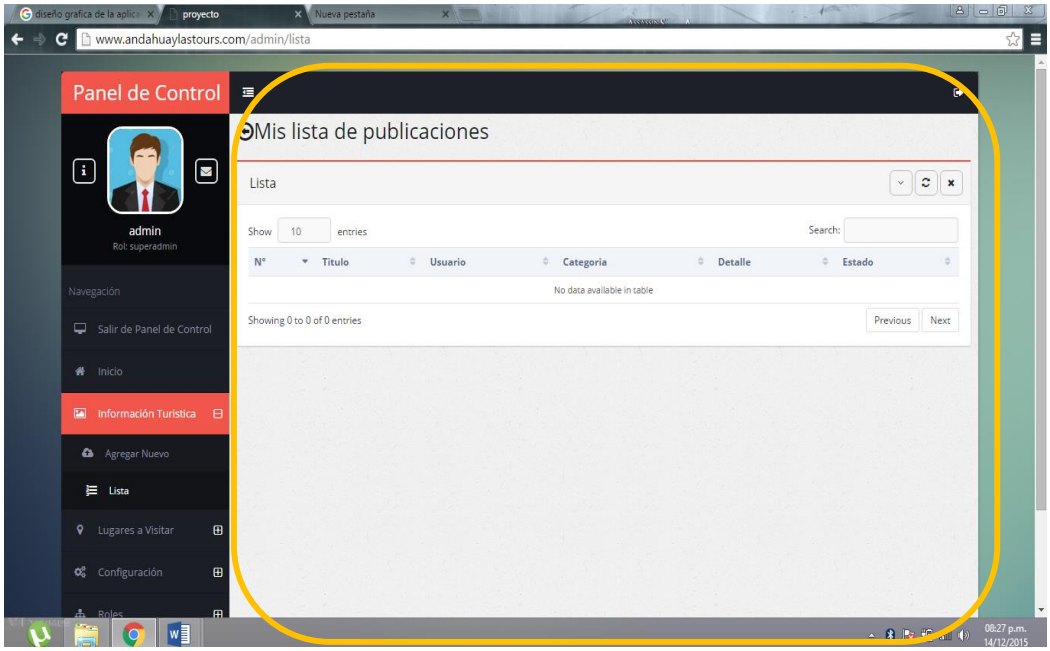
Título, categoría, lugar y fotos: esa información es muy importante para identificar el sitio o festividad turística.



Ubicación geográfica: esta opción ayuda al usuario que recibe la información final a visualizar la ubicación del sitio turístico o identificar donde se lleva acaba alguna festividad turística o folclórica.

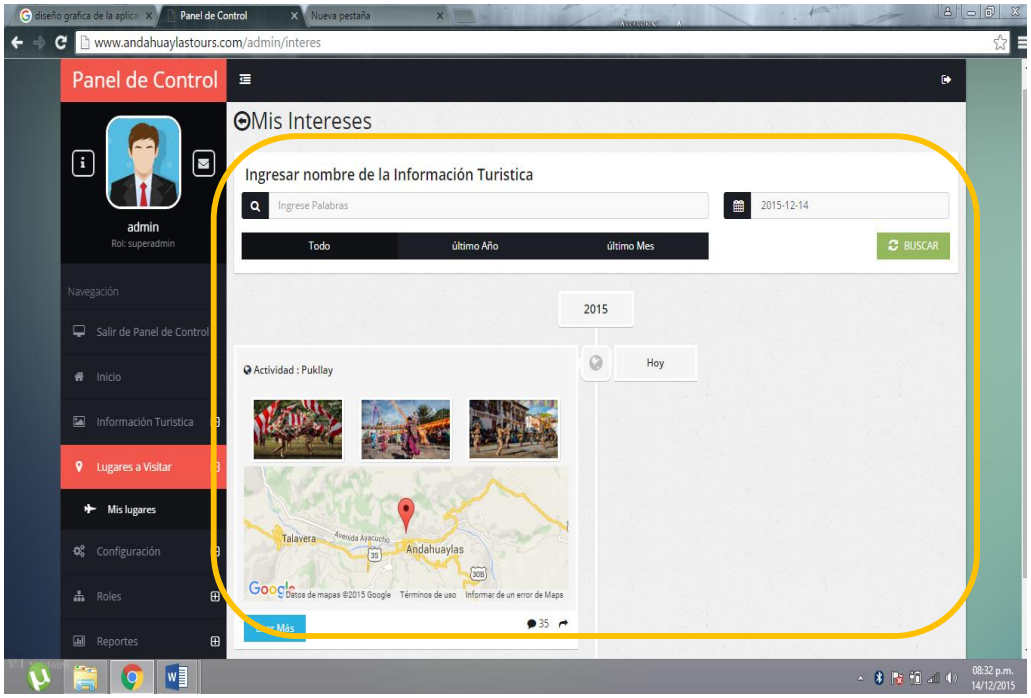


En esta parte se enlistan todas las publicaciones realizadas por el usuario que a inicia sesión.

