UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

SEDE VIÑA DEL MAR – JOSÉ MIGUEL CARRERA

RED SOCIAL ORIENTADA A LA PUBLICACIÓN DE PRODUCTOS Y SERVICIOS
"Feria en Línea"

Trabajo de titulación para optar al título de Técnico Universitario en INFORMÁTICA

Alumnos:

Juan Moreno Galleguillos

Patrick Carrizo Gradón

Profesor Guía:

Oscar Carrasco Vera

2018

RESUMEN

Keywords: Red Social, Pagina Web, Compra, Venta, Comunidad, Facebook, MercadoLibre, Yapo.cl

El presente trabajo tiene por nombre Red Social Orientada a la Publicación de Productos y Servicios “Feria en Línea”.

El trueque fue uno de los inicios del comercio para obtener algún producto de necesidad o deseo, entregando algo de similar valor para la contraparte. Esto ha ido evolucionando con el paso del tiempo y el desarrollo de nuevas tecnologías por lo que en la actualidad, existen sitios webs que se dedican a esta actividad. No obstante, estos no satisfacen completamente la necesidad de los usuarios. Es por ello que se idea un proyecto que busca contener –y mejorar– las funciones y características más fuertes de cada una de ellas.

La finalidad es implementar una página web que permita la publicación de productos y servicios en Chile, de manera que esta web tenga por objetivo principal crear y facilitar una instancia de compra y/o venta entre uno o más usuarios y que éstos, con el paso del tiempo, no pierdan el contacto. Esto, con el fin de crear una comunidad virtual que los represente con base a sus intereses.

Para exponerlo de manera más clara, este trabajo se ha estructurado en tres capítulos:

CAPÍTULO 1: En este capítulo, se realiza una descripción de la situación del mercado en Chile, con el propósito de identificar los problemas y analizar la forma de solucionarlo, además de entregar los beneficios del sistema, entradas y salidas del mismo, exponiendo todas las entidades que se utilizan. Por último, se adjunta el modelo de datos del sistema.

CAPÍTULO 2: Una vez definido los objetivos a alcanzar y concebido el modelo de datos que permite solucionar los problemas detectados, se procede al diseño de la Base de datos en donde quedarán registrados los datos y los requisitos de hardware y software necesarios para que la aplicación pueda ejecutarse satisfactoriamente.

CAPÍTULO 3: En este capítulo se describen las interfaces de usuarios y los procesos que se deben ejecutar para el registro de datos, su procesamiento y obtención de resultados. El diseño se basa en los propios gustos de los autores, por lo que no se tiene restricciones al respecto.

Finalmente, se adjuntan anexos sobre el código, se exponen las conclusiones del trabajo, se evalúan sus resultados y se proponen mejoras a aplicar.

ÍNDICE

INTRODUCCIÓN.....1

1. ASPECTOS RELEVANTES PARA EL DISEÑO LÓGICO. 3

1.1. DESCRIPCIÓN DE LA ORGANIZACIÓN.....3

1.2. DESCRIPCIÓN DE LA SITUACIÓN ACTUAL.....3

1.3. PROBLEMAS DETECTADOS 6

1.4. DESCRIPCIÓN DEL SISTEMA PROPUESTO. 8

1.4.1. Objetivo General. 8

1.4.2. Objetivos Específicos. 8

1.4.3. Beneficios del sistema. 8

1.4.4. Descripción General de la Solución Propuesta. 9

1.4.5. Estructura funcional del sistema. 11

1.4.6. Descripción de fórmulas y cálculos empleados. 14

1.4.7. Información Manejada por el Sistema. 14

1.4.8. Entidades de Información.....16

1.4.9. Estructura de códigos. 17

1.4.10. Condicionantes de diseño (tiempo de respuesta, periodicidad, otros).....19

1.4.11. Modelo de datos.....20

2. MEDIO AMBIENTE COMPUTACIONAL Y DISEÑO FÍSICO. 22

2.1. AMBIENTE COMPUTACIONAL 22

2.1.1. Configuración del sistema. 22

2.1.2. Software utilizado. 23

2.2. LISTADO DE TABLAS 25

2.3. DESCRIPCIÓN DE TABLAS 26

2.3.1. Tabla Usuario. 27

2.3.2. Tabla Publicacion. 28

2.3.3. Tabla Ciudad. 29

2.3.4. Tabla Región. 29

2.3.5. Tabla Registro_transaccion 30

2.3.6. Tabla Comentario. 31

2.3.7. Tabla Usuario_Seguidor. 31
2.3.8. Tabla Preferencia. 32
2.3.9. Tabla Mensaje. 32
2.3.10. Tabla SubCategoria. 33
2.3.11. Tabla Categoría. 33
2.3.12. Tabla Imagen. 34
2.3.13. Tabla Amigo. 34
2.3.14. Tabla Administrador. 35
2.3.15. Tabla Interés. 35
2.3.16. Tabla Login. 36
2.3.17. Tabla Notificación. 36

3. DESCRIPCION DE PROGRAMAS. 38

3.1. DIAGRAMA DE MENÚS. 38
3.2. DIAGRAMA MODULAR. 39
3.3. LAYOUT DEL SITIO WEB 40
3.4. ESPECIFICACIÓN DE PROGRAMAS 41
3.5. DESCRIPCIÓN DETALLADA DE LOS PROGRAMA SELECCIONADOS 42
3.5.1. Iniciar Sesión. 42
3.5.2. Crear Publicación. 44
3.5.3. Comprar Producto/Servicio. 46
3.5.4. Mensajes. 48
3.5.5. Usuarios. 50

CONCLUSIONES 52

BIBLIOGRAFÍA 53

ANEXO 54

ÍNDICE DE FIGURAS

Figura 1-1. Modelo de Datos.....20
Figura 3-1. Diagrama de Menús.....38
Figura 3-2. Diagrama Modular.....39
Figura 3-3. Diseño Layout 40
Figura 3-4. Diagrama de Bloque Iniciar Sesión.....42
Figura 3-5. Pantalla Iniciar Sesión 43
Figura 3-6. Diagrama de bloque crear publicación. 44
Figura 3-7. Pantalla Crear Publicación. 45
Figura 3-8. Diagrama de bloque Transacción. 46
Figura 3-9. Pantalla Publicación. 47
Figura 3-10. Pantalla Venta.....48
Figura 3-11. Diagrama de bloque Mensajes.....48
Figura 3-12. Diseño pantalla Chat.....49
Figura 3-13. Diseño de pantalla Listado usuarios 51

ÍNDICE DE TABLAS

Tabla 1-1. Comparación de 3 plataformas. 5
Tabla 2-1. Descripción de Tabla: USUARIO 27
Tabla 2-2. Descripción de Tabla: PUBLICACIÓN 28
Tabla 2-3. Descripción de Tabla: CIUDAD.....29
Tabla 2-4. Descripción de Tabla: REGION 29
Tabla 2-5. Descripción de Tabla: REGISTRO_VENTA 30
Tabla 2-6. Descripción de Tabla: COMENTARIO 31
Tabla 2-7. Descripción de Tabla: USUARIO_SEGUIDOR 31
Tabla 2-8. Descripción de Tabla: PREFERENCIA 32
Tabla 2-9. Descripción de Tabla: Mensaje.....32
Tabla 2-10. Descripción de Tabla: SUBCATEGORIA 33
Tabla 2-11. Descripción de Tabla: CATEGORIA 33
Tabla 2-12. Descripción de Tabla: IMAGEN 34
Tabla 2-13. Descripción de Tabla: AMIGO.....34
Tabla 2-14. Descripción de Tabla: ADMINISTRADOR.....35
Tabla 2-15. Descripción de Tabla: INTERES.....35
Tabla 2-16. Descripción de Tabla: LOGIN.....36
Tabla 2-17. Descripción de Tabla: NOTIFICACION.....36
Tabla 3-1. Listado de programas.....41

INTRODUCCIÓN

En la actualidad, las páginas web de compra y venta son uno de los puntos fuertes de Internet, pues gracias a éstas, las personas (usuarios) logran tener acceso a objetos y comodidad a precios más razonables que los que ofrece el retail actual, sea mediante pequeñas empresas, o terceros, y sean estos nuevos o usados. Sin embargo no todas las páginas web cuentan con todo lo que el usuario quiere para hacerlas más atractivas y cómodas para éstos.

Tras la realización de este trabajo de título, se busca encontrar solución a estos requerimientos y lograr obtener una solución más acorde a los que los usuarios buscan, además de la detección de errores, otorgando sus respectivas soluciones, con el fin de mejorar la experiencia de navegación del usuario.

Además de todo lo ya nombrado, un punto importante que se desea destacar antes de comenzar, es el incentivo de sembrar en cada usuario que ingrese y sea parte del software, el cual se basa en una creación de una comunidad a la cual le interesa el “rubro” de la compra/venta, para así poder formar lazos con personas desconocidas, pero que compartan los mismos intereses el uno con el otro, con el fin de mejorar las instancias de compra y venta.

CAPÍTULO I. ASPECTOS RELEVANTES DEL DISEÑO LÓGICO

1. ASPECTOS RELEVANTES PARA EL DISEÑO LÓGICO.

1.1. DESCRIPCIÓN DE LA ORGANIZACIÓN

El presente trabajo de título consiste en un proyecto de autoría propia y que no está destinada para ser empleada por empresa alguna. Por lo tanto, el proyecto se basa en requerimientos definidos por los mismos autores y en base a esto de diseña e implementa una solución informática.

1.2. DESCRIPCIÓN DE LA SITUACIÓN ACTUAL

En la actualidad, existen 3 sitios web en Chile que sirven de intermediarios entre personas que desean vender o comprar productos y servicios por internet: Mercadolibre.cl, Yapo.cl y, aunque no diseñado con este propósito, Facebook.com.

Mercadolibre.cl fue fundada el año 1999 en Argentina, y tiene presencia en gran parte de Latinoamérica. Este sitio web funciona a través del registro de los usuarios, los cuales, tienen la facultad de publicar artículos para su venta o comprar productos publicados por otros usuarios.

El funcionamiento de este sistema se explica en los siguientes pasos:

1. Determina el producto que desea comprar.
2. Tiene la opción de realizar comentarios u consultas, en recuadro de diálogo específico para el producto.
3. Se realiza la compra al pulsar el botón “Comprar”.
4. Los datos del vendedor son enviados al comprador y viceversa, para hacer efectiva la compra.
5. Luego de efectuada la compra, los usuarios se califican mutuamente.
6. La publicación es dada de baja automáticamente por “Mercado Libre” luego de confirmada la venta.

Por su parte, el sitio web Yapo.cl recientemente adherido a la competencia, pertenece a un grupo de inversión internacional de origen noruego llamado Schibsted, el cual tiene presencia mundial a través diversos diarios y sitios web.

El funcionamiento de Yapo.cl se puede simplificar en los siguientes pasos:

1. Determina el producto que desea comprar.
2. Los datos del vendedor son de carácter público, por lo tanto, se puede contactar directamente con él (teléfono o mail) o mandar un mensaje a través de la página.
3. Luego, la compra se realiza a través de la coordinación de las partes.
4. El vendedor tiene la posibilidad de dar de baja la publicación personalmente o el sitio web lo hace automáticamente expirado un plazo.

Por último, Facebook.com es un mundialmente conocido sitio web de redes sociales creado en Estados Unidos el año 2004. Facebook.com está orientado a entregar una comunicación fluida entre sus usuarios, permitiendo a éstos compartir cualquier tipo de contenido (excepto los prohibidos por ley) de una manera ágil y sencilla.

Producto de la flexibilidad que entrega Facebook.com, han surgido (en el caso chileno), las llamadas “Ferias de las Pulgas” en donde se conforman comunidades de usuarios que están en busca de un espacio para poder visibilizar sus ofertas de artículos y servicios. Estos espacios dentro de Facebook.com han surgido espontáneamente, producto de la necesidad de un lugar en donde poder realizar este tipo de actividades.

El funcionamiento a la hora de crear la conexión entre los usuarios es el siguiente:

1. El usuario busca algún grupo de Facebook, en donde se realicen actividades de compra y venta de artículos y servicios.
2. Luego, se procede a buscar dentro de las publicaciones el artículo o servicio que se desea adquirir.
3. Por último, se envía un mensaje privado o público al vendedor para concretar la venta.

Las funcionalidades detectadas en cada uno de los sitios web son las siguientes:

Tabla 1-1. Comparación de las funcionalidades de 3 plataformas.

Funcionalidad	MercadoLibre	Yapo.cl	Facebook.com
Publicación de productos			
Perfil necesario para vender	SI	SI	SI
Perfil necesario para comprar	SI	NO	SI
Categorización de publicaciones	SI	SI	NO
Chat público para cada publicación	SI	NO	SI
Datos de contacto públicos al momento de publicar	NO	SI	NO
Sistema de reserva de productos mientras se llega a acuerdo entre las partes	NO	NO	NO
Sistema de medición de interés de usuarios con la publicación(me gusta)	NO	NO	SI
Interacción entre usuarios			
Seguimiento a otros usuarios	NO	NO	NO
Chat privado entre usuarios	NO	NO	SI
Calificaciones entre usuarios	SI	NO	NO
Interacción con interfaz			
Página de inicio personalizada, de acuerdo a productos comprados y usuarios seguidos	NO	NO	PARCIALMENTE
Recomendación de publicaciones, de acuerdo a actividad previa del usuario	NO	NO	SI
Página de perfil que muestre actividad del usuario	NO	NO	SI

Fuente: Elaboración propia.

1.3. PROBLEMAS DETECTADOS

a) Mercadolibre.cl

Su sistema de comunicación entre los usuarios es deficiente. Los usuarios no pueden interactuar entre ellos dentro del sitio web de forma sencilla debido a:

- El único medio de comunicación con el que cuentan los usuarios es una caja de comentarios pública adjunta a cada publicación. No existe forma de interactuar personalmente con el usuario dueño de la publicación.

- Para poder realizar un contacto directo entre las partes, se debe realizar por otros medios (correo electrónico, teléfono, etc.), los cuales, solo se vuelven visibles en el momento en el que el usuario ya está seguro de realizar la compra.

  Esta característica de Mercadolibre.cl es un problema, producto de que cuando se realiza una compra, la mayor cantidad de dudas por parte del comprador se producen antes de estar seguros de adquirir el producto o servicio. Es por esto, que la fluidez y sencillez de la comunicación entre las partes debe iniciarse desde el momento en que el posible comprador muestra interés sobre dicho producto o servicio.

- Otra característica de este sitio web es la falta de interacción entre los usuarios más allá del momento de compra/venta, los usuarios no pueden manifestarse dentro de este espacio. Los usuarios, solo pueden manifestar su opinión a través de la calificación del usuario vendedor del producto o servicio, pero no pueden manifestarse en cuanto a sus intereses sobre algún producto/servicio, sobre otros usuarios, sobre preferencias, etc. Esta característica crea que el sitio web sea poco dinámico para los usuarios y, por lo tanto, resulta poco atractiva la oferta de permanecer en el sitio por más tiempo que el estrictamente necesario.

b) Yapo.cl

En la búsqueda de agilizar la compra/venta entre los usuarios, sacrifica la privacidad del usuario vendedor. Algunos datos de los usuarios (teléfono y ciudad), que realizan publicaciones, son expuestos públicamente, y no sólo para otros usuarios de Yapo.cl, sino que también, para cualquier persona que ingrese al sitio sin previo registro, lo que provoca que el vendedor no tenga ninguna referencia de quién es la persona con la que realiza la transacción, por lo que se presta para que personas intenten realizar algún tipo de engaño sin riesgos de recibir algún tipo de sanción por parte del sitio web.

Otra característica es la de falta de interacción entre los usuarios, la cual comparte con Mercadolibre.cl, siendo en el caso de Yapo.cl más evidente, debido a que ni siquiera existe un sistema de calificaciones para los usuarios vendedores.

c) Facebook.com

Al no ser un sitio web dedicado a la interacción de personas con el interés de comprar y vender productos, sus falencias son básicas. Entre ellas se pueden encontrar que no hay categoría de productos para su publicación, por lo que la búsqueda específica del producto es bastante difícil, por ende, si en alguno momento se visualiza un producto de interés, quizás con el tiempo no sea posible volver a encontrarlo. Además, no maneja ningún tipo de sistema de reputación para sus usuarios, por lo que los usuarios no se pueden generar una idea de quienes participan en la transacción.

A modo de resumen los problemas detectados de forma general son**:**

1. No existe un sitio web que otorgue confianza y agilidad, a la vez, durante el proceso de transacción.
2. No se fomenta el desarrollo de una comunidad orientada a la compra y venta.
3. Poca o nula personalización de su perfil mostrando sus productos.
4. Nulo sistema de seguimiento entre los compradores y vendedores.
5. Los sitios especializados en servir como nexo entre personas que realizan compres/ventas, no fomentan la comunicación y la interacción entre sus usuarios, lo cual impide el desarrollo de una comunidad.

1.4. DESCRIPCIÓN DEL SISTEMA PROPUESTO.

1.4.1. Objetivo General.

El objetivo general del presente trabajo es: “Diseñar e implementar una red social orientada a la venta/compra o intercambio a través de la publicación de productos y servicios en Chile”, esto con el fin de incrementar la oferta y demanda de diversos productos, crear una comunidad a partir de la compra y venta de productos y sobre todo impulsar en Chile el comercio virtual.

1.4.2. Objetivos Específicos.

Para dar cumplimiento al Objetivo General, se proponen los siguientes objetivos específicos:

1. Generar un medio de comunicación y contacto entre un comprador y un vendedor.
2. Definir el diseño del sitio web con sus respectivas páginas.
3. Incentivar el intercambio de opiniones a través de Internet.
4. Optimizar la compra y venta virtual
5. Disminuir la tasa de estafas en transacciones virtuales.