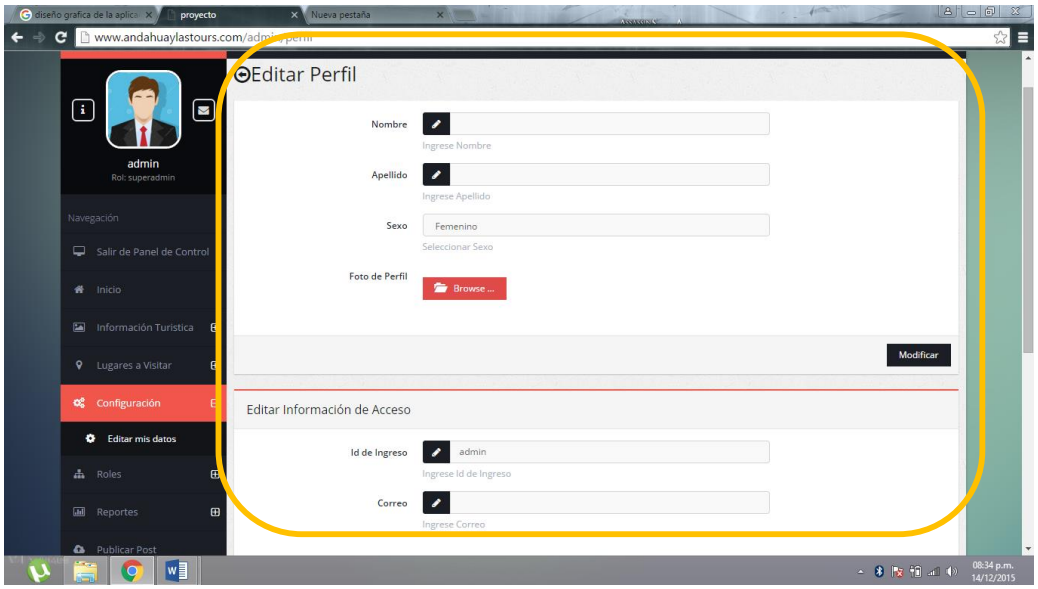


- Tenemos la segunda opción que se denomina lugares a visitar dentro del cual tenemos mis lugares, esta opción es una de las opciones que es accesible para todos los usuarios registrados sin excepción.

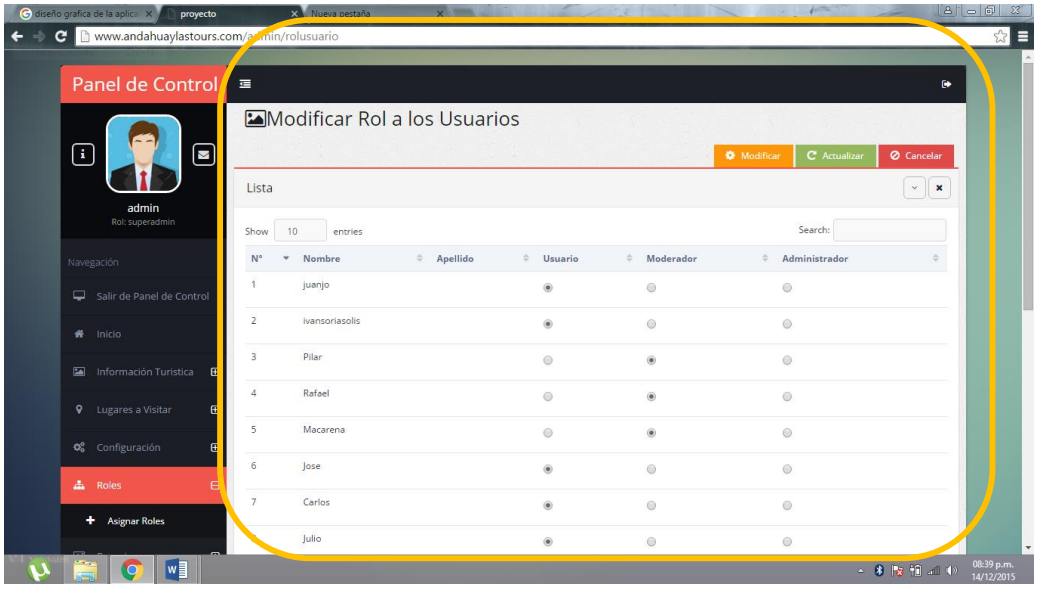
- De esta manera se visualiza una agenda de lugares a visitar para cada usuario.



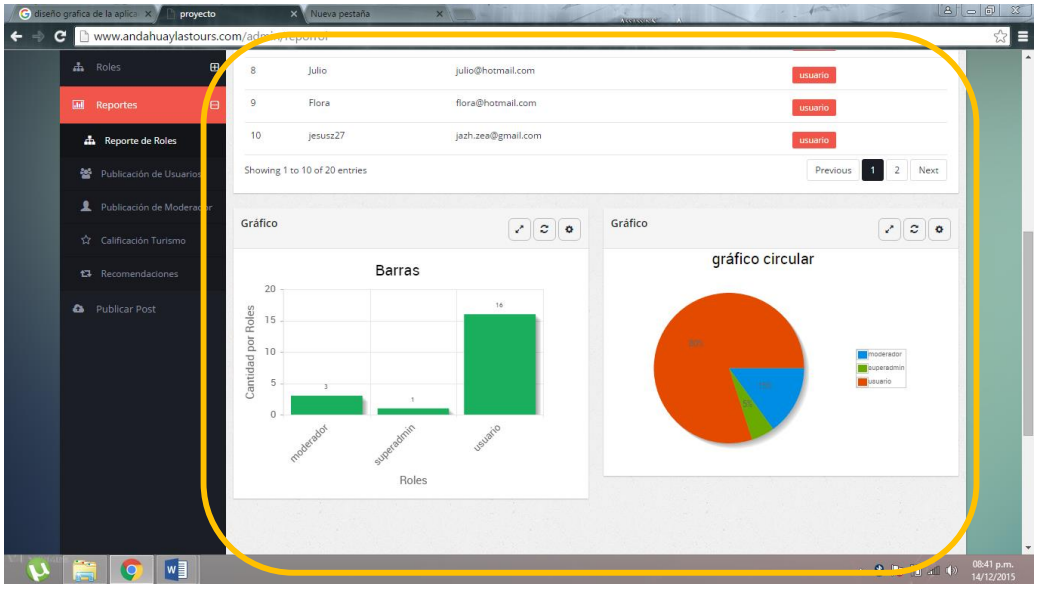
- Tenemos la tercera opción que se denomina configuración, que es básicamente el de editar los datos de los usuarios registrados, esta opción está disponible para todos los usuarios.

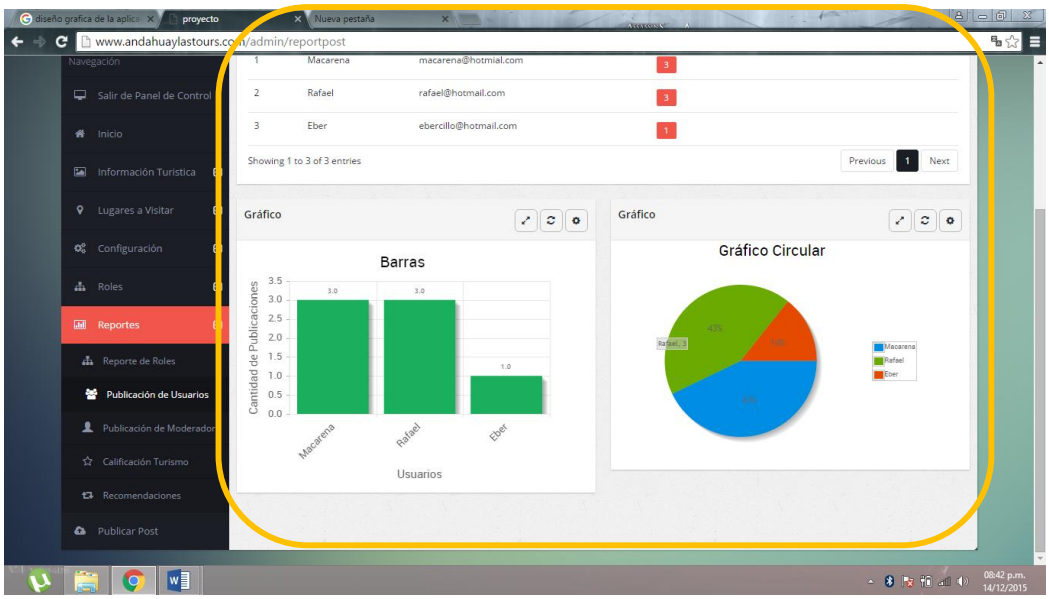


- Tenemos la cuarta opción que se denomina Roles: Esta opción es propia del Administrador, el usuario puede seleccionar que tipo de rol tendrá cada usuario registrado.