1.4.3. Beneficios del sistema.

El sistema propuesto trae consigo un beneficio a las personas que utilizan internet para realizar ventas y compras de productos y servicios. Primero, se potencia la creación de una comunidad, fomentando la confianza entre los usuarios, lo cual generara un ambiente propicio para realizar compras y ventas. Y segundo, se potenciará la comunicación entre los usuarios a través de la implementación de chat privados y públicos, y de la posibilidad de calificar a los otros usuarios y sus productos/servicios vendidos.

En definitiva, este sistema beneficiará a las personas a través de la entrega de un espacio especialmente diseñado para dar fluidez en la comunicación y confianza entre los usuarios que deseen comprar y vender productos por internet.

Lo descrito anteriormente, queda simplificado en los siguientes puntos:

1. Disponer un medio efectivo de vinculación entre comprador y vendedor: Esto se logra a través de los chats incorporados en el sitio junto con la posibilidad de contacto externo previo a la compra.

2. Fomentar el intercambio de opiniones: A través de los box de cada publicación, o en la calificación de los usuarios.

3. Mejorar la eficiencia en el proceso de venta y búsqueda a partir de la categorización de productos y mantención de la vigencia de publicaciones.

4. Propiciar un ambiente confiable. Esto se logra producto de: Sistema de reputación de usuarios, sistema de comunicación entre usuarios, sistema de seguidores de usuarios.

5. Disponer de publicaciones clasificadas.

6. El usuario podrá evaluar las Publicaciones.

1.4.4. Descripción General de la Solución Propuesta.

A partir de todo lo descrito anteriormente se hace necesario el desarrollo de un sistema que cumpla con las expectativas señaladas en sus objetivos y beneficios y que reúna las virtudes de cada plataforma señalada, principalmente que genere la confianza que otorga MercadoLibre con su sistema de reputación, sea fácil y expedito de utilizar como lo es Yapo.cl, y generar una comunidad dedicada a compartir un mismo interés y fomentando el diálogo, intercambio de opiniones/ideas tanto de forma pública (perfil) como privada (chat) lo cual permite Facebook.

El sistema a desarrollar va dirigido para 4 tipos de usuarios:

1. **Usuario comprador**: Aquella persona que desea encontrar productos/servicios de su interés, a través de Internet de forma rápida y sencilla, con el fin de crear una instancia para realizar la compra.

2. **Usuario vendedor emprendedor**: Aquella persona que utiliza Internet para ofrecer sus productos/servicios de forma periódica, con el fin de generar una rentabilidad con ello.

3. **Usuario vendedor de ocasión**: Aquella persona que no se dedica a vender productos por Internet, pero ocasionalmente desea vender alguno.

4. **Usuario social**: Aquella persona que no desea comprar ni vender, pero desea ser parte de la comunidad.

La propuesta a grandes rasgos funcionará de la siguiente forma:

1. Las personas que deseen hacer uso del sistema deberán registrase, en primera instancia. En el registro se deberán ingresar los datos personales de los usuarios, tales como nombre, ciudad y datos de contacto.

2. Luego de registrarse, los usuarios podrán realizar publicaciones de productos/servicios. Los datos de contactos serán de carácter público y podrán ser contactados abiertamente por los otros usuarios, ya sea, por medios del sistema (chat privado entre usuarios o chat publico adjunto a cada publicación) o a través de medios de contactos personales (email, teléfono, etc.).

3. Los usuarios realizarán la compra/venta por medios externos al sistema, según ellos estimen conveniente. El sitio web solo sirve como medio para vincular a las personas interesadas.

4. Los usuarios que hayan llevado a una transacción deberán calificar a su contraparte, con el fin de llevar un registro del actuar de los usuarios a la hora de realizar una compra/venta.

5. Luego de confirmar la transacción se procede a cerrar la publicación, para luego, pasar a ser parte de las publicaciones concretadas.

En cuanto a la interacción entre los usuarios, se podrán realizar las siguientes acciones:

1. Los usuarios tendrán a su disposición un chat privado, el que podrán utilizar como medio de comunicación para agilizar el acuerdo de una transacción o simplemente para relacionarse con otro usuario.

2. Las publicaciones tendrán incorporado un chat público, en la que todos los usuarios interesados podrán participar y resolver sus dudas en cuanto al producto/servicio publicado.

3. Los usuarios podrán calificar después de realizar una transacción a su contraparte, con el fin de comunicar a la comunidad la calidad de los usuarios a la hora de realizar una compra/venta.

4. Los usuarios podrán realizar seguimiento de la actividad de los demás usuarios, recibiendo recomendaciones de las publicaciones de estos.

En cuanto a la interacción de los usuarios con el sistema, se tendrán las siguientes funcionalidades:

1. Los usuarios dispondrán de una página de perfil personalizada, la cual será pública para el resto de los usuarios. En ella se llevará el registro de la actividad de los usuarios (compras y ventas realizadas), publicaciones activas del usuario, calificación y seguidores.

2. Los usuarios dispondrán de una página de inicio personalizada. En ella se mostrarán las últimas publicaciones realizadas por los usuarios seguidos o las últimas publicaciones en base a las compras realizadas por el usuario.

3. Los usuarios dispondrán de categorías y subcategorías las que podrá podrá elegir, esto con el fin de desplegar publicaciones relacionadas con sus preferencias.

1.4.5. Estructura funcional del sistema.

a) Administración de cuentas de usuario:

1. **Registro cuenta de usuario**: Cuando un usuario nuevo quiere hacer uso de la aplicación, debe pasar por el proceso de registro, en el cual deberá ingresar sus datos personales. El sistema validara que el usuario no este registrado anteriormente, para luego generar su perfil.

2. **Suspender cuenta de usuario**: Las personas tendrán la capacidad de suspender sus cuentas por un tiempo indefinido, cuando decidan dejar la comunidad.

3. **Modificación de cuentas de usuario**: Los usuarios podrán realizar modificación a sus cuentas de usuario, con el fin de cambiar datos de contacto, ubicación, etc.

b) Administración de sesión de usuario:

1. **Inicio de sesión**: Se pedirá un usuario y contraseña, luego se validará que el usuario exista para dirigirlo a su perfil, en donde se mostraran sus publicaciones.

2. **Cierre de sesión**: El usuario finaliza su sesión.

c) Administración de publicaciones:

1. **Crear publicaciones**: Cuando un usuario desea vender un artículo, debe llenar un formulario breve en donde se especificarán categoría, stock, descripción, precio y fotografías.

2. **Eliminar publicaciones**: Hay distintas formas para que la publicación sea eliminada:

    a. El usuario podrá eliminar una publicación realizada por el mismo cuando lo estime conveniente.

    b. La publicación será eliminada si la venta se concreta y el stock es igual a 0.

    c. Cada 60 días se notificará al usuario la posibilidad de republicar su producto o será eliminada automáticamente.

3. **Reponer publicación**: Esta opción estará disponible luego de 60 días desde que la publicación fue realizada. Esta opción permitirá tomar los datos de la publicación original, modificar su fecha y publicar nuevamente.

4. **Modificar publicaciones**: El usuario podrá modificar los datos de la publicación cuando lo estime conveniente.

d) Administración de contacto entre usuario:

1. ***Chat privado***: Los usuarios podrán contactarse personalmente con otros usuarios, a través de un chat interno entre los usuarios.

2. ***Chat público en Publicaciones***: Los usuarios podrán interactuar a través de un medio de comunicación en cada una de las publicaciones, con el fin de resolver dudas propias de cada una de las publicaciones.

e) Administración de reservas de productos/servicios:

1. **Crear reserva de producto/servicio**: El usuario que quiera reservar un producto/servicio para efectuar la compra deberá pulsar el botón correspondiente. Al instante se notificará al usuario dueño de la publicación que posee una reserva, junto con los datos del usuario que realizo la reserva.

2. **Cancelar reserva de producto/servicio:** El usuario que realizó una publicación y el que realizó una reserva podrá cancelarla en cualquier momento. Al instante se enviará una notificación a la contraparte indicando que la reserva ha sido cancelada.

f) Búsquedas en sistema.

1. **Realizar búsqueda de productos/servicios**: El usuario podrá buscar productos de su interés en la aplicación.

2. **Realizar búsqueda de usuarios:** El usuario podrá buscar a otros usuarios dentro de la aplicación.

g) Páginas personalizadas dinámicas:

1. **Página de inicio personalizada**: Se generará una página de inicio para cada usuario que responda a la actividad de los usuarios y a las preferencias que manifiesten dentro del sitio web.

2. **Página de perfil personalizada**: Se generará una página de perfil personalizada para cada usuario, la cual, muestre la actividad de dicho usuario y sus publicaciones activas.

h) Administración de reputación:

1. **Calificar comprador**: El usuario dueño de la publicación podrá calificar a su contraparte al momento de lograr una transacción o al momento de ser cancelada.

2. **Calificar Vendedor**: El usuario comprador podrá calificar a su contraparte al momento de lograr una transacción o al momento de ser cancelada.

i) Administrador de categorías:

1. **Agregar categoría**: Agregar una categoría nueva al sistema.

2. **Modificar categoría**: Modificar una categoría existente en el sistema.

3. **Eliminar categoría**: Eliminar una categoría existente en el sistema.

j) Administrador de sub categorías:

1. **Agregar sub categoría**: Agregar una sub categoría nueva al sistema.

2. **Modificar sub categoría**: Modificar una sub categoría existente en el sistema.

3. **Eliminar sub categoría**: Eliminar una sub categoría existente en el sistema.

k) Administrador de regiones:

1. **Agregar región**: Agregar una región nueva al sistema.
2. **Modificar región**: Modificar una región existente en el sistema.
3. **Eliminar región**: Eliminar una región existente en el sistema.

l) Administrador de ciudades:

1. **Agregar ciudad**: Agregar una ciudad nueva al sistema.
2. **Modificar ciudad**: Modificar una ciudad existente en el sistema.
3. **Eliminar ciudad**: Eliminar una ciudad existente en el sistema.

1.4.6. Descripción de fórmulas y cálculos empleados.

El sistema no emplea cálculos ni fórmulas matemáticas.

1.4.7. Información Manejada por el Sistema.

1.4.7.1. Entradas

1. **Formulario de registro de usuario**: Corresponde a los datos personales y de contacto de cada uno de los usuarios del sistema. Estos datos constituyen la base de necesaria para poder realizar publicaciones. Los datos que se ingresarán son los siguientes:

   Nombre, Teléfono, Mail, Nombre de usuario, Contraseña, Ubicación, Imagen de perfil.

2. **Formulario de registro de publicación**: Corresponde a los datos que describen el producto/servicio en publicación. La información servirá para poder generar las vistas de publicaciones y para poder categorizar la publicación dentro de nuestro sitio web.

   Título, Descripción, Categoría, Stock, Precio, Link de referencia,

3. **Formulario de registro de ciudades**: Corresponde a la creación de una nueva ciudad perteneciente a una región existente en el sistema. Esto solo puede ser manejado por un administrador.

   Nombre, Región, Código postal.

4. **Formulario de registro de región**: Corresponde a la creación de una región. Solo puede ser manejado por un administrador.

   Nombre. Id de la región.

5. **Formulario de registro de venta**: Corresponde al registro de una compra/venta, en donde se utilizarán datos almacenados en la base de datos, además de solicitar a los usuarios los siguientes datos.

   Calificación del vendedor, Calificación del comprador, Observación.

6. **Formulario de registro de comentario en la publicación**: Corresponde a la creación de un comentario en alguna publicación específica por algún interesado.

   Comentario.

7. **Formulario de mensaje entre usuarios**: Corresponde a la creación y envió de mensajes entre usuarios pertenecientes al sistema.

   Mensaje.

8. **Formulario de registro de subcategoría**: Corresponde a la creación de una nueva subcategoría perteneciente a una categoría existente en el sistema. Esto solo puede ser utilizado por un administrador.

   Nombre de la subcategoría, Descripción, Id de la categoría perteneciente.

9. **Formulario de registro de categoría**: Corresponde a la creación de una nueva categoría. Esto solo puede ser utilizado por un administrador.

   Nombre de la categoría, descripción.

10. **Formulario de registro de administrador**: Corresponde a la creación de un nuevo administrador en el sistema, esto solo puede ser utilizado por otro administrador.

    Nombre, Correo.

1.4.7.2. Salidas

Las salidas que generará el sistema serán un conjunto de interfaces que permitirán la interacción entre el sitio web y los usuarios.

- **Vista login**: Corresponde a vista que permitirá la autentificación de los usuarios para ingresar al sistema.

- **Vista de perfiles**: Corresponden a las vistas que mostrarán la información de cada uno de los usuarios.

- **Vistas de publicaciones**: Corresponden a las vistas de las publicaciones realizadas en el sitio web. Mostrarán los datos de una publicación en específico.

- **Vistas de formularios de usuarios y publicaciones**: Corresponde a la vista que permitirá que los usuarios puedan ya sea registrarse como usuario o realizar una publicación.

- **Vista de inicio**: Vista principal del usuario, en donde se mostrarán publicaciones y usuarios según las preferencias manifestadas.

- **Vista de modificación de usuario**: Vista la cual despliega la información del usuario con la posibilidad de poder modificar algunos datos o suspender la cuenta.

- **Vista de modificación de publicación**: Corresponde a la vista que permite modificar información o datos de la publicación además de permitir la eliminación de la publicación si el usuario lo desea.

- **Vista de registro de venta**: Corresponde a la vista en la cual ambos usuarios (comprador y vendedor), pondrán calificar a la contraparte además de ingresar una observación si lo desea.

- **Vista de mensajes**: Corresponde a la vista en la cual dos usuarios pueden comunicarse a través de mensajes enviados por el sistema.

1.4.8. Entidades de Información.

A continuación, se presenta una lista de las entidades utilizadas, con un breve resumen de su contenido, las cuales se utilizarán para la creación de la aplicación.

1. **Login**: Almacena la información necesaria para autentificar si la sesión es correcta y si es un usuario o un administrador.

2. **Administrador**: Almacena los datos relacionados de cada administrador que pertenezca al sistema, como el nombre y el mail.

3. **Notificación**: Almacena los avisos para cada usuario, como una venta realizada o un nuevo amigo.

4. **Amigo**: Almacena los contactos de cada usuario.

5. **Interés**: Almacena los usuarios interesados para cada publicación.

6. **Usuario**: Almacena los datos relacionados con cada usuario que podrá acceder al sistema, como su nombre, teléfono, ciudad etc.

7. **Ciudad**: Corresponde a la información de la ciudad a la cual pertenece un usuario.

8. **Región**: Corresponde a la información de la región a la cual pertenece una ciudad.

9. **Publicación**: Almacena la información de la publicación en donde se expondrá el articulo a vender.

10. **Registro_transaccion**: Contiene la información de una transacción realizada.

11. **Categoría**: Corresponde a la categoría a la cual corresponde la sub categoría.

12. **Subcategoría**: Corresponde a la subcategoría a cuál pertenece la publicación realizada.

13. **Usuario_seguidor**: Corresponde a la relación entre un usuario y su seguidor.

14. **Preferencia**: Corresponde a la relación entre una categoría y un usuario que la selecciona.

15. **Imagen**: Almacena los datos de las imágenes asociadas a las publicaciones.

16. **Comentario**: Representa los comentarios de los usuarios dentro de una publicación.

17. **Mensaje**: Almacena los mensajes enviados entre dos usuarios distintos.

1.4.9. Estructuras de códigos.

1. **Entidad:** Usuario
   Clave principal: id_usuario, corresponde a un número correlativo.
2. **Entidad:** Ciudad
   Clave principal: id_ciudad, código identificador de ciudad, de acuerdo al código postal.
3. **Entidad:** Región
   Clave principal: id_region, código identificador de región. Corresponde a un número desde el 1 al 15.
4. **Entidad:** Publicación
   Clave principal: Clave compuesta por id_usuario (número correlativo) + id_publicacion (número correlativo).
5. **Entidad:** Registro transacción
   Clave principal: Clave compuesta por Id_usuario (número correlativo) + id_publicacion (número correlativo) + id_transaccion (número correlativo)

6. **Entidad:** Categoría

   Clave principal: Id_categoria, corresponde a un número correlativo.

7. **Entidad:** Sub Categoría

   Clave principal: id_subcategoria, corresponde a un número correlativo.

8. **Entidad:** Usuario_seguidor

   Clave principal: Clave compuesta por id_usuario (número correlativo) + id_usuario_seguidor (número correlativo).

9. **Entidad:** Preferencia

   Clave principal: Clave compuesta por id_usuario (número correlativo) + id_subcategoria (número correlativo).

10. **Entidad:** Imagen

    Clave principal: Clave compuesta por id_usuario (número correlativo) + id_publicacion (número correlativo) + id_imagen (número correlativo).

11. **Entidad:** Comentario

    Clave principal: Clave compuesta por id_usuario (número correlativo) + id_publicacion (número correlativo) + id_usuario_comentario (número correlativo) + fecha_hora (Fecha y hora en que se genera el comentario).

12. **Entidad:** Mensaje

    Clave principal: Clave Compuesta por id_usuario_emisor (número correlativo) + id_usuario_receptor (número correlativo) + fecha_hora_chat (Fecha y hora en que se genera el mensaje).

13. **Entidad:** Administrador

    Clave principal: id_administrador corresponde a número correlativo.

14. **Entidad:** Login

    Clave principal: Correo corresponde a formato de correo electrónico.

15. **Entidad:** Amigo

    Clave principal: Clave Compuesta por id_usuario (número correlativo) + id_usuario_amigo (número correlativo).

16. **Entidad:** Notificación

    Clave principal: Clave Compuesta por id_usuario (número correlativo) + id_notificación (número correlativo).

17. **Entidad:** Interés

    Clave principal: Clave Compuesta por id_usuario (número correlativo) + id_publicacion (número correlativo) + id_usuario_interesado (número correlativo).

1.4.10. Condicionantes de diseño (tiempo de respuesta, periodicidad, otros).

A continuación, se presentarán las condicionantes de diseño para el sistema propuesto:

- El sistema contará con un usuario administrador el cual tendrá acceso a todos los mantenedores, además este administrador permitirá eliminar o crear nuevos administradores del sistema.
- Se utilizará software libre y gratuito para el desarrollo del proyecto, para la Base de Datos y para la página web.
- Se pretende realizar un respaldo diario programado de la Base de Datos, esto con el fin de darle sustentabilidad al sistema en los inicios, ya que se espera recibir una cantidad moderadamente baja de usuario, este respaldo puede ser modificado con el tiempo.
- El servidor será hosteado en un servidor casero, lo cual con el tiempo se piensa migrar a uno dedicado.

1.4.11. Modelo de datos.

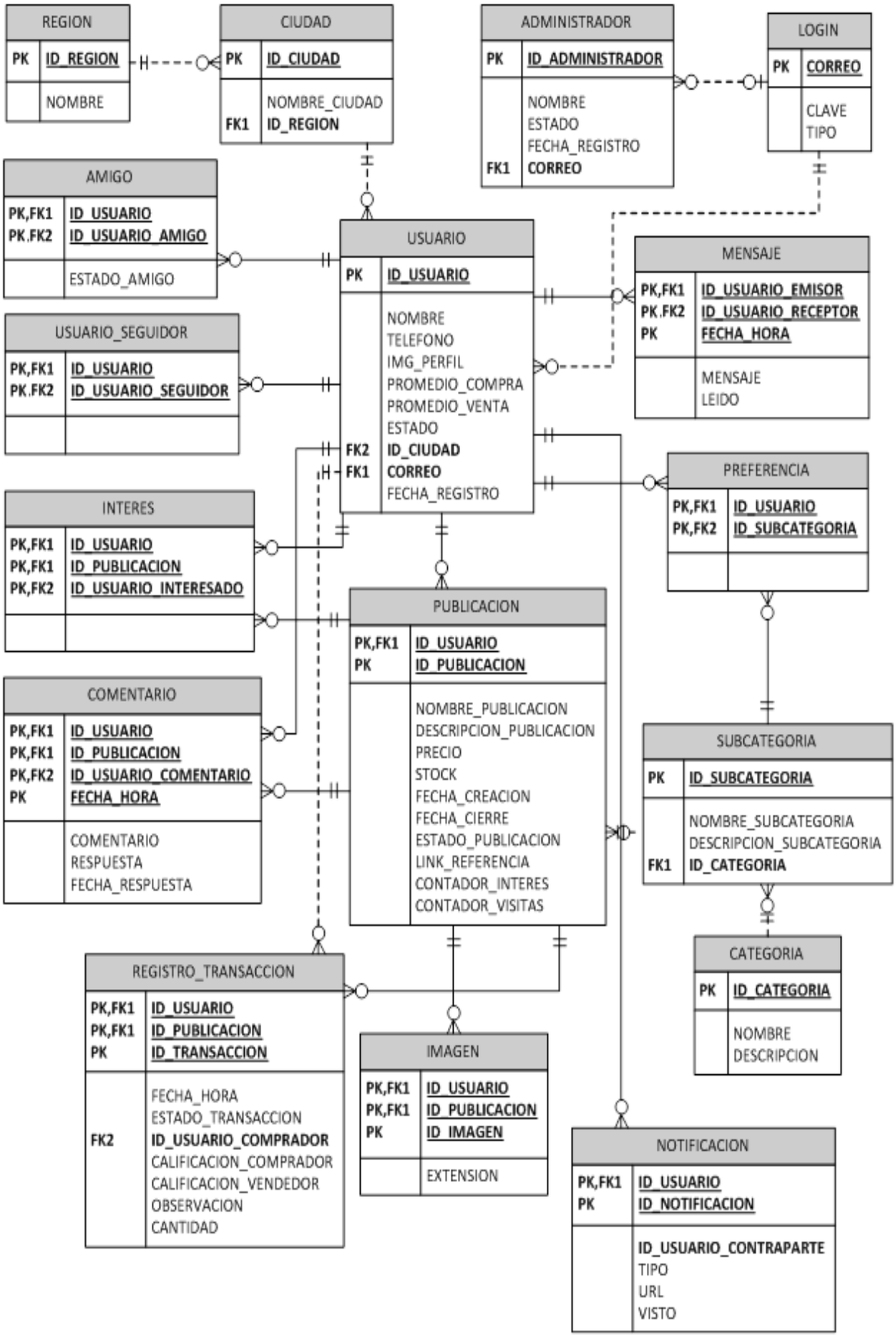

Figura 1-1. Modelo de Datos.

CAPÍTULO II. MEDIO AMBIENTE COMPUTACIONAL Y DISEÑO FÍSICO

2. MEDIO AMBIENTE COMPUTACIONAL Y DISEÑO FÍSICO.

2.1. AMBIENTE COMPUTACIONAL

2.1.1. Configuración del sistema.

A continuación, se detallan las características de los equipos que se emplearán en el desarrollo del sistema, como las posibles características del servidor que se deberían considerar como aceptables. Además se detallan los requerimientos mínimos necesarios para desplegar el sistema en cualquier plataforma (ordenador).

a) **Equipos de Desarrollo**: Para el desarrollo se emplearon 2 equipos de trabajo, cada uno de ellos corresponden al equipamiento personal de los autores del presente trabajo.

CPU:	QuadCore Intel Core i5-7600k, 4200 MHz
Memoria:	Corsair Vengeance LPX 8144 MB DDR4-2300
Almacenamiento:	SSD Crucial 240BX (240GB, SATA-III)
Pantalla:	ViewSonic VX2276 SERIES 22”
Conectividad:	Realtek PCle GBE Family Controller

CPU:	QuadCore Intel Core I5-6200U, 2300 MHz
Memoria:	Samsung 8192 MB DDR3-1333
Almacenamiento:	Hitachi 1 TB Sata
Pantalla:	Generic PnP Monitor 15”
Conectividad:	Realtek Tarjeta integrada

b) **Equipo servidor**: La presente descripción corresponde a las características mínimas de un equipo servidor. Estos podrían variar dependiendo de las siguientes situaciones:

- Cantidad de usuarios
- Otras aplicaciones que corran en el servidor
- Tamaño y cantidad de documentos
- Rol del usuario que utilice la plataforma de trabajo

CPU:	Dual Core 3.2 GHz
Memoria:	2 GB
Almacenamiento:	Capacidad: 80 GB
Conectividad:	Descripción: tarjeta integrada, Banda ancha de 100 Mbps.

c) **Equipo Usuario**: El equipo usuario debería tener las siguientes características de memoria y conectividad.

Memoria:	1 GB
Conectividad:	Tarjeta red integrada o wireless, que cuente con una conexión a Internet.

2.1.2. Software utilizado.

a) **Sistema operativo**.

El sistema operativo empleado para el desarrollo del sistema será Windows 10; la versión específica usada para desarrollar será su versión profesional. Las características principales que entrega este sistema operativo son:

— Una interfaz gráfica amigable y de fácil uso para el usuario.

— Permite una fácil instalación, soporte y una amplia compatibilidad para la mayoría de programas y dispositivos.

— Su diseño permite que varias aplicaciones se ejecuten simultáneamente, además de garantizar una rápida respuesta y una gran estabilidad del sistema.

— Gran aceptación del SO por parte de los usuarios a nivel mundial, los cual permite abarcar a la gran mayoría de éstos, eliminando así posibles problemas de compatibilidad por SO.

b) **Herramientas de desarrollo de software**.

— CodeIgniter: Framework para PHP. Permite a los desarrolladores realizar proyectos de forma rápida otorgando un conjunto de bibliotecas para tareas comunes, además de una interfaz simple.

— Bootstrap 4.0: Es un Framework con orientación mobile first y desarrollo responsivo; permite fácil generación de HTML, CSS y JS.

— XAMPP: Es un pack de programas que reúne las siguientes aplicaciones, Apache, OpenSSL, MySQL, PHP, phpMyAdmin, Perl, FileZilla FTP Server y Mercury Mail.

— Brackets 1.9: Es un proyecto de código abierto integrado libre, hecho principalmente por desarrolladores web, lo cual, otorga una gran facilidad para el desarrollo en ambiente web.

— Atom 1.23.1: Editor de código abierto multiplataforma desarrollado en Electron, que permite escribir código para distintos lenguajes de programación.

c) **Lenguajes de programación**.

— **PHP**: Lenguaje de programación, diseñado originalmente para desarrollo web de contenido dinámico. Compatible con la mayoría de sistemas operativos y servidores web.

— **HTML**: HyperText Markup Language (lenguaje de marcas de hipertexto), es un lenguaje estándar que sirve de referencia para la elaboración de páginas web en sus diferentes versiones. Define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, etc.

— **Java Script**: es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipo e imperativo. Se utiliza principalmente en su forma del client-side (lado del cliente), implementado como parte de un desarrollo web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas. Su sintaxis es similar a C, aunque adopta nombres y convenciones de JAVA.

d) **Servidor web**.

Apache: Un servidor web HTTP de código abierto, multiplataforma, modular y extensible.

e) **Servidor de base de datos**:

MySQL: Es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones.

f) **GUI**:

phpMyAdmin: Es una herramienta para manejar la administración de MySQL a través de páginas web, utilizando Internet. Actualmente puede crear y eliminar Bases de Datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL, administrar claves en campos, administrar privilegios y exportar datos en varios formatos.

2.2. LISTADO DE TABLAS

1. **Login**: Almacena la información necesaria para autentificar si la sesión es correcta y si es un usuario o un administrador.
2. **Administrador**: Almacena los datos relacionados de cada administrador que pertenezca al sistema, como el nombre y el mail.
3. **Notificación**: Almacena los avisos para cada usuario, como una venta realizada o un nuevo amigo.
4. **Amigo**: Almacena los contactos de cada usuario, y si están en línea.
5. **Interés**: Almacena los usuarios interesados para cada publicación.
6. **Usuario**: Almacena los datos relacionados con cada usuario que podrá acceder al sistema, como su nombre, teléfono, ciudad etc.
7. **Ciudad**: Corresponde a la información de la ciudad a la cual pertenece un usuario.
8. **Región**: Corresponde a la información de la región a la cual pertenece una ciudad.
9. **Publicación**: Almacena la información de la publicación en donde se expondrá el articulo a vender.
10. **Registro_transaccion**: Contiene la información de una transacción realizada.
11. **Categoría**: Corresponde a la categoría a la cual corresponde la sub categoría.
12. **Subcategoría**: Corresponde a la subcategoría a cuál pertenece la publicación realizada.
13. **Usuario_seguidor**: Corresponde a la relación entre un usuario y su seguidor.
14. **Preferencia**: Corresponde a la relación entre una categoría y un usuario que la selecciona.
15. **Imagen**: Almacena los datos de las imágenes asociadas a las publicaciones.
16. **Comentario**: Representa los comentarios de los usuarios dentro de una publicación.
17. **Mensaje**: Almacena los mensajes enviados entre dos usuarios distintos.

2.3. DESCRIPCIÓN DE TABLAS

Para la descripción de cada tabla, se empleará un diseño como el siguiente.

NOMBRE:		
DESCRIPCIÓN:		
CLAVE PRINCIPAL:		
CLAVES FORÁNEAS:		
DESCRIPCIÓN DE REGISTRO		
CAMPO	**TIPO**	**DESCRIPCIÓN**

1. **NOMBRE**: Nombre identificatorio que tendrá la tabla.

2. **DECRIPCIÓN**: Breve descripción de los datos que almacenará la tabla.

3. **CLAVE PRINCIPAL**: Nombre de uno o más campos que permiten individualizar a cada registro de la tabla. En el caso que se trate de una clave compuesta, se empleará el signo + para unir un campo con otro.

4. **CLAVES FORÁNEAS**: Nombre de uno o más campos que permiten referenciar a cada registro de la tabla con otra tabla. En el caso que se trate de una clave compuesta, se empleará el signo + para unir un campo con otro. Seguido se anotará una flecha y al lado derecho de ésta el nombre de la tabla que referenciará, por ejemplo:

   id_ciudad → **CIUDAD**

5. **CAMPO**: Nombre identificatorio del campo.

6. **TIPO**: Tipo de dato asignado al campo. Este puede ser uno de los que se muestran en la siguiente tabla y que maneja MariaDB:

Int:	Desde -2147483648 a 2147483647. Sin signo el rango va desde 0 a 4294967295.
TinyInt:	Desde -128 a 127. Sin signo el rango de valores es de 0 a 255.
Datetime:	Combinación de fecha y hora. El formato de almacenamiento es de YY-MM-DD HH:MM:SS.
Varchar:	Almacena una cadena de longitud variable. La cadena podrá contener desde 0 a 255 caracteres.
Char:	Almacena una cadena de longitud física. La cadena podrá contener desde 0 a 255 caracteres.

7. **DESCRIPCIÓN**: Breve descripción que detalla aspectos relevantes del campo descrito.

2.3.1. Tabla Usuario.

Tabla 2-1. Descripción de Tabla: USUARIO

DESCRIPCIÓN DE TABLA	
NOMBRE:	**USUARIO**
DESCRIPCIÓN:	Representa los datos de los usuarios que están registrados en el sistema.
TIPO DE TABLA:	Maestra.
CLAVE PRIMARIA:	id_usuario.
CLAVES FORÁNEAS:	id_ciudad → **CIUDAD** correo → **LOGIN**

DESCRIPCIÓN DE REGISTRO		
NOMBRE	**TIPO**	**DESCRIPCIÓN**
id_usuario	Int (7)	Código identificador de cada usuario. Corresponde a un número correlativo desde el 0 hasta 9.999.999.
nombre	Varchar(30)	Nombre del usuario en el sistema. Puede estar compuesto por letras y números, sin caracteres especiales.
telefono	Int(8)	Número telefónico del usuario. Compuesto de 8 dígitos.
img_perfil	Varchar(4)	Nombre de extensión de imagen de usuario, ejemplo ".png".
promedio_compra	TinyInt(2)	Promedio de calificaciones como comprador de un usuario. Corresponde a un número del 1 al 10.
promedio_venta	TinyInt(2)	Promedio de calificaciones como vendedor de un usuario. Corresponde a un número del 1 al 10.
correo	Varchar(50)	Dirección de correo electrónico del usuario.
id_ciudad	Int(7)	Código identificador de ciudad a que pertenece el usuario, de acuerdo al código postal.
estado	Char(1)	Carácter que representa el estado del usuario en el sistema "v" para Vigente, "e" para expulsado y "c" para cancelado.
Fecha_registro	DateTime	Fecha de registro de usuario. Con el formato YYYY-MM-DD HH:MM:SS.

2.3.2. Tabla Publicacion.

Tabla 2-2. Descripción de Tabla: PUBLICACIÓN

DESCRIPCIÓN DE TABLA		
NOMBRE:	**PUBLICACION**	
DESCRIPCIÓN:	Representa los datos de las publicaciones que se realizarán en el sistema.	
TIPO DE TABLA:	Maestra.	
CLAVE PRIMARIA:	id_usuario + id_publicacion.	
CLAVES FORÁNEAS:	id_usuario → **USUARIO** id_subcategoria → **SUBCATEGORIA**	
DESCRIPCIÓN DE REGISTRO		
NOMBRE	**TIPO**	**DESCRIPCIÓN**
id_usuario	Int (7)	Código identificador de usuario dueño de la publicación. Corresponde a un número correlativo desde el 0 hasta 9.999.999.
id_publicacion	Int (4)	Código identificador de publicación, representada por un número correlativo para cada publicación desde el 0 al 9999 por usuario.
Nombre_publicacion	Varchar(50)	Nombre del título de la publicación. Puede estar compuesta por cualquier tipo de carácter.
Descripción_publicacion	Varchar(1000)	Representa la descripción de cada publicación.
precio	Int(10)	Representa el valor monetario que tiene el producto o servicio publicado. Valor numérico desde el 1 hasta el 99.999.999.
stock	Int(3)	Cantidad de productos de la publicación.
fecha_creación	Datetime	Fecha en que la publicación fue creada. Con el formato YYYY-MM-DD HH:MM:SS.
fecha_cierre	Datetime	Fecha en que la publicación expirara. Con el formato YYYY-MM-DD HH:MM:SS.
estado_publicacion	Char(1)	Estado en que se encuentra la publicación, V para vigente y C para cerrada.
link_referencia	Varchar(30)	Link externo opcional que tiene alguna relación con el producto o servicio publicado.
contador_interes	Int(5)	Contador del interés de los usuarios sobre la publicación.
contador_visitas	Int(9)	Contador de visitas de la publicación.
id_subcategoria	Int(4)	Código identificador de la sub categoría de la publicación. Corresponde a un número correlativo desde 0 hasta 9.999.

2.3.3. Tabla Ciudad.

Tabla 2-3. Descripción de Tabla: CIUDAD

DESCRIPCIÓN DE TABLA		
NOMBRE:	**CIUDAD**.	
DESCRIPCIÓN:	Representa la ciudad a la que pertenecen los usuarios.	
TIPO DE TABLA:	Maestra.	
CLAVE PRIMARIA:	id_ciudad.	
CLAVES FORÁNEAS:	id_region → **REGION**	
DESCRIPCIÓN DE REGISTRO		
NOMBRE	**TIPO**	**DESCRIPCIÓN**
id_ciudad	Int(7)	Código identificador de ciudad, de acuerdo al código postal.
Nombre_ciudad	Varchar(30)	Nombre sin abreviaturas de la ciudad, considerando espacios en blanco.
id_region	TinyInt(2)	Código identificador de región a la cual pertenece la ciudad. Corresponde a un número desde el 1 al 15. 13 para Región Metropolitana.