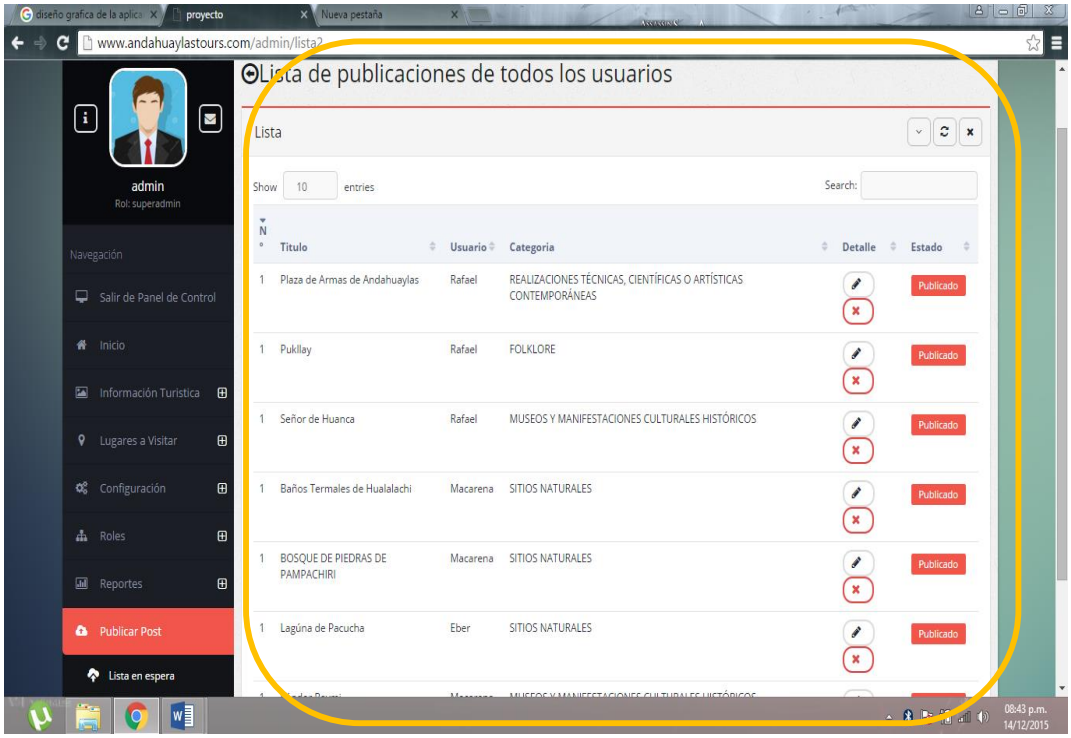


- Tenemos la quinta opción que se denomina reportes: Esta opción es propiamente del Administrado, los reportes permiten al administrador visualizar de forma selectiva la información que se desea obtener.