2.3.4. Tabla Región.

Tabla 2-4. Descripción de Tabla: REGION

DESCRIPCIÓN DE TABLA		
NOMBRE:	**REGION**	
DESCRIPCIÓN:	Representa a la región del país a la cual pertenecen los usuarios.	
TIPO DE TABLA:	Maestra.	
CLAVE PRIMARIA:	id_region.	
CLAVES FORÁNEAS:		
DESCRIPCIÓN DE REGISTRO		
NOMBRE	**TIPO**	**DESCRIPCIÓN**
id_region	TinyInt(2)	Código identificador de región. Corresponde a un número desde el 1 al 15. 13 para Región Metropolitana.
Nombre_region	Varchar(30)	Nombre sin abreviatura de la región

2.3.5. Tabla Registro_transaccion

Tabla 2-5. Descripción de Tabla: REGISTRO_VENTA

DESCRIPCIÓN DE TABLA		
NOMBRE:	**REGISTRO_TRANSACCION**	
DESCRIPCIÓN:	Representa los datos que registran una transacción entre dos usuarios.	
TIPO DE TABLA:	Intersección.	
CLAVE PRIMARIA:	Id_usuario + id_publicacion + id_transaccion	
CLAVES FORÁNEAS:	Id_publicacion+Id_usuario → **PUBLICACION** id_usuario_comprador → **USUARIO (id_usuario)**	
DESCRIPCIÓN DE REGISTRO		
NOMBRE	**TIPO**	**DESCRIPCIÓN**
Id_usuario	Int(7)	Código identificador de usuario dueño de la publicación. Corresponde a un número correlativo desde el 0 hasta 9.999.999.
id_publicacion	Int(4)	Código identificador de la publicación a la cual pertenece el registro de venta, representada por un número correlativo para cada publicación desde el 0 al 999.
id_transaccion	Int(3)	Código identificador de cada venta realizada, en curso o cancelada en el sistema. Corresponde a un número correlativo desde el 1 hasta el 999
fecha_hora	Datetime	Fecha y hora en que se genera la reserva, la venta o la cancelación (actualizable). Con el formato YYYY-MM-DD HH:MM:SS.
Estado_transaccion	char(1)	Representa el estado actual del proceso de venta. R para realizada, C para cancelada y E para en curso.
id_usuario_comprador	Int(7)	Código identificador de usuario interesado en la publicación. Corresponde a un número correlativo desde el 0 hasta 9.999.999.
calificacion_vendedor	Tinyint	Calificaciones del usuario dueño de la publicación, recibida del comprador. Corresponde a un número del 1 al 10.
calificacion_comprador	Tinyint	Calificaciones del usuario comprador de la publicación, recibida del vendedor. Corresponde a un número del 1 al 10.
Observación	Varchar(255)	Observación que describe el estado de la transacción y el usuario que la alteró.
Cantidad	Int(3)	Cantidad de productos involucrados en la transacción.

2.3.6. Tabla Comentario.

Tabla 2-6. Descripción de Tabla: COMENTARIO

DESCRIPCIÓN DE TABLA		
NOMBRE:	**COMENTARIO**	
DESCRIPCIÓN:	Representa los comentarios de los usuarios dentro de una publicación y la respuesta del autor.	
TIPO DE TABLA:	Intersección	
CLAVE PRIMARIA:	id_usuario + id_publicacion + id_usuario_comentario + fecha_hora	
CLAVES FORÁNEAS:	id_publicacion + id_usuario → **PUBLICACION** id_usuario_comentario → **USUARIO (id_usuario)**	
DESCRIPCIÓN DE REGISTRO		
NOMBRE	**TIPO**	**DESCRIPCIÓN**
Id_usuario	Int(7)	Código identificador de usuario dueño de la publicación. Corresponde a un número correlativo desde el 0 hasta 9.999.999.
id_publicacion	Int(4)	Código identificador de la publicación a la cual pertenece el comentario, representada por un número correlativo para cada publicación desde el 0 al 999.
id_usuario_comentario	Int(7)	Código identificador de usuario que realiza el comentario en la publicación. Corresponde a un número correlativo desde el 0 hasta 9.999.999.
fecha_hora	Datetime	Fecha y hora en que se genera el comentario. Con el formato YYYY-MM-DD HH:MM:SS.
Comentario	Varchar(255)	Contenido del comentario realizado en una publicación. Permite cualquier carácter alfanumérico y especial.
Respuesta	Varchar(255)	Contenido de la respuesta del usuario vendedor. Permite cualquier carácter alfanumérico y especial.
Fecha_respuesta	Datetime	Fecha y hora en que se genera la respuesta del vendedor. Con el formato YYYY-MM-DD HH:MM:SS.

2.3.7. Tabla Usuario_Seguidor.

Tabla 2-7. Descripción de Tabla: USUARIO_SEGUIDOR

DESCRIPCIÓN DE TABLA		
NOMBRE:	**USUARIO_SEGUIDOR ***	
DESCRIPCIÓN:	Representa la relación entre un usuario y otro usuario el cual se convierte en su seguidor.	
TIPO DE TABLA:	Intersección	
CLAVE PRIMARIA:	id_usuario + id_usuario_seguidor.	
CLAVES FORÁNEAS:	id_usuario → **USUARIO** Id_usuario_seguidor → **USUARIO (id_usuario)**	
DESCRIPCIÓN DE REGISTRO		
NOMBRE	**TIPO**	**DESCRIPCIÓN**
id_usuario	Int(7)	Código identificador de usuario seguido. Corresponde a un número correlativo desde el 0 hasta 9.999.999.
id_usuario_seguidor	Int(7)	Código identificador de usuario seguidor. Corresponde a un número correlativo desde el 0 hasta 9.999.999.

2.3.8. Tabla Preferencia.

Tabla 2-8. Descripción de Tabla: PREFERENCIA

DESCRIPCIÓN DE TABLA		
NOMBRE:	**PREFERENCIA**	
DESCRIPCIÓN:	Representa la preferencia de un usuario y una subcategoría de publicaciones.	
TIPO DE TABLA:	Intersección	
CLAVE PRIMARIA:	id_usuario + id_subcategoria.	
CLAVES FORÁNEAS:	id_usuario → USUARIO id_subcategoria → SUBCATEGORIA	
DESCRIPCIÓN DE REGISTRO		
NOMBRE	**TIPO**	**DESCRIPCIÓN**
id_usuario	Int(7)	Código identificador de usuario que manifiesta su preferencia. Corresponde a un número correlativo desde el 0 hasta 9.999.999.
id_subcategoria	Int(3)	Código identificador de una sub categoría de publicaciones. Corresponde a un número correlativo desde 0 hasta 999.

2.3.9. Tabla Mensaje.

Tabla 2-9. Descripción de Tabla: Mensaje

DESCRIPCIÓN DE TABLA		
NOMBRE:	MENSAJE *	
DESCRIPCIÓN:	Representa los mensajes de los usuarios enviados entre ellos.	
TIPO DE TABLA:	Maestra	
CLAVE PRIMARIA:	id_usuario_emisor + id_usuario_receptor + fecha_hora.	
CLAVES FORÁNEAS:	id_usuario_emisor → **USUARIO (id_usuario)** id_usuario_receptor → **USUARIO (id_usuario)**	
DESCRIPCIÓN DE REGISTRO		
NOMBRE	**TIPO**	**DESCRIPCIÓN**
id_usuario_emisor	Int(7)	Código identificador de usuario que crea el mensaje. Corresponde a un número correlativo desde el 0 hasta 9.999.999.
id_usuario_receptor	Int(7)	Código identificador de usuario que recibe el mensaje. Corresponde a un número correlativo desde el 0 hasta 9.999.999.
fecha_hora	Datetime	Fecha y hora en que se genera el mensaje. Con el formato YYYY-MM-DD HH:MM:SS.
Mensaje	Varchar(250)	Contenido del mensaje realizado. Permite cualquier carácter alfanumérico y especial.
Leído	Char(1)	Representa si el mensaje fue leído por la contraparte, “s” para leído y “n” para no leído.

2.3.10. Tabla SubCategoria.

Tabla 2-10. Descripción de Tabla: SUBCATEGORIA

DESCRIPCIÓN DE TABLA		
NOMBRE:	**SUBCATEGORIA**	
DESCRIPCIÓN:	Representa los datos de una subcategoría a la cual pertenecen las publicaciones.	
TIPO DE TABLA:	Maestra.	
CLAVE PRIMARIA:	id_subcategoria	
CLAVES FORÁNEAS:	id_categoria → **CATEGORIA**	
DESCRIPCIÓN DE REGISTRO		
NOMBRE	**TIPO**	**DESCRIPCIÓN**
id_subcategoria	Int(4)	Código identificador de una sub categoría de publicaciones. Corresponde a un número correlativo desde 0 hasta 9.999.
nombre_subcategoria	Varchar(30)	Nombre sin abreviaturas de la sub categoría.
descripción_subcategoria	Varchar(200)	Descripción breve de la sub categoría.
id_categoria	Int(2)	Código identificador de una categoría a la cual pertenece una sub categoría. Corresponde a un número correlativo desde 0 hasta 99.

2.3.11. Tabla Categoría.

Tabla 2-11. Descripción de Tabla: CATEGORIA

DESCRIPCIÓN DE TABLA		
NOMBRE:	**CATEGORIA**	
DESCRIPCIÓN:	Representa los datos de una categoría, a la cual pertenecen las sub categoría.	
TIPO DE TABLA:	Maestra	
CLAVE PRIMARIA:	Id_categoria	
CLAVES FORÁNEAS:	-	
DESCRIPCIÓN DE REGISTRO		
NOMBRE	**TIPO**	**DESCRIPCIÓN**
id_categoria	Int(2)	Código identificador de una categoría. Corresponde a un número correlativo desde 0 hasta 99.
nombre_categoria	Varchar(30)	Nombre sin abreviatura de la categoría.
descripción_categoria	Varchar(200)	Descripción breve de la categoría.

2.3.12. Tabla Imagen.

Tabla 2-12. Descripción de Tabla: IMAGEN

DESCRIPCIÓN DE TABLA		
NOMBRE:	**IMAGEN**	
DESCRIPCIÓN:	Representa los datos de las imágenes asociadas a las publicaciones	
TIPO DE TABLA:	Maestra	
CLAVE PRIMARIA:	id_usuario + id_publicacion + id_imagen	
CLAVES FORÁNEAS:	id_usuario+ id_publicacion → **PUBLICACIÓN**	
DESCRIPCIÓN DE REGISTRO		
NOMBRE	**TIPO**	**DESCRIPCIÓN**
id_usuario	Int(7)	Código identificador de usuario dueño de la publicación, corresponde a un número correlativo desde el 0 hasta el 9.999.999
id_publicacion	Int(4)	Código identificador de la publicación a la cual pertenece la imagen, representada por un número correlativo para cada publicación desde el 0 al 999.
id_imagen	Int(2)	Código identificador de las imágenes, representado por un número correlativo del 1 al 20.
imagen	Char(4)	Extension de la imagen adjunta a la publicación. Ejemplo ".jpg".

2.3.13. Tabla Amigo.

Tabla 2-13. Descripción de Tabla: AMIGO

DESCRIPCIÓN DE TABLA		
NOMBRE:	**AMIGO**	
DESCRIPCIÓN:	Representa una relación de amistad entre dos usuarios. Con el fin de facilitar el contacto entre ellos.	
TIPO DE TABLA:	Maestra.	
CLAVE PRIMARIA:	id_usuario + id_usuario_amigo	
CLAVES FORÁNEAS:	id_usuario → **USUARIO** Id_usuario_amigo → **USUARIO**	
DESCRIPCIÓN DE REGISTRO		
NOMBRE	**TIPO**	**DESCRIPCIÓN**
id_usuario	Int(7)	Código identificador de usuario. Corresponde a un número correlativo desde el 0 hasta 9.999.999.
id_usuario_amigo	Int(7)	Código identificador de usuario amigo. Corresponde a un número correlativo desde el 0 hasta 9.999.999.
Estado_amigo	Char(1)	Representa el estado de la amistad. "s" para solicitud y "a" para amigos.

2.3.14. Tabla Administrador.

Tabla 2-14. Descripción de Tabla: ADMINISTRADOR

DESCRIPCIÓN DE TABLA		
NOMBRE:	**ADMINISTRADOR**	
DESCRIPCIÓN:	Representa a los administradores con los que cuenta el sistema.	
TIPO DE TABLA:	Maestra	
CLAVE PRIMARIA:	id_administrador	
CLAVES FORÁNEAS:	correo → **LOGIN**	
DESCRIPCIÓN DE REGISTRO		
NOMBRE	**TIPO**	**DESCRIPCIÓN**
id_administrador	Int(2)	Código identificador de usuario. Corresponde a un número correlativo desde el 0 hasta 99.
Nombre	Varchar(30)	Nombre del administrador en el sistema. Puede estar compuesto por letras y números, sin caracteres especiales.
correo	Varchar(50)	Dirección de correo electrónico del administrador.
Estado	Char(1)	Estado del administrador en el sistema, “v” para vigente y “c” para cancelado.
Fecha_registro	Datetime	Fecha de registro de administrador en sistema. Con el formato YYYY-MM-DD HH:MM:SS.

2.3.15. Tabla Interés.

Tabla 2-15. Descripción de Tabla: INTERES

DESCRIPCIÓN DE TABLA		
NOMBRE:	**INTERES**	
DESCRIPCIÓN:	Representa el interés de un usuario en una publicación	
TIPO DE TABLA:	Intersección	
CLAVE PRIMARIA:	id_usuario + id_publicacion + id_usuario_interesado	
CLAVES FORÁNEAS:	id_publicacion + id_usuario → **PUBLICACION** id_usuario_interesado → **USUARIO (id_usuario)**	
DESCRIPCIÓN DE REGISTRO		
NOMBRE	**TIPO**	**DESCRIPCIÓN**
Id_usuario	Int(7)	Código identificador de usuario dueño de la publicación. Corresponde a un número correlativo desde el 0 hasta 9.999.999.
id_publicacion	Int(4)	Código identificador de la publicación a la cual pertenece el interés, representada por un número correlativo para cada publicación desde el 0 al 999.
id_usuario_interesado	Int(7)	Código identificador de usuario que manifiesta el interés en la publicación. Corresponde a un número correlativo desde el 0 hasta 9.999.999.

2.3.16. Tabla Login.

Tabla 2-16. Descripción de Tabla: LOGIN

DESCRIPCIÓN DE TABLA		
NOMBRE:	**LOGIN**	
DESCRIPCIÓN:	Representa los datos de un usuario o administrador para validarse en el sistema.	
TIPO DE TABLA:	Maestra	
CLAVE PRIMARIA:	Correo	
CLAVES FORÁNEAS:	-	
DESCRIPCIÓN DE REGISTRO		
NOMBRE	**TIPO**	**DESCRIPCIÓN**
Correo	Varchar(50)	Correo electrónico del usuario o administrador para validarse en el sistema.
clave	Varchar(100)	Contraseña encriptada en SHA1 del usuario o administrador para validarse en el sistema.
Tipo	Char(1)	Representa el tipo de sesión que posee el usuario que se valida en el sistema, “u” para usuario y “a” para administrador.

2.3.17. Tabla Notificación.

Tabla 2-17. Descripción de Tabla: NOTIFICACION

DESCRIPCIÓN DE TABLA		
NOMBRE:	**NOTIFICACIÓN ***	
DESCRIPCIÓN:	Representa las notificaciones de eventos que tiene un usuario en el sistema	
TIPO DE TABLA:	Intersección	
CLAVE PRIMARIA:	id_usuario + id_notificacion	
CLAVES FORÁNEAS:	id_usuario → **USUARIO** id_usuario_contraparte → **USUARIO**	
DESCRIPCIÓN DE REGISTRO		
NOMBRE	**TIPO**	**DESCRIPCIÓN**
Id_usuario	Int(7)	Código identificador de usuario dueño de la publicación. Corresponde a un número correlativo desde el 0 hasta 9.999.999.
id_notificacion	Int(4)	Código identificador de notificación del usuario, representada por un número correlativo para cada publicación desde el 0 al 999.
id_usuario_contraparte	Int(7)	Código identificador de usuario que realiza la acción notificada al usuario. Corresponde a un número correlativo desde el 0 hasta 9.999.999.
Tipo	Tinyint(1)	Numero correlativo desde el 1 al 9 que representa el tipo de notificación que recibe el usuario.
url	Varchar(100)	Representa la dirección que el usuario debe seguir para poder ver la acción notificada.
Visto	Char(1)	Representa si el usuario ha visto la publicación, “s” para si y “n” para no.

CAPÍTULO III. DESCRIPCIÓN DE PROGRAMAS

3. DESCRIPCION DE PROGRAMAS.

3.1. DIAGRAMA DE MENÚS.

A continuación, se presenta el diagrama de menús de usuarios para el sistema a desarrollar.

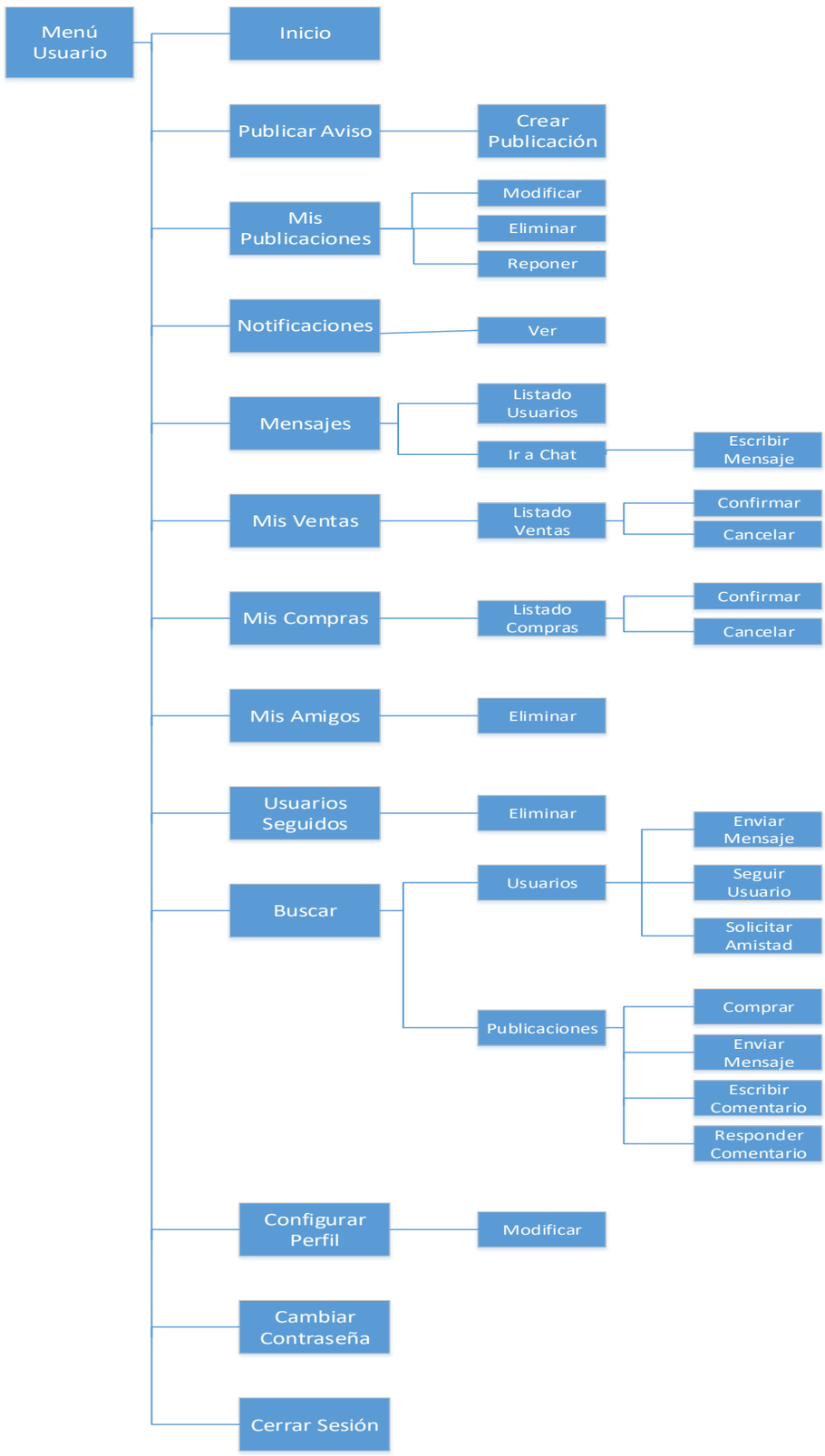

Figura 3-1. Diagrama de Menús

3.2. DIAGRAMA MODULAR.

Se presenta el diagrama modular del sistema a desarrollar.

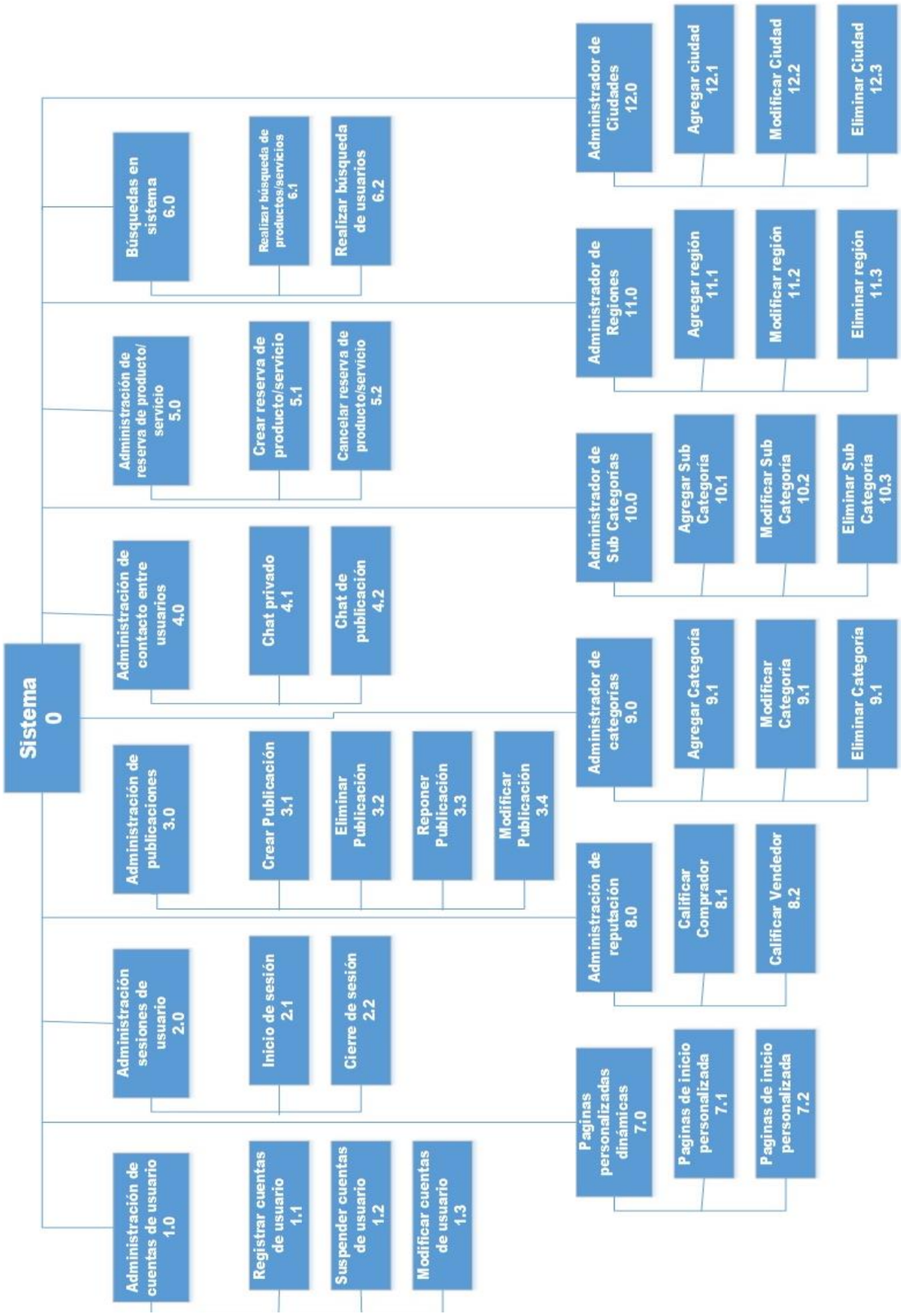

Figura 3-2. Diagrama Modular

3.3. LAYOUT DEL SITIO WEB

El diseño de pantalla, estará organizado en 3 frames o secciones: HEADER, ASIDE y BODY.

- HEADER: En esta sección se encontrarán funciones globales del sistema, tales como botón inicio, notificaciones, configuración de la cuenta, perfil, cerrar sesión etc.
- ASIDE: En esta sección serán desplegadas las preferencias de categorías de usuarios, junto con búsquedas avanzadas y amigos online.
- BODY: En esta sección se encuentra todo el contenido a desplegar dependiendo de la opción ejecutada. En el aparecerán las principales funciones de la web tales como publicaciones, detalles de compra/venta, amigos, detalles de publicaciones, etc.

Figura 3-3. Diseño Layout de la página web

3.4. ESPECIFICACIÓN DE PROGRAMAS

A continuación, se muestra una tabla con la totalidad de programas que involucrará el sistema a desarrollar, con sus respectivos objetivos y destacando aquellos que se encuentran detallados mediante (*) en el siguiente punto.

Tabla 3-1. Listado de programas.

N°	Nombre	Objetivo
1	Autentificación*	Permitir a los usuarios y administradores ingresar al sistema y registrarse en él, en caso de no estar registrado.
2	Home Usuario	Permitir generar un punto de partida para el usuario, en donde podrá realizar todas sus funcionalidades.
3	Home Administrador	Permitir generar un punto de partida para el administrador, en donde podrá realizar todas sus funcionalidades.
4	Mis publicaciones*	Mantenedor que permite gestionar las publicaciones del usuario, permitiendo crear, modificar, reponer y eliminar publicaciones.
5	Publicaciones	Permitir la navegación de los usuarios a través de las publicaciones del sistema, permitiendo buscar, generar instancias de compra y comentar.
6	Transacciones*	Permitir realizar operaciones de transacción de publicaciones, como comprar y cancelar compra.
7	Amigos	Permitir registrar o eliminar amigos en el sistema.
8	Mensajes*	Permitir enviar mensajes privados a cualquier usuario dentro del sistema.
9	Notificaciones	Generar notificaciones a los usuarios con respecto a eventos dentro del sistema que los involucran.
10	Perfiles	Permitir la navegación dentro del sistema a través de los perfiles de usuarios registrados en el sistema.
11	Preferencias	Permitir agregar categorías de publicación como preferencias a los usuarios, con el fin de filtrar y facilitar la navegación entre publicaciones.
12	Seguidores	Permite a los usuarios poder seguir la actividad de otros usuarios.
13	Categorías	Mantenedor de Categorías de publicaciones, permitiendo crear, modificar y eliminar.
14	Sub categorías	Mantenedor de Sub Categorías de publicaciones, permitiendo crear, modificar y eliminar.
15	Región	Mantenedor de Regiones, permitiendo crear, modificar y eliminar.
16	Ciudad	Mantenedor de Ciudades, permitiendo crear, modificar y eliminar.
17	Usuarios*	Mantenedor de Usuarios, permitiendo expulsar o reingresar usuarios.

3.5. DESCRIPCIÓN DETALLADA DE LOS PROGRAMA SELECCIONADOS

3.5.1. Iniciar Sesión.

Nombre Lógico: Autentificación.

Nombre Físico: Auth.php

LoginModel.php

LoginView.php

UsuarioModel.php

AdministradorModel.php

Descripción y objetivo: Permite el ingreso de los distintos usuarios al sistema. Para que el usuario pueda hacer uso del sistema debe ingresar su dirección de correo electrónico y contraseña. Si los datos ingresados son correctos dentro de la tabla Login, el usuario podrá tener acceso a las funcionalidades permitidas para su rol; caso contrario se desplegar un mensaje de alerta indicando que el correo y/o la contraseña son incorrectas**.**

Diagrama de Bloques:

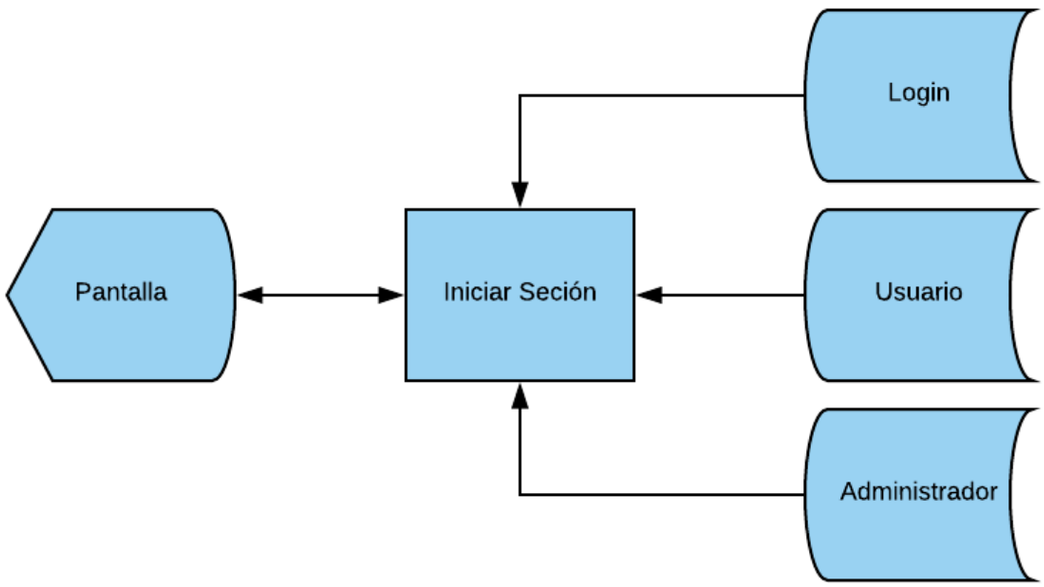

Figura 3-4. Diagrama de Bloque Iniciar Sesión

Reglas de proceso:

- El usuario ingresa al sistema.
- Si el usuario no se encuentra validado, será enviado a la pantalla de iniciar sesión.
- Se Ingresan los datos solicitados.
- Se presiona el botón Ingresar.
- El sistema buscará en la tabla Login los datos correspondientes a los datos ingresados por el usuario.
- Si los datos se corresponden con algún registro, se procederá a determinar el tipo de usuario que está intentando ingresar al sistema; por el contrario, si los datos son inválidos, se desplegará una alerta que indique que el usuario y/o contraseña son incorrectas.
- Si los datos son válidos se procederá a redirigir al usuario a la pantalla de inicio que corresponda dependiendo del tipo de usuario que ingresado.

Diseño de pantalla:

Figura 3-5. Pantalla Iniciar Sesión

3.5.2. Crear Publicación.

Nombre Lógico: Crear Publicación.

Nombre Físico: MisPublicaciones.php

PublicacionModel.php

CrearPublicacionView.php

ImagenModel.php

Descripción y objetivo: Permitir al usuario crear una nueva publicación en el sistema. Para que el usuario pueda crear una publicación debe seleccionar el botón "Publicar aviso" el cual estará disponible en la parte superior de la web. Luego deberá llenar un formulario con campos como nombre, precio, stock, descripción, etc., además de opcionalmente adjuntar fotografías que describan su producto/servicio. Al presionar el botón "Crear" el sistema validará que los campos obligatorios no estén en blanco. De estar todo correcto procederá a registrar la publicación en el sistema dentro de la tabla Publicación.

Diagrama de Bloques:

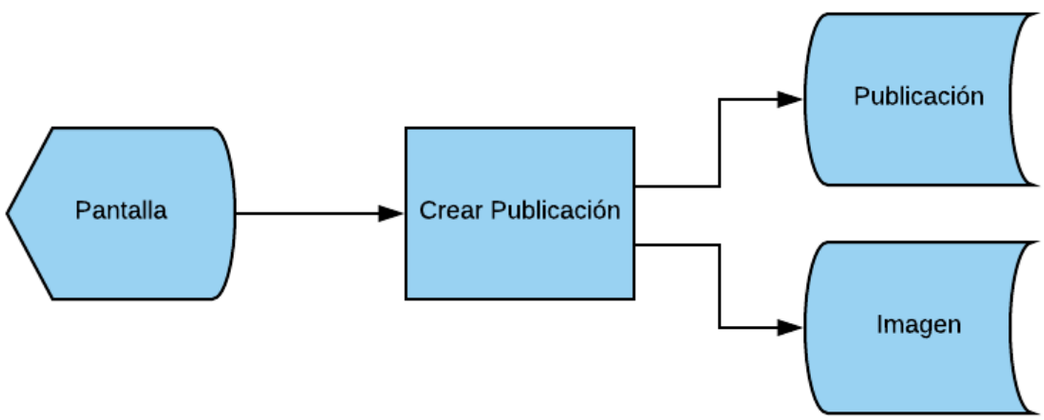

Figura 3-6. Diagrama de bloque crear publicación.

Reglas de proceso:

— El usuario presiona el botón "Publicar aviso" desde cualquier ventana del sistema.

— El sistema despliega un formulario con los campos necesarios para poder registrar una publicación. Los campos son:

- Nombre publicación (Obligatorio).

- Stock (Obligatorio).
- Precio (Obligatorio).
- Categoría (Obligatorio).
- Descripción (Obligatorio).
- Link de referencia.
- Fotografías.

— El sistema validará en primera instancia que los datos obligatorios para el formulario estén completados. Luego realizará las validaciones de formato de cada campo impidiendo ingresar un tipo de dato o una longitud incorrecta.

— Si los datos son correctos, procederá a registrar la publicación en la tabla publicación, además de registrar las fotografías en la tabla Imagen, para luego redirigir al usuario a la pantalla "Listar mis publicaciones". Caso contrario, si los datos presentan errores, se notificará al usuario con un mensaje indicando el o los campos que presentan errores.

Diseño de pantalla

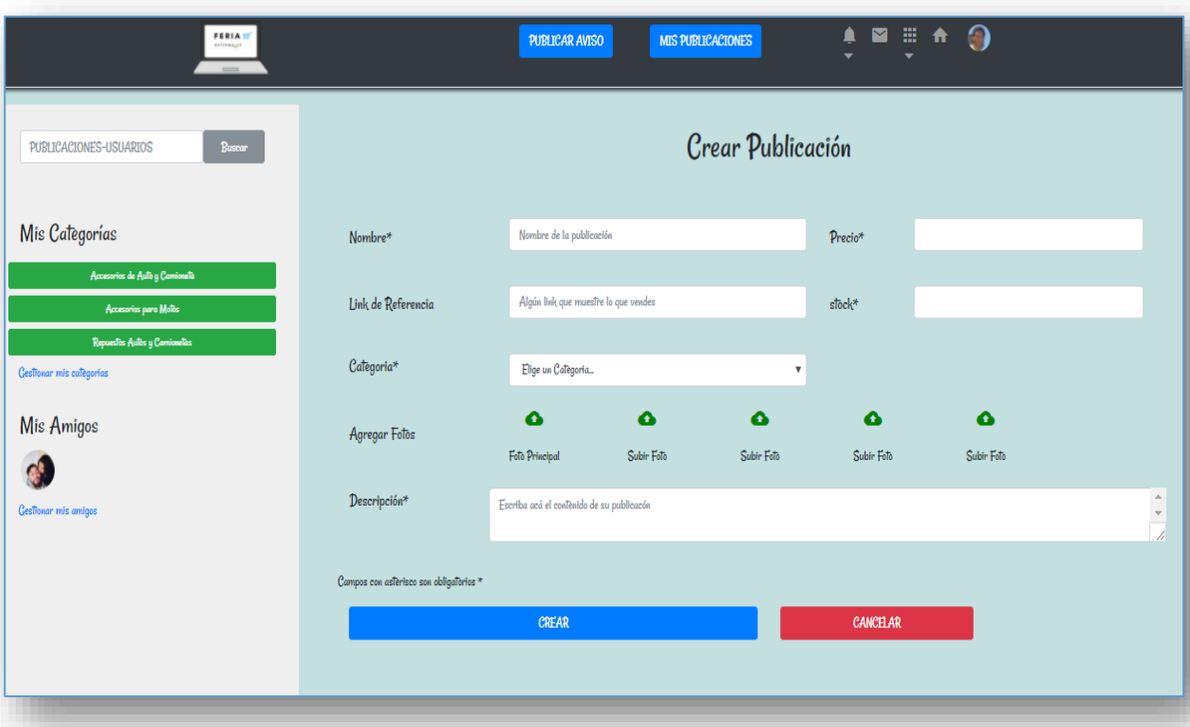

Figura 3-7. Pantalla Crear Publicación.