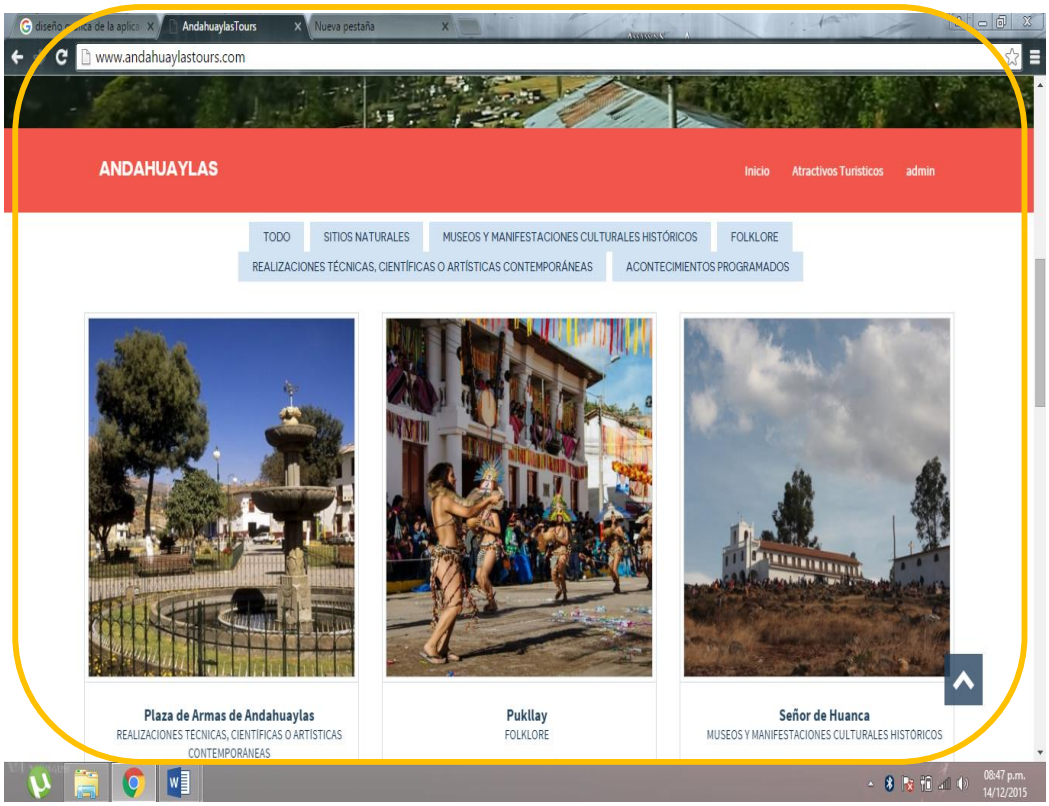




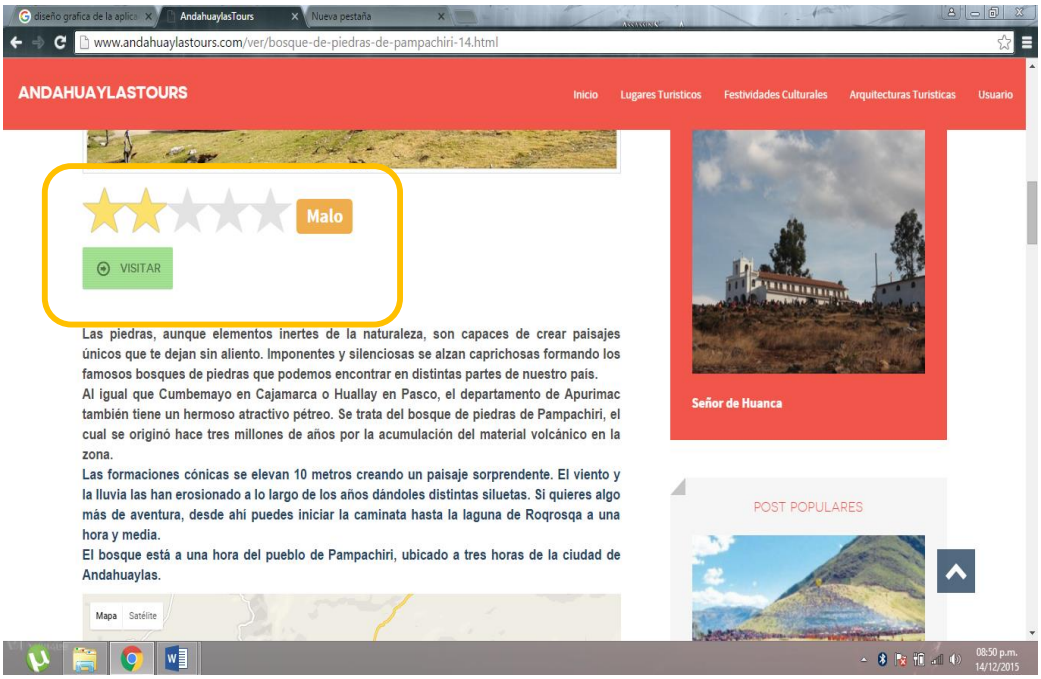
- Y como sexta opción tenemos a la opción Post: a esta opción solo tiene acceso los usuarios moderadores y administradores, se pretende seleccionar los post que pueden ser publicados o editar algunas publicaciones y después publicar la información.



4. **Entorno principal que se muestra a todo usuario registrado.**
   - Se muestran los sitios o festividades turísticas de la provincia, a su vez se aprecia que la información está siendo agrupada por categorías.