3.5.3. Comprar Producto/Servicio.

Nombre Lógico: Transacción

Nombre Físico: Transaccion.php

TransaccionModel.php

ComprarView.php

VenderView.php

PublicacionView.php

Notificacion.php

NotificacionModel.php

Descripción y objetivo: Permitir a los usuarios comprar productos/servicios publicados. Los usuarios podrán generar una instancia de transacción dentro del sistema. El usuario comprador selecciona una publicación la cual comprar. El vendedor recibe una notificación que informa de la intensión de compra. Luego de concretar la transacción por medios externos al sistema los usuarios dan la transacción por concretada y proceden a calificarse mutuamente.

Diagrama de bloques:

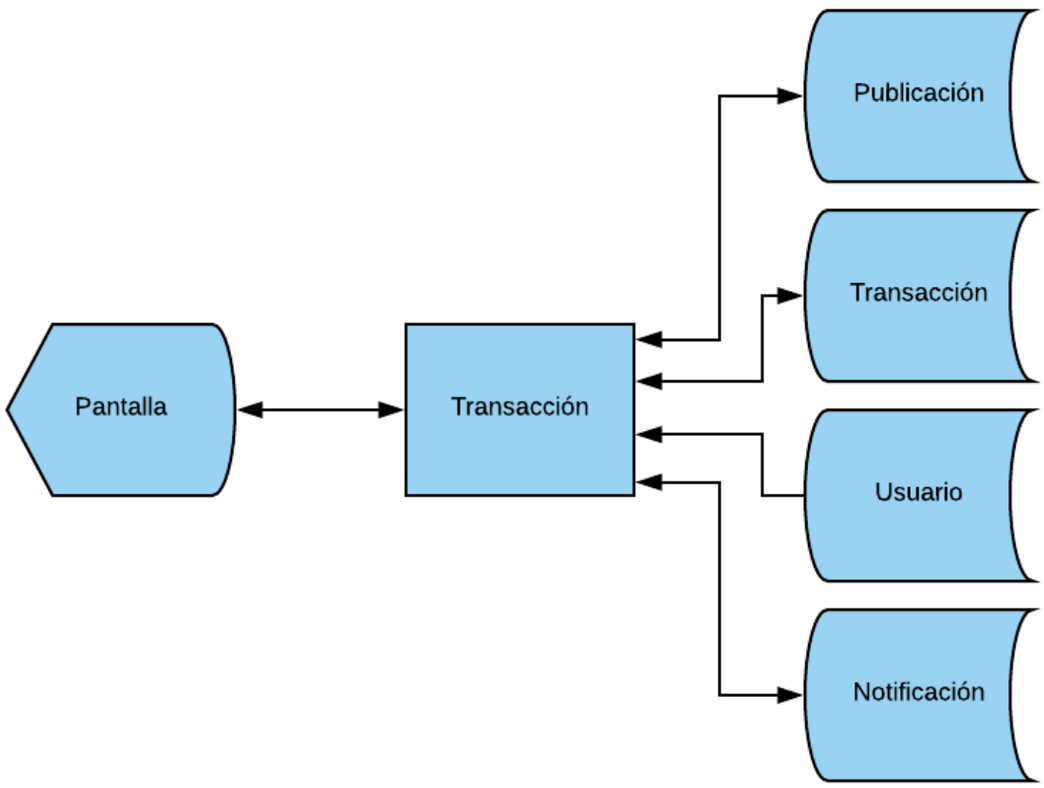

Figura 3-8. Diagrama de bloque Transacción.

Reglas de proceso:

- El usuario se posiciona en la vista “publicación” del producto/servicio que desea comprar.
- El usuario comprador pulsa el botón comprar.
- El sistema registra los datos del usuario comprador y de la publicación para generar un registro en la tabla Transacción con el estado “En proceso”. Luego se notifica al usuario vendedor de la intención de compra.
- Si la transacción es concretada exitosamente, el vendedor se sitúa en la vista “venta” para presionar el botón “Confirmar”. Luego procede a calificar al usuario comprador.
- El usuario comprador es notificado de la confirmación y procede a realizar la confirmación y calificación del usuario vendedor a través de una nueva pantalla que será desplegada.
- El sistema actualiza el registro en la tabla Transacción con el estado “Concretada” y notifica a los usuarios que la transacción fue registrada con éxito.
- Si la transacción no es concretada, cualquiera de los usuarios involucrados puede cancelar la transacción. El sistema actualizará el registro de la tabla Transacción con el estado “Cancelado”. Luego se notificará a la contraparte de la cancelación modificando el color de la campana de notificaciones a rojo.

Diseño de pantalla:

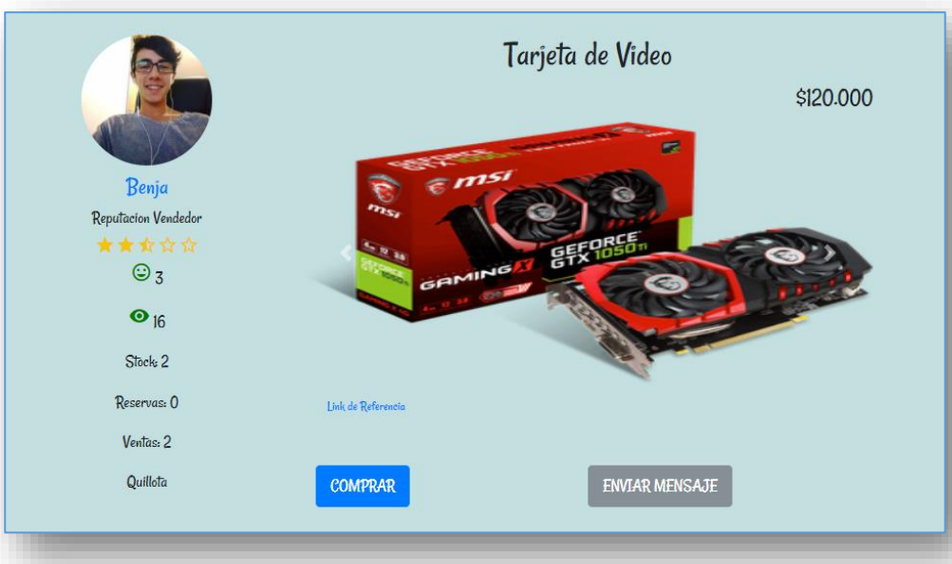

Figura 3-9. Pantalla Publicación.

Figura 3-10. Pantalla Venta.

3.5.4. Mensajes.

Nombre Lógico: Mensajes

Nombre Físico: Mensaje.php

MensajeView.php

MensajeModel.php

PublicacionView.php

PerfilView.php

Descripción y objetivo: Permitir a los usuarios enviarse mensajes privados en el sistema. Los usuarios pueden enviar un mensaje privado a otros usuarios a través de la vista de una "publicación" de un usuario o de la vista del "Perfil" de éste. Pulsando el botón "Enviar Mensaje" se ingresa a la vista "Chat" en donde se dispone de un formulario para enviar los mensajes.

Diagrama de bloques:

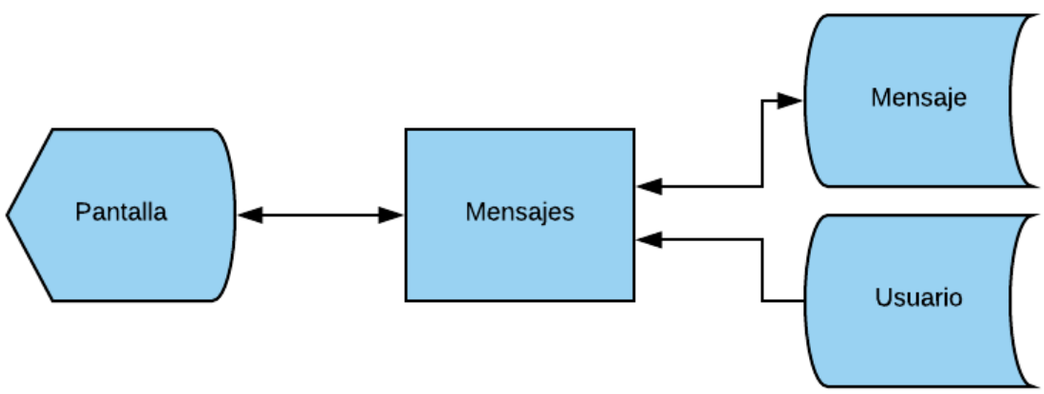

Figura 3-11. Diagrama de bloque Mensajes

Reglas del proceso:

- El usuario se posiciona en la vista “Publicación” o “Perfil” de un usuario.
- El usuario pulsa el botón “Enviar mensaje”.
- El sistema redirige al usuario a la vista “Chat”.
- Se despliega un formulario, el cual dispone de un Textarea para escribir un mensaje.
- El usuario pulsa el botón “Enviar”.
- El sistema valida que el Textarea no esté vacío y procede a registrar el mensaje en la tabla Mensaje.
- En caso de estar vacío el campo Textarea se procede a despegar un mensaje indicando el error.

Diseño de pantalla:

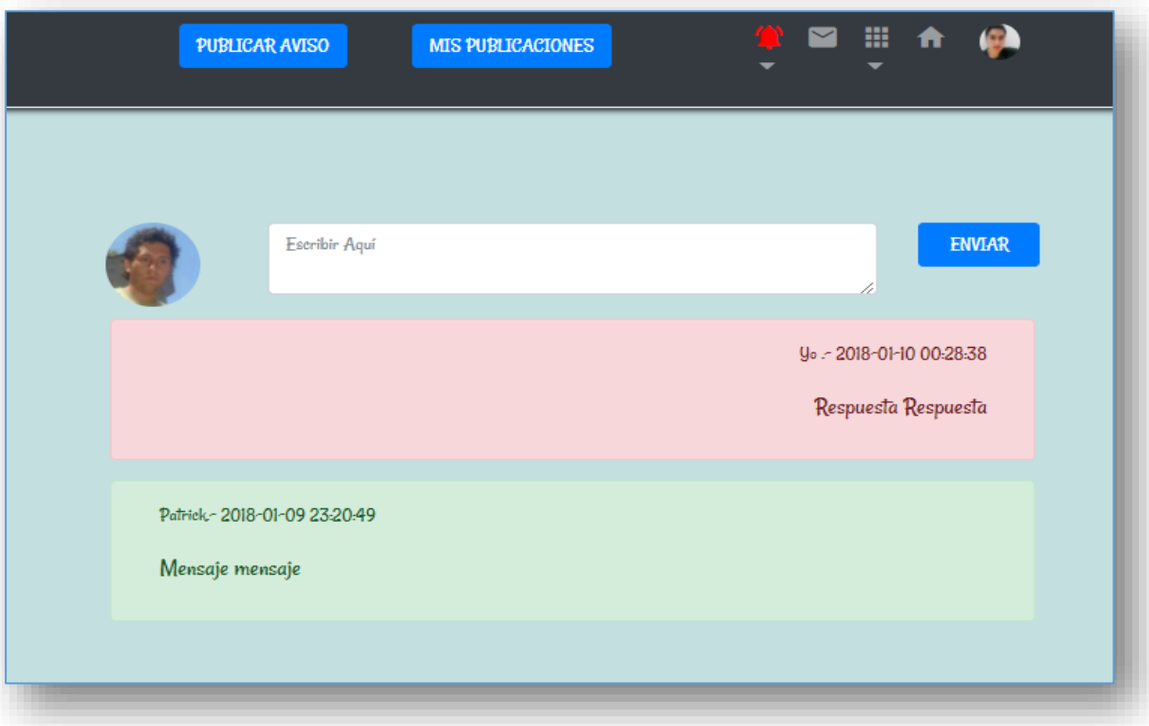

Figura 3-12. Diseño pantalla Chat

3.5.5. Usuarios.

Nombre Lógico: Usuarios

Nombre Físico: Usuario.php

ListadoUsuariosView.php

UsuarioModel.php

PublicacionModel.php

TransaccionModel.php

Descripción y objetivo: Permitir a los administradores expulsar o reactivar un usuario en el sistema. El usuario administrador podrá expulsar a un usuario activo en el sistema pulsando el botón "Expulsar". En caso de que el usuario se encuentre previamente expulsado se podrá pulsar el botón "Reactivar" para poner al usuario en estado activo.

Diagrama de bloques:

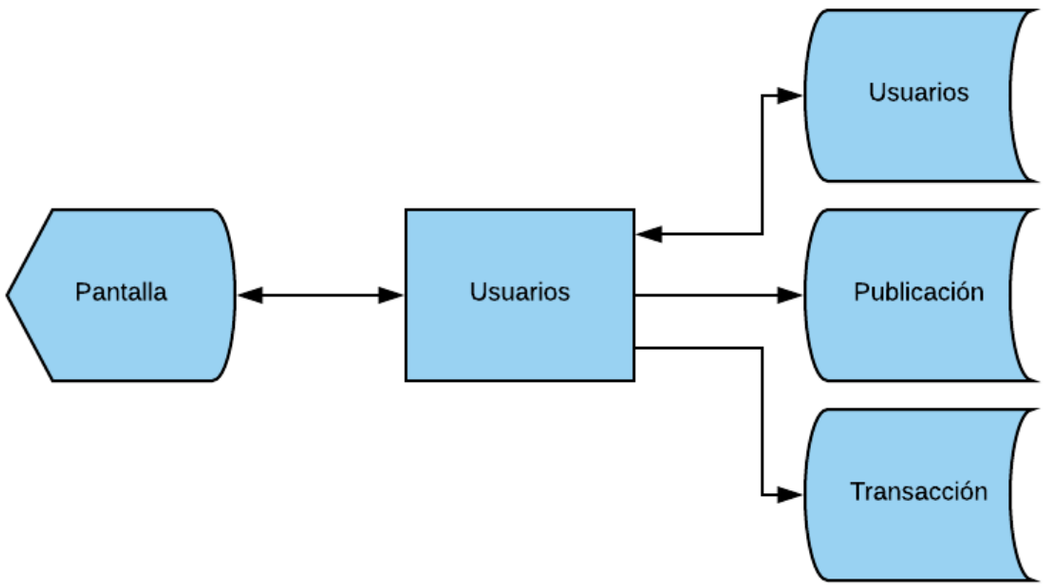

Figura 3-12. Diagrama de bloque usuarios

Reglas del proceso.

- El administrador se posiciona en la vista "Usuarios".
- El sistema desplegará un listado con todos los usuarios registrados en el sistema.
- En administrador presiona el botón "Expulsar".

— El sistema revisa si el usuario a expulsar posee publicaciones activas y transacciones en curso para proceder a eliminarlas.

— El sistema actualiza el registro del usuario en la tabla “usuario” para poner su estado en “expulsado”.

— Si el usuario se encuentra actualmente expulsado el administrador presiona el botón “Reactivar”.

— El sistema procede a actualizar el registro del usuario de la tabla Usuario para modificar su estado “Activo”.

Diseño de Pantalla

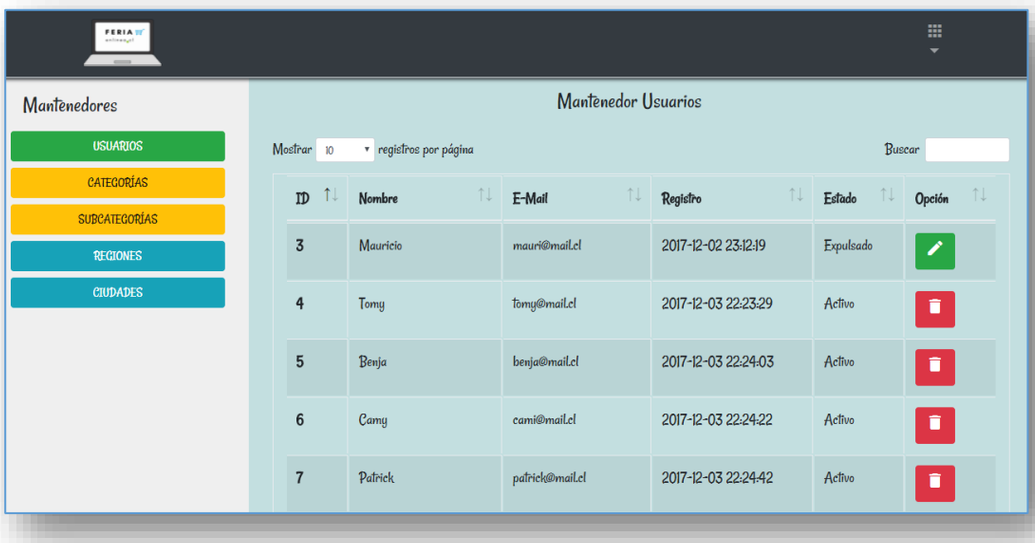

Figura 3-13. Diseño de pantalla Listado usuarios

CONCLUSIONES

La creación de este proyecto se basó en la necesidad de una competencia directa en el mercado de las compras y ventas por Internet en donde, específicamente, en Chile hay mucho potencial. Por lo que al conversar sobre este proyecto en el año 2015 el agrado sobre esta idea fue de inmediato por ambas partes, y ya en el año 2017, al momento de tener que realizar nuestro trabajo de título, utilizando todos nuestros conocimientos aprendidos en la Universidad, decidimos dar un paso adelante con el proyecto.

Nuestra mayor fortaleza fue la excelente comunicación que logramos ambos tanto en la toma de decisiones como en las innovaciones que implementamos en nuestro proyecto, además, del claro entendimiento de nuestras ideas con nuestro profesor guía lo cual nos permitió pulir grandes detalles durante el desarrollo.

La decisión más difícil que debimos tomar en nuestro proyecto, surgió una semana antes de empezar con la programación, la cual fue elegir el lenguaje de programación. Desde el inicio pensábamos en utilizar *Ruby* en conjunto con *Laravel*, pero decidimos inclinarnos por PHP y HTML gracias al óptimo manejo que poseíamos en este lenguaje. Esto nos permitió implementar todas las funcionalidades que deseábamos desde un principio.

Estamos muy satisfechos con nuestro desempeño en nuestro trabajo de título. Cumplimos con todo lo que nos propusimos desde un comienzo y de una excelente manera. Lo único que nos falta mejorar es la parte visual de nuestro proyecto, la que deseamos mejorar con el paso del tiempo y así poder lanzar nuestra primera versión para competir directamente en el mercado chileno.

BIBLIOGRAFÍA

Documentación Codeigniter 3 [en línea] 2017 [consulta constante]. Disponible en: < https://www.codeigniter.com/user_guide/>

Documentación Bootstrap 4 [en línea] 2017 [consulta constante]. Disponible en: < https://getbootstrap.com/docs/4.0/getting-started/introduction/>

ANEXO

Autentificación

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

class Auth extends CI_Controller {

  public function __construct(){
    parent::__construct();
    $this->load->model("LoginModel");
    $this->load->model("AdministradorModel");
    $this->load->model("UsuarioModel");
    $this->load->model('PreferenciasModel');
    date_default_timezone_set("America/Santiago");
    $this->load->model("AmigoModel");
    $this->load->model("SubCategoriaModel");
    $this->load->model("RegionModel");
  }

  public function index(){
    if ($this->session->userdata("login")){
      if ($this->session->tipo == 'a') {
        redirect(base_url()."admin/usuario");
      }elseif ($this->session->tipo == 'u') {
        redirect(base_url()."user/Home");
      }
    }else{
      $this->load->view("login");
    }
  }

  public function login(){
    if($this->input->post()){
      $email = $this->input->post("email");
      $clave = $this->input->post("clave");
      $login = $this->LoginModel->login($email, sha1($clave));

      if(!$login){
        $this->session->set_flashdata("error","El Mail y/o Contraseña son inconrrectas");
        redirect(base_url());
      }else{
        if ($login->tipo == "a") {
          $admin = $this->AdministradorModel->getAdministradorPorMail($email);
          if($admin->estado == "c"){
            $this->session->set_flashdata("error","Administrador tiene cuenta cancelada");
            redirect(base_url());
          }else{
            $data = array(
              'id' => $admin->id_administrador,
              'nombre' => $admin->nombre,
              'correo' => $admin->correo,
              'login' => true,
              'tipo' => $login->tipo
            );
            $this->session->set_userdata($data);
            redirect(base_url()."admin/usuario");
          }
        }elseif ($login->tipo == "u") {
          $user = $this->UsuarioModel->getUsuarioPorMail($email);
          if($user->estado == "c"){
            $this->session->set_flashdata("error","Usuario tiene cuenta cancelada");
          }elseif ($user->estado == "e"){
```

```
            $this->session->set_flashdata("error","Usuario fue expulsado");
          }else{
            $data = array(
              'id' => $user->id_usuario,
              'nombre' => $user->nombre,
              'correo' => $user->correo,
              'login' => true,
              'tipo' => $login->tipo,
              'telefono' => $user->telefono,
              'img' => $user->img_perfil,
              'id_ciudad' =>$user->id_ciudad,
              'promedio_venta' => $user->promedio_venta,
              'promedio_compra' => $user->promedio_compra,
              'amigos' => $this->AmigoModel->get9Amigos($user->id_usuario),
              'categorias' => $this->PreferenciasModel->getPreferenciasConNombre($user->id_usuario),
              'regiones' => $this->RegionModel->getRegiones(),
              'subcategorias' => $this->SubCategoriaModel->getSubCategorias()
            );
            $this->session->set_userdata($data);
            redirect(base_url()."user/Home");
          }
          redirect(base_url());
        }
      }
    }
  }

  public function registrar(){
    if($this->input->post()){
      $email = $this->input->post("email");
      $clave = $this->input->post("clave");
      $nombre = $this->input->post("nombre");
      $this->form_validation->set_rules("email","E-mail","is_unique[login.correo]");
      $this->form_validation->set_rules("email","E-mail","is_unique[usuario.correo]");
      $this->form_validation->set_rules("nombre","nombre","is_unique[usuario.nombre]");
      if($this->form_validation->run()){
        $datalogin = array(
          "correo" => ($email),
          "clave" => sha1($clave),
          "tipo" => "u"
        );
        $fecha_actual = date_format(date_create(), 'Y-m-d H:i:s');
        $dataUsuario = array(
          "correo" => $email,
          "nombre" => $nombre,
          "fecha_registro" => $fecha_actual,
          "estado" => 'v',
          "img_perfil" => 'n',
          "promedio_venta" => 5,
          "promedio_compra" => 5
        );
        if($this->LoginModel->setLogin($datalogin)){
          if($this->UsuarioModel->setUsuario($dataUsuario)){
            $user = $this->UsuarioModel->getUsuarioPorMail($email);
            $data = array(
              'id' => $user->id_usuario,
              'nombre' => $user->nombre,
              'correo' => $user->correo,
              'login' => true,
              'tipo' => 'u',
              'telefono' => "",
              'img' => "n",
              'id_ciudad' =>"",
```

```
            'promedio_venta'=> 5,
            'promedio_compra' => 5,
            'subcategorias' => $this->SubCategoriaModel->getSubCategorias(),
            'amigos' => array(),
            'categorias' => array()
          );
          $this->session->set_userdata($data);
          //conectarme por ftp para crear carpeta de usuario
          $config['hostname'] = 'localhost';
          $config['username'] = 'usuario';
          $config['password'] = '123456';
          $config['debug'] = TRUE;
          $this->ftp->connect($config);
          $this->ftp->mkdir($user->id_usuario.'/pub');
          $this->ftp->mkdir($user->id_usuario.'/perfil');
          $this->ftp->close();
          //fin cambios ftp
          redirect(base_url()."user/Perfil/configurar");
        }else{
          $this->session->set_flashdata("error","No se pudo crear la cuenta");
          redirect(base_url()."auth");
        }
      }else{
        $this->session->set_flashdata("error","No se pudo crear la cuenta");
        redirect(base_url()."auth");
      }
    }else{
      $this->session->set_flashdata("error","El E-mail y/o el usuario ya existe");
      redirect(base_url()."auth");
    }
  }
}

public function logout(){
  $this->session->sess_destroy();
  redirect(base_url());
}
}
```

```
<?php
defined('BASEPATH') OR exit("No direct script access allowed");
class LoginModel extends CI_Model{
  public function login($email,$clave){
    $this->db->where("correo", $email);
    $this->db->where("clave", $clave);
    $resultado = $this->db->get("login");

    if($resultado->num_rows() > 0){
      return $resultado->row();
    }else{
      return false;
    }
  }
  public function setLogin($data){
    return $this->db->insert("login",$data);
  }
  public function updatePassword($mail,$pass){
    $this->db->set("clave",$pass);
    $this->db->where("correo",$mail);
    return $this->db->update("login");
  }
}
```

Mis publicaciones

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

class MisPublicaciones extends CI_Controller {
  private $dataheader = array(
    "notificacion_nueva" =>false,
    "notificaciones" => array(),
    "mensaje" =>false
  );
  public function __construct(){
    parent::__construct();
    if ((!$this->session->userdata("login") || ($this->session->tipo != "u"))) {
      redirect(base_url());
    }
    //datos para notificaciones y mensajes nuevos
    $this->load->model("MensajeModel");
    $this->load->model("NotificacionModel");
    $this->dataheader["mensaje"] = $this->MensajeModel->existeMensajesNuevos($this->session->id);
    $this->dataheader["notificacion_nueva"] = $this->NotificacionModel->existeNotificacionesNuevas($this->session->id);
    $this->dataheader["notificaciones"] = $this->NotificacionModel->getNotificacionesNuevas($this->session->id);
    date_default_timezone_set("America/Santiago");
    $this->load->model('PublicacionModel');
    $this->load->model('SubCategoriaModel');
    $this->load->model('ImagenModel');
  }

  public function index(){
    $data = array(
      "pubs" => $this->PublicacionModel->getMisPublicaciones()
    );
    $this->load->view("user/layout/header",$this->dataheader);
    $this->load->view("user/layout/aside");
    $this->load->view("user/mispublicaciones/listarpublicaciones", $data);
    $this->load->view("user/layout/footer");
  }

  public function agregar(){
    $data= array(
      "subcategorias" => $this->SubCategoriaModel->getSubCategorias()
    );
    $this->load->view("user/layout/header",$this->dataheader);
    $this->load->view("user/layout/aside");
    $this->load->view("user/mispublicaciones/crearpublicacion",$data);
    $this->load->view("user/layout/footer");
  }

  public function modificar($id_publicacion){
    if ($this->PublicacionModel->getPublicacion($id_publicacion, $this->session->id)){
      $data= array(
        "subcategorias" => $this->SubCategoriaModel->getSubCategorias(),
        "publicacion" => $this->PublicacionModel->getPublicacion($id_publicacion, $this->session->id),
        "imagenes" => $this->ImagenModel->getImagenesPublicacion($id_publicacion, $this->session->id)
      );
      $this->load->view("user/layout/header",$this->dataheader);
      $this->load->view("user/layout/aside");
      $this->load->view("user/mispublicaciones/modificarpublicacion",$data);
      $this->load->view("user/layout/footer");
    }else{
      redirect(base_url());
    }