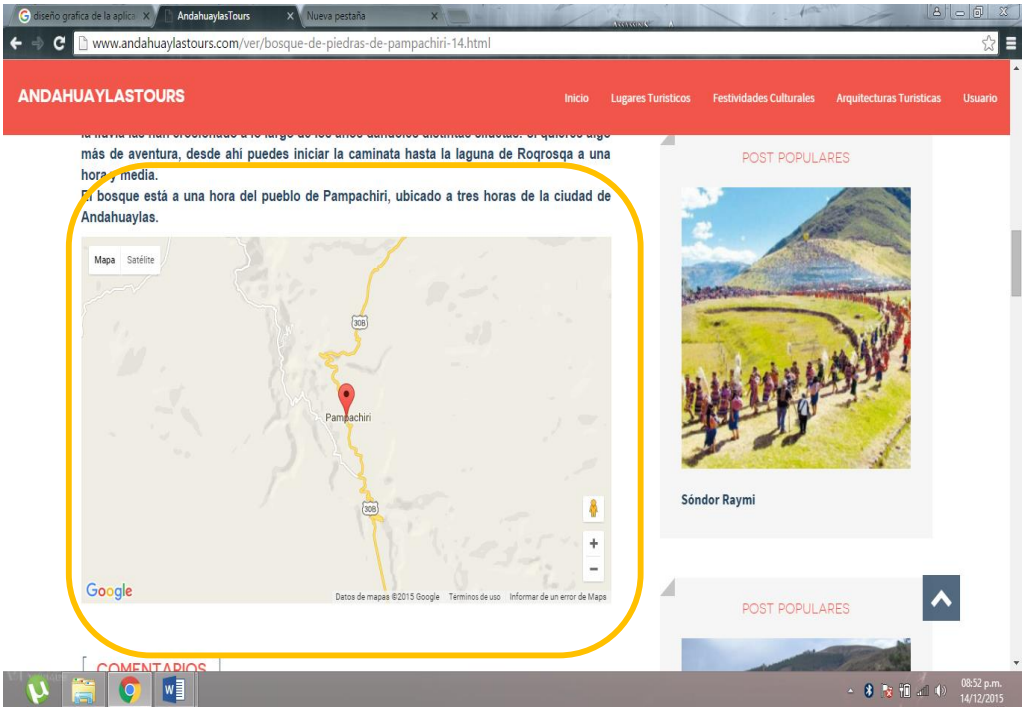


- Al hacer clic en cualquiera de los atractivos turísticos, se muestra la información sobre este, y a su vez se realizar la calificación del sitio y marcarlo como visitar o no vistar

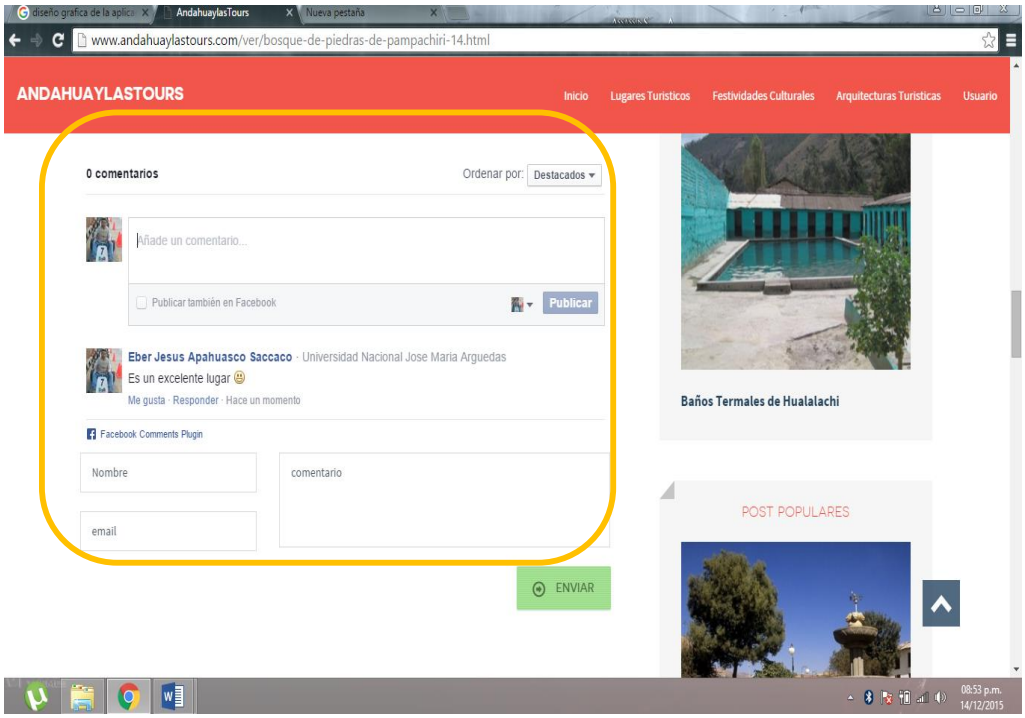


- Dentro de la información Contenida en los atractivos turísticos publicados, se encuentra la ubicación geográfica en donde se encuentra

en caso de que sea un sitio turístico o en que lugar se realiza si es el caso de una festividad: religiosa, folclórica o cultural.



- Tenemos un plugin incorporado que permite realizar comentarios de usuarios de Facebook sobre nuestros sitios turísticos.

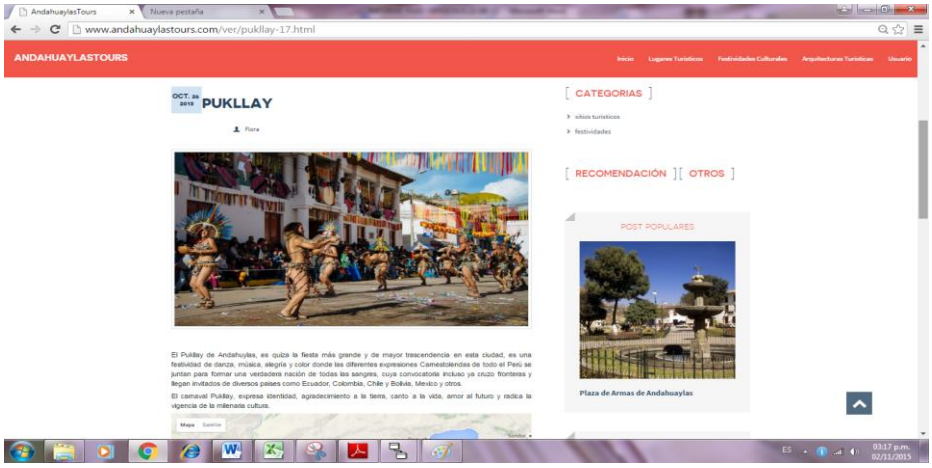


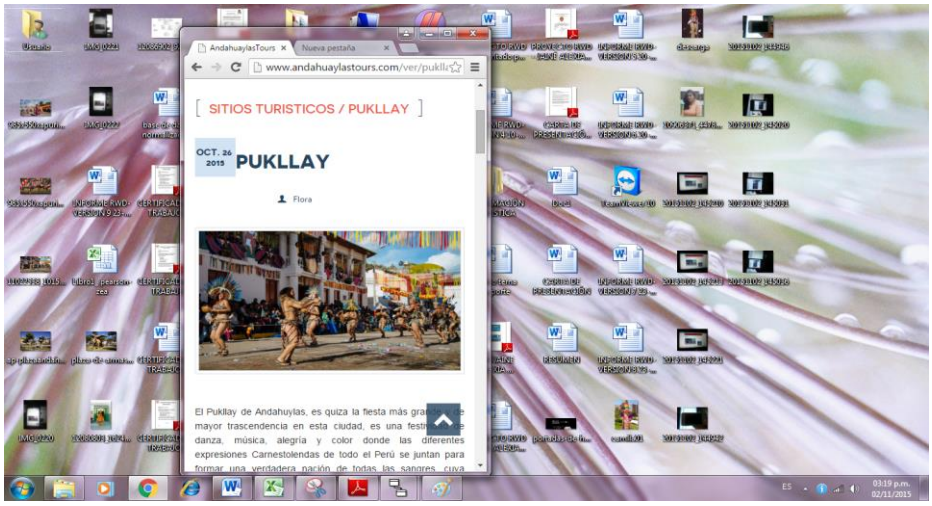
- Finalmente los módulos de recomendación se encuentra al lado derecho de la página.

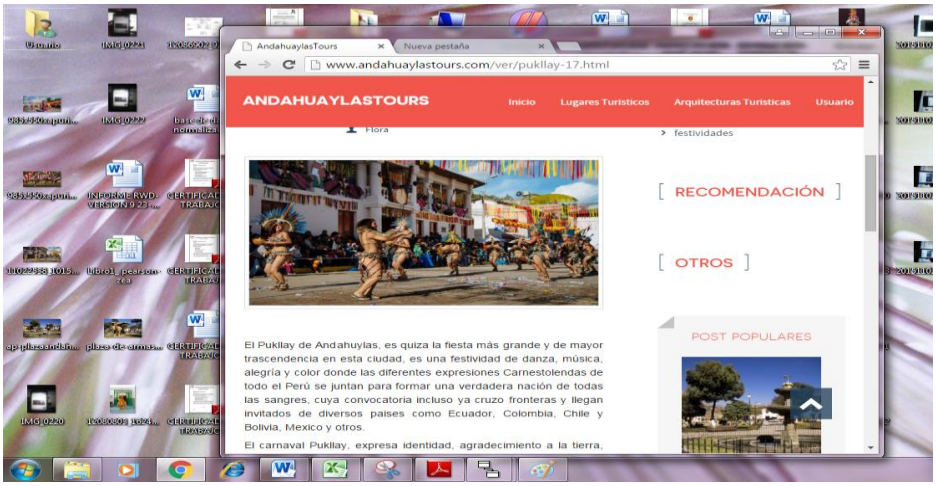
ANEXOS 5

DISEÑO WEB ADAPTABLE DEL SITIO WEB

- En el siguiente entorno se muestra que la página no pierde sus propiedades si se cambia en tamaño de la ventana de navegación.

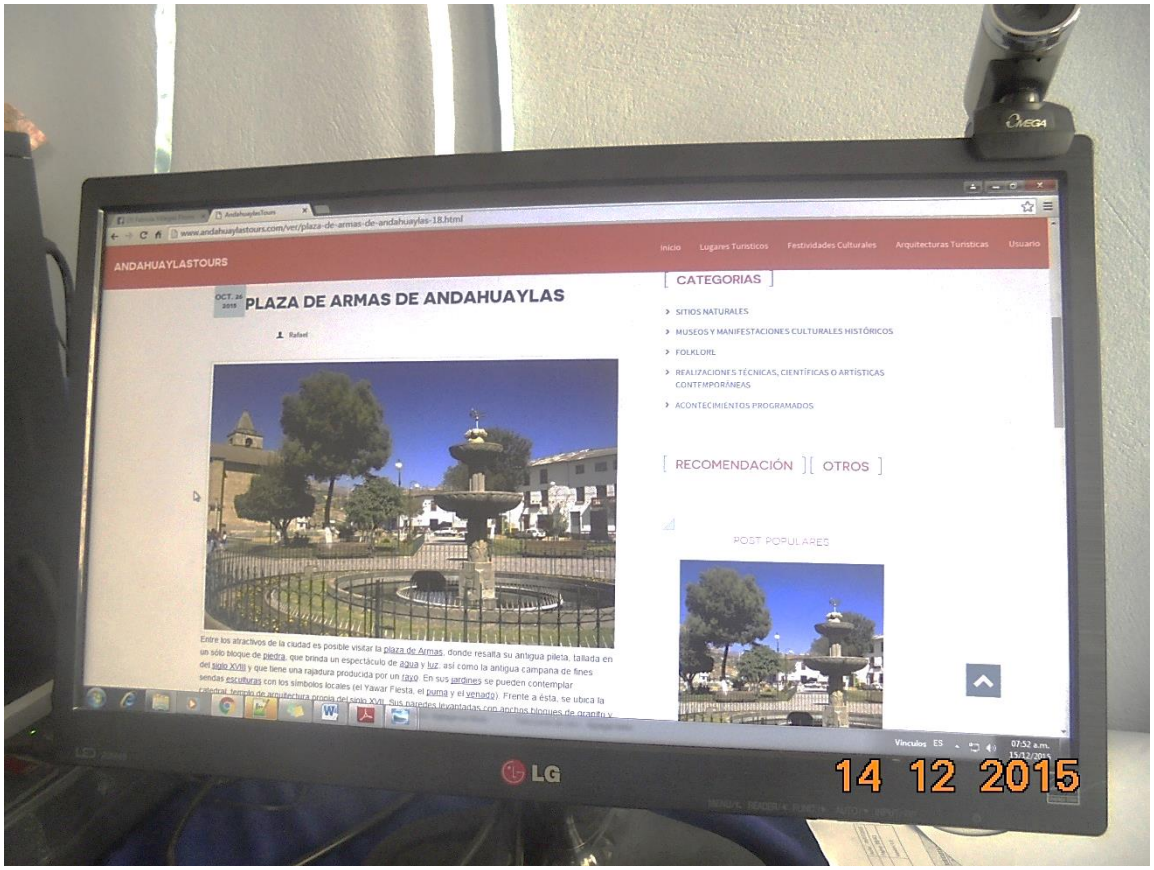






- Algunos quipos más en los que la aplicación demostró su portabilidad.

-



ANEXOS 6

HISTORIAS DE USUARIO

- Historia de usuario Numero 01, restricciones de la tabla usuario.

HISTORIAL DE USUARIO	
Número: 01	**Usuario: Administrador**
Nombre de Historia: Restricciones de la tabla usuario	
Prioridad: Alta	**Riesgo en desarrollo:** Alto
Programador responsable: Jainé Alexia Monzón Flores	
Descripción: Sobre los atributos en la tabla Usuario, el administrador solicito que la información solicitada, como son la de edad, sexo, teléfono y sexo no sean considerados como campos obligatorios.	
Observaciones: En este caso se separara la información en dos tablas diferentes pero estrechamente relacionas, para así evitar que haya espacios en blanco que se guarden en la base de datos.	

- Historia de usuario Numero 02, Mostrar sugerencias en la programación de visitas.

HISTORIAL DE USUARIO	
Número: 02	**Usuario: Moderador**
Nombre de Historia: Mostrar sugerencias en la programación de visitas.	
Prioridad: Media	**Riesgo en desarrollo:** Nulo
Programador responsable: Jainé Alexia Monzón Flores	
Descripción: En el módulo de programación de vistas a sitios turísticos o a eventos culturales, el moderado solicito se incorpore en la parte derecha un motor de sugerencias en donde el usuario pueda programar su visita sin tener que buscarla. Caso que no fue considerado, debido a que este módulo de recomendaciones se encontraba de forma general.	
Observaciones: Es correcto considera el motor de sugerencias dentro de la programación de visitas, debido a que ciertamente facilitara programar visitas de acuerdo a sus posibles preferencias.	

- Historia de usuario Número 03, Reportes estadísticos actualizables.

HISTORIAL DE USUARIO	
Número: 03	**Usuario: Cliente Administrador**
Nombre de Historia: Reportes estadísticos actualizables.	
Prioridad: Media	**Riesgo en desarrollo:** Bajo
Programador responsable: Henry Ortiz Centeno	
Descripción: El usuario solicito que se empleen los Dashword para mostrar los reportes estadísticos, debido a que emplea una forma dinámica de mostrar la información.	
Observaciones: Se realizar pruebas para verificar que lo que solicito el usuario sea una idea aceptable o considerar convencer al usuario de que la herramienta que actualmente estamos usando es la adecuada.	