```

```
}
public function eliminar($id_publicacion){
  if ($this->PublicacionModel->getPublicacion($id_publicacion, $this->session->id)){
    $data= array(
      "subcategorias" => $this->SubCategoriaModel->getSubCategorias(),
      "publicacion" => $this->PublicacionModel->getPublicacion($id_publicacion, $this->session->id),
    );
    $this->load->view("user/layout/header",$this->dataheader);
    $this->load->view("user/layout/aside");
    $this->load->view("user/mispublicaciones/eliminarpub",$data);
    $this->load->view("user/layout/footer");
  }else{
    redirect(base_url());
  }
}

public function reponer($id_publicacion){
  if ($this->PublicacionModel->getPublicacion($id_publicacion, $this->session->id)){
    $data= array(
      "subcategorias" => $this->SubCategoriaModel->getSubCategorias(),
      "publicacion" => $this->PublicacionModel->getPublicacion($id_publicacion, $this->session->id),
    );
    $this->load->view("user/layout/header",$this->dataheader);
    $this->load->view("user/layout/aside");
    $this->load->view("user/mispublicaciones/reponerpub",$data);
    $this->load->view("user/layout/footer");
  }else{
    redirect(base_url());
  }
}

public function setPublicacion(){
  if($this->input->post()){
    $fecha_actual = date_create();
    $data = array(
      "nombre_publicacion" => $this->input->post("nombre"),
      "id_publicacion" => $this->PublicacionModel->getPosicionMaxima(),
      "id_usuario" => $this->session->id,
      "descripcion_publicacion" => $this->input->post("descripcion"),
      "link_referencia" => $this->input->post("link"),
      "precio" => $this->input->post("precio"),
      "id_subcategoria" => $this->input->post("id_categoria"),
      "stock" => $this->input->post("stock"),
      "fecha_creacion" => date_format($fecha_actual,'Y-m-d H:i:s'),
      "estado_publicacion" => 'v',
      "contador_visitas" => 0,
      "contador_interes" => 0
    );
    $erroragregar = false;
    if($this->PublicacionModel->setPublicacion($data)){
      $this->crearCarpeta($data["id_publicacion"]);
      $config['upload_path'] = './assets/img/users/'.$this->session->id.'/pub/'.$data["id_publicacion"].'/';
      $config['allowed_types'] = 'gif|jpg|png';
      $config['overwrite'] = true;
      $config['max_size'] = '4048';
      $config['max_width'] = '4024';
      $config['max_height'] = '3008';
      $this->load->library('upload',$config);
      $error_imagen = false;
      if ($_FILES["foto1"]["error"] != 4){
        $config['file_name'] = "1";
        $this->upload->initialize($config);
        if ($this->upload->do_upload("foto1")){
```

```
•           $file_info = $this->upload->data();
•           $dataimagen["extension"] = substr($file_info['file_name'],-4,4);
•           $dataimagen["id_usuario"] = $this->session->id;
•           $dataimagen["id_publicacion"] = $data["id_publicacion"];
•           $dataimagen["id_imagen"] = 1;
•           $this->ImagenModel->setImagen($dataimagen);
•           //$this->crearImagen($file_info['file_name'],$data["id_publicacion"]);
•           $this->CrearImagenPortada($file_info['file_name'],$data["id_publicacion"]);
•         }else{
•           $error_imagen = true;
•         }
•       }
•
•       if ($_FILES["foto1"]["error"] != 4){
•         $config['file_name'] = "2";
•         $this->upload->initialize($config);
•         if ($this->upload->do_upload("foto2")){
•           $nombre = "2".substr($_FILES["foto2"]["name"],-4,4);
•           $dataimagen["extension"] = substr($nombre,-4,4);
•           $dataimagen["id_usuario"] = $this->session->id;
•           $dataimagen["id_publicacion"] = $data["id_publicacion"];
•           $dataimagen["id_imagen"] = 2;
•           $this->ImagenModel->setImagen($dataimagen);
•           //$this->crearImagen($nombre,$data["id_publicacion"]);
•         }else{
•           $error_imagen = true;
•         }
•       }
•       if ($_FILES["foto1"]["error"] != 4){
•         $config['file_name'] = "3";
•         $this->upload->initialize($config);
•         if ($this->upload->do_upload("foto3")){
•           $nombre = "3".substr($_FILES["foto3"]["name"],-4,4);
•           $dataimagen["extension"] = substr($nombre,-4,4);
•           $dataimagen["id_usuario"] = $this->session->id;
•           $dataimagen["id_publicacion"] = $data["id_publicacion"];
•           $dataimagen["id_imagen"] = 3;
•           $this->ImagenModel->setImagen($dataimagen);
•           //$this->crearImagen($nombre,$data["id_publicacion"]);
•         }else{
•           $error_imagen = true;
•         }
•       }
•       if ($_FILES["foto1"]["error"] != 4){
•         $config['file_name'] = "4";
•         $this->upload->initialize($config);
•         if ($this->upload->do_upload("foto4")){
•           $nombre = "4".substr($_FILES["foto4"]["name"],-4,4);
•           $dataimagen["extension"] = substr($nombre,-4,4);
•           $dataimagen["id_usuario"] = $this->session->id;
•           $dataimagen["id_publicacion"] = $data["id_publicacion"];
•           $dataimagen["id_imagen"] = 4;
•           $this->ImagenModel->setImagen($dataimagen);
•           //$this->crearImagen($nombre,$data["id_publicacion"]);
•         }else{
•           $error_imagen = true;
•         }
•       }
•       if ($_FILES["foto1"]["error"] != 4){
•         $config['file_name'] = "5";
•         $this->upload->initialize($config);
•         if ($this->upload->do_upload("foto5")){
•           $nombre = "5".substr($_FILES["foto5"]["name"],-4,4);
```

```
•            $dataimagen["extension"] = substr($nombre,-4,4);
•            $dataimagen["id_usuario"] = $this->session->id;
•            $dataimagen["id_publicacion"] = $data["id_publicacion"];
•            $dataimagen["id_imagen"] = 5;
•            $this->ImagenModel->setImagen($dataimagen);
•            //$this->crearImagen($nombre,$data["id_publicacion"]);
•          }else{
•            $error_imagen = true;
•          }
•        }
•      }else{
•        $erroragregar = true;
•      }
•      if(!$erroragregar){
•        if (!$error_imagen){
•          $this->session->set_flashdata("mensaje","Publicación agregada correctamente");
•          redirect(base_url()."user/MisPublicaciones");
•        }else{
•            $this->session->set_flashdata("mensaje","Publicación agregada, *Algunas imagenes podrián no haber sido
agregadas*");
•          redirect(base_url()."user/MisPublicaciones");
•        }
•
•      }else{
•        $this->session->set_flashdata("error","No se pudo agregar la Publicación");
•        redirect(base_url()."user/MisPublicaciones/agregar");
•      }
•    }
•  }
•  public function updatePublicacion(){
•    if($this->input->post()){
•      $fecha_actual = date_create();
•      $id_usuario = $this->input->post("id_usuario");
•      $id_publicacion = $this->input->post("id_publicacion");
•      $data = array(
•        "nombre_publicacion" => $this->input->post("nombre"),
•        "descripcion_publicacion" => $this->input->post("descripcion"),
•        "link_referencia" => $this->input->post("link"),
•        "precio" => $this->input->post("precio"),
•        "id_subcategoria" => $this->input->post("id_categoria"),
•        "stock" => $this->input->post("stock"),
•      );
•      $erroragregar = false;
•      if($this->PublicacionModel->updatePublicacion($id_usuario, $id_publicacion, $data)){
•        $config['upload_path'] = './assets/img/users/'.$this->session->id.'/pub/'.$id_publicacion.'/';
•        $config['allowed_types'] = 'gif|jpg|png';
•        $config['overwrite'] = true;
•        $config['max_size'] = '4048';
•        $config['max_width'] = '2024';
•        $config['max_height'] = '2008';
•        $this->load->library('upload',$config);
•        $error_imagen = false;
•        if ($_FILES["foto1"]["error"] != 4){
•          $config['file_name'] = "1";
•          $this->upload->initialize($config);
•          if ($this->upload->do_upload("foto1")){
•            $file_info = $this->upload->data();
•            $dataimagen["extension"] = substr($file_info['file_name'],-4,4);
•            if ($this->ImagenModel->existeImagen($id_usuario, $id_publicacion, 1)){
•              $this->ImagenModel->UpdateImagen($id_usuario, $id_publicacion, 1, $dataimagen);
•            }else{
•              $dataimagen["id_imagen"] = 1;
•              $dataimagen["id_usuario"] = $id_usuario;
```

```
•            $dataimagen["id_publicacion"] = $id_publicacion;
•            $this->ImagenModel->setImagen($dataimagen);
•          }
•          //$this->crearImagen($file_info['file_name'],$id_publicacion);
•          $this->CrearImagenPortada($file_info['file_name'],$id_publicacion);
•        }else{
•          $error_imagen = true;
•        }
•      }
•      if ($_FILES["foto2"]["error"] != 4){
•        $config['file_name'] = "2";
•        $this->upload->initialize($config);
•        if ($this->upload->do_upload("foto2")){
•          $nombre = "2".substr($_FILES["foto2"]["name"],-4,4);
•          unset($dataimagen);
•          $dataimagen["extension"] = substr($nombre,-4,4);
•          if ($this->ImagenModel->existeImagen($id_usuario, $id_publicacion, 2)){
•            $this->ImagenModel->UpdateImagen($id_usuario, $id_publicacion, 2, $dataimagen);
•          }else{
•            $dataimagen["id_imagen"] = 2;
•            $dataimagen["id_usuario"] = $id_usuario;
•            $dataimagen["id_publicacion"] = $id_publicacion;
•            $this->ImagenModel->setImagen($dataimagen);
•          }
•          //$this->crearImagen($nombre,$data["id_publicacion"]);
•        }else{
•          $error_imagen = true;
•        }
•      }
•      if ($_FILES["foto3"]["error"] != 4){
•        $config['file_name'] = "3";
•        $this->upload->initialize($config);
•        if ($this->upload->do_upload("foto3")){
•          $nombre = "3".substr($_FILES["foto3"]["name"],-4,4);
•          unset($dataimagen);
•          $dataimagen["extension"] = substr($nombre,-4,4);
•          if ($this->ImagenModel->existeImagen($id_usuario, $id_publicacion, 3)){
•            $this->ImagenModel->UpdateImagen($id_usuario, $id_publicacion, 3, $dataimagen);
•          }else{
•            $dataimagen["id_imagen"] = 3;
•            $dataimagen["id_usuario"] = $id_usuario;
•            $dataimagen["id_publicacion"] = $id_publicacion;
•            $this->ImagenModel->setImagen($dataimagen);
•          }
•          //$this->crearImagen($nombre,$id_publicacion);
•        }else{
•          $error_imagen = true;
•        }
•      }
•      if ($_FILES["foto4"]["error"] != 4){
•        $config['file_name'] = "4";
•        $this->upload->initialize($config);
•        if ($this->upload->do_upload("foto4")){
•          $nombre = "4".substr($_FILES["foto4"]["name"],-4,4);
•          unset($dataimagen);
•          $dataimagen["extension"] = substr($nombre,-4,4);
•          if ($this->ImagenModel->existeImagen($id_usuario, $id_publicacion, 4)){
•            $this->ImagenModel->UpdateImagen($id_usuario, $id_publicacion, 4, $dataimagen);
•          }else{
•            $dataimagen["id_imagen"] = 4;
•            $dataimagen["id_usuario"] = $id_usuario;
•            $dataimagen["id_publicacion"] = $id_publicacion;
•            $this->ImagenModel->setImagen($dataimagen);
```

```
            }
            //$this->crearImagen($nombre,$id_publicacion);
          }else{
            $error_imagen = true;
          }
        }
        if ($_FILES["foto1"]["error"] != 4){
          $config['file_name'] = "5";
          $this->upload->initialize($config);
          if ($this->upload->do_upload("foto5")){
            $nombre = "5".substr($_FILES["foto5"]["name"],-4,4);
            unset($dataimagen);
            $dataimagen["extension"] = substr($nombre,-4,4);
            if ($this->ImagenModel->existeImagen($id_usuario, $id_publicacion, 5)){
              $this->ImagenModel->UpdateImagen($id_usuario, $id_publicacion, 5, $dataimagen);
            }else{
              $dataimagen["id_imagen"] = 5;
              $dataimagen["id_usuario"] = $id_usuario;
              $dataimagen["id_publicacion"] = $id_publicacion;
              $this->ImagenModel->setImagen($dataimagen);
            }
            //$this->crearImagen($nombre,$id_publicacion);
          }else{
            $error_imagen = true;
          }
        }
      }else{
        $erroragregar = true;
      }
      if(!$erroragregar){
        if (!$error_imagen){
          $this->session->set_flashdata("mensaje","Publicación modificada correctamente");
          redirect(base_url()."user/MisPublicaciones");
        }else{
         $this->session->set_flashdata("mensaje","Publicación modificada, *Algunas imagenes pudieron no haber sido agregadas*");
          redirect(base_url()."user/MisPublicaciones");
        }

      }else{
        $this->session->set_flashdata("error","No se pudo Modificar la Publicación");
        redirect(base_url()."user/MisPublicaciones/modificar/".$id_publicacion);
      }
    }
  }

  public function deletePublicacion(){
    if($this->input->post()){
      $id_usuario = $this->input->post("id_usuario");
      $id_publicacion = $this->input->post("id_publicacion");
      $this->PublicacionModel->cerrarPublicacion($id_usuario, $id_publicacion);
      $this->session->set_flashdata("error","Se ha cancelado una publicación");
      redirect(base_url()."user/MisPublicaciones");
    }
  }
  public function reponerUpdate(){
    if($this->input->post()){
      $id_usuario = $this->input->post("id_usuario");
      $stock = $this->input->post("stock");
      $id_publicacion = $this->input->post("id_publicacion");

      $this->PublicacionModel->reponerUpdate($id_usuario, $id_publicacion,$stock);
      $this->session->set_flashdata("error","Se ha cancelado una publicación");
```

```
•       redirect(base_url()."user/MisPublicaciones");
•     }
•   }
•
•   private function crearCarpeta($id){
•     //conectarme por ftp para crear carpeta de publicacion
•     $configftp['hostname'] = 'localhost';
•     $configftp['username'] = 'usuario';
•     $configftp['password'] = '123456';
•     $configftp['debug'] = TRUE;
•     $this->ftp->connect($configftp);
•     if(!file_exists("assets/img/users/".$this->session->id."/pub/".$id."/")){
•       $this->ftp->mkdir($this->session->id."/pub/".$id);
•     }
•     $this->ftp->close();
•     //fin cambios ftp
•   }
•
•   private function crearImagen($nombre_imagen,$id_publicacion){
•     $config['image_library'] = 'gd2';
•     $config['source_image'] =  './assets/img/users/'.$this->session->id.'/pub/'.$id_publicacion.'/'.$nombre_imagen;
•     //$config['create_thumb'] = true;
•     $config['maintrain_ratio'] = true;
•     $config['new_image'] = './assets/img/users/'.$this->session->id.'/pub/'.$id_publicacion.'/'.$nombre_imagen;
•     //$config['thumb_marker'] = '_250x250';
•     $config['width'] = 250;
•     $config['height'] = 150;
•     $this->load->library('image_lib',$config);
•     $this->image_lib->initialize($config);
•     $this->image_lib->resize();
•     $this->image_lib->clear();
•     }
•   private function CrearImagenPortada($nombre_imagen,$id_publicacion){
•     $config['image_library'] = 'gd2';
•     $config['source_image'] =  './assets/img/users/'.$this->session->id.'/pub/'.$id_publicacion.'/'.$nombre_imagen;
•     $config['create_thumb'] = true;
•     $config['maintrain_ratio'] = true;
•     $config['new_image'] = './assets/img/users/'.$this->session->id.'/pub/'.$id_publicacion.'/';
•     $config['thumb_marker'] = '_200x200';
•     $config['width'] = 200;
•     $config['height'] = 200;
•     $this->load->library('image_lib',$config);
•     $this->image_lib->initialize($config);
•     $this->image_lib->resize();
•     $this->image_lib->clear();
•   }
• }


• <?php
• defined('BASEPATH') OR exit("No direct script access allowed");
•
• class PublicacionModel extends CI_Model{
•
•   public function getMisPublicaciones(){
•         $this->db->select('id_usuario, id_publicacion, nombre_publicacion, fecha_creacion, fecha_cierre,
estado_publicacion');
•     $this->db->where('id_usuario',$this->session->id);
•     $this->db->order_by('fecha_creacion','DESC');
•     $resultado = $this->db->get("publicacion");
•     return $resultado->result();
•   }
•
```

```
• public function getPublicacionesUsuarios(){
• $this->db->select("publicacion.id_publicacion,usuario.id_usuario,nombre_publicacion,nombre,publicacion.id_subcategoria,nombre_subcategoria,precio,stock,promedio_venta,contador_interes,contador_visitas");
•   $this->db->from('publicacion');
•   $this->db->join('usuario','publicacion.id_usuario = usuario.id_usuario', 'inner');
•   $this->db->join('subcategoria','subcategoria.id_subcategoria = publicacion.id_subcategoria', 'inner');
•   $this->db->order_by('nombre_publicacion', 'RANDOM');
•   $this->db->where("publicacion.estado_publicacion","v");
•   $resultado = $this->db->get();
•   return $resultado->result();
• }
•
• public function getPublicacionesUsuariosBusqueda($busqueda){
• $this->db->select("publicacion.id_publicacion,usuario.id_usuario,id_ciudad,nombre_publicacion,nombre,publicacion.id_subcategoria,nombre_subcategoria,precio,stock,promedio_venta,contador_interes,contador_visitas, id_categoria");
•   $this->db->from('publicacion');
•   $this->db->join('usuario','publicacion.id_usuario = usuario.id_usuario', 'inner');
•   $this->db->join('subcategoria','subcategoria.id_subcategoria = publicacion.id_subcategoria', 'inner');
•   $this->db->order_by('nombre_publicacion', 'RANDOM');
•   $this->db->where("publicacion.estado_publicacion","v");
•   $this->db->like('nombre_publicacion',$busqueda);
•   $this->db->or_like('nombre_subcategoria', $busqueda);
•   $this->db->or_like('nombre', $busqueda);
•   $resultado = $this->db->get();
•   return $resultado->result();
• }
•
• public function getPublicacionesUsuariosDeSubCategoria($id_sub){
• $this->db->select("publicacion.id_publicacion,usuario.id_usuario,nombre_publicacion,nombre,publicacion.id_subcategoria,nombre_subcategoria,precio,stock,promedio_venta,contador_interes,contador_visitas");
•   $this->db->from('publicacion');
•   $this->db->join('usuario','publicacion.id_usuario = usuario.id_usuario', 'inner');
•   $this->db->join('subcategoria','subcategoria.id_subcategoria = publicacion.id_subcategoria', 'inner');
•   $this->db->where("publicacion.estado_publicacion","v");
•   $this->db->where("publicacion.id_subcategoria",$id_sub);
•   $this->db->order_by('nombre_publicacion', 'RANDOM');
•   $resultado = $this->db->get();
•   return $resultado->result();
• }
•
• public function getDatosParaPublicacion($id_usuario, $id_publicacion){
• $this->db->select("publicacion.id_publicacion,estado_publicacion,descripcion_publicacion,nombre_publicacion, precio, stock, contador_interes, contador_visitas, link_referencia, nombre, img_perfil, promedio_venta ,usuario.id_usuario, id_ciudad");
•   $this->db->from('usuario');
•   $this->db->join('publicacion','publicacion.id_usuario = usuario.id_usuario', 'inner');
•   $this->db->where('usuario.id_usuario',$id_usuario);
•   $this->db->where('publicacion.id_publicacion',$id_publicacion);
•   $resultado = $this->db->get();
•   return $resultado->row();
• }
•
• public function getPublicacion($id_publicacion, $id_usuario){
•   $this->db->where('id_usuario',$id_usuario);
•   $this->db->where('id_publicacion',$id_publicacion);
•   $resultado = $this->db->get('publicacion');
•   if($resultado->num_rows() > 0){
•     return $resultado->row();
•   }else{
•     return false;
•   }
• }
```

```
•
•   public function getPublicacionesDeUsuario($id_usuario){
•                                                                                                                $this->db-
>select("publicacion.id_publicacion,usuario.id_usuario,nombre_publicacion,publicacion.id_subcategoria
,nombre,nombre_subcategoria,precio,stock,promedio_venta,contador_interes,contador_visitas");
•     $this->db->from('publicacion');
•     $this->db->join('usuario','publicacion.id_usuario = usuario.id_usuario', 'inner');
•     $this->db->join('subcategoria','subcategoria.id_subcategoria = publicacion.id_subcategoria', 'inner');
•     $this->db->where("publicacion.id_usuario",$id_usuario);
•     $this->db->order_by('publicacion.fecha_creacion', 'DESC');
•     $resultado = $this->db->get();
•     return $resultado->result();
•   }
•
•   public function agregarVisita($id_publicacion, $id_usuario){
•     $this->db->set("contador_visitas","contador_visitas+1",FALSE);
•     $this->db->where("id_usuario", $id_usuario);
•     $this->db->where("id_publicacion", $id_publicacion);
•     return $this->db->update("publicacion");
•   }
•
•   public function getPosicionMaxima(){
•     $this->db->select_max('id_publicacion');
•     $this->db->where('id_usuario',$this->session->id);
•     $resultado = $this->db->get('publicacion');
•     if(!empty($resultado->row()->id_publicacion)){
•       return $resultado->row()->id_publicacion + 1;
•     }else{
•       return 1;
•     }
•   }
•
•   public function setPublicacion($data){
•     return $this->db->insert("publicacion",$data);
•   }
•
•   public function updatePublicacion($id_usuario, $id_publicacion, $data){
•     $this->db->set($data);
•     $this->db->where("id_usuario", $id_usuario);
•     $this->db->where("id_publicacion", $id_publicacion);
•     return $this->db->update("publicacion");
•   }
•
•   public function updateSubcategoria($id_nueva,$id_vieja){
•     $this->db->set("id_subcategoria",$id_nueva);
•     $this->db->where("id_subcategoria",$id_vieja);
•     return $this->db->update("publicacion");
•   }
•
•   public function cerrarPublicacionesUsuario($id_usuario){
•     $this->db->set("estado_publicacion","c");
•     $this->db->where("id_usuario", $id_usuario);
•     return $this->db->update("publicacion");
•   }
•   public function cerrarPublicacion($id_usuario,$id_publicacion){
•     $this->db->set("estado_publicacion","c");
•     $this->db->set("fecha_cierre",date_format(date_create(),'Y-m-d H:i:s'));
•     $this->db->where("id_usuario", $id_usuario);
•     $this->db->where("id_publicacion",$id_publicacion);
•     return $this->db->update("publicacion");
•   }
•
•   public function reponerUpdate($id_usuario, $id_publicacion,$stock){
```

```
    $this->db->set("estado_publicacion","v");
    $this->db->set("stock",$stock);
    $this->db->set("fecha_cierre",null);
    $this->db->where("id_usuario", $id_usuario);
    $this->db->where("id_publicacion",$id_publicacion);
    return $this->db->update("publicacion");
  }
```

Transacciones

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

class Transaccion extends CI_Controller {
  private $dataheader = array(
    "notificacion_nueva" =>false,
    "notificaciones" => array(),
    "mensaje" =>false
  );
  public function __construct(){
    parent::__construct();
    if ((!$this->session->userdata("login") || ($this->session->tipo != "u"))) {
      redirect(base_url());
    }
    //datos para notificaciones y mensajes nuevos
    $this->load->model("MensajeModel");
    $this->load->model("NotificacionModel");
    $this->dataheader["mensaje"] = $this->MensajeModel->existeMensajesNuevos($this->session->id);
    $this->dataheader["notificacion_nueva"] = $this->NotificacionModel->existeNotificacionesNuevas($this->session->id);
    $this->dataheader["notificaciones"] = $this->NotificacionModel->getNotificacionesNuevas($this->session->id);
    date_default_timezone_set("America/Santiago");
    $this->load->model("RegistroTransaccionModel");
    $this->load->model("UsuarioModel");
    $this->load->model("PublicacionModel");
    $this->load->model("Ciudadmodel");
  }

  public function Solicitud(){
    if($this->input->post()){
      $id_usuario = $this->input->post("id_usuario");
      $id_usuario_comprador = $this->input->post("id_usuario_comprador");
      $id_publicacion = $this->input->post("id_publicacion");
      $fecha = date_format(date_create(),'Y-m-d H:i:s');
      $id_transaccion = $this->RegistroTransaccionModel->asignarID($id_usuario,$id_publicacion);
      $data = array(
        "id_usuario" => $id_usuario,
        "id_publicacion" => $id_publicacion,
        "id_transaccion" => $id_transaccion,
        "id_usuario_comprador" => $id_usuario_comprador,
        "fecha_hora" => $fecha,
        "estado_transaccion" => "v"
      );
      $data_notificacion = array(
        "id_usuario" => $id_usuario,
        "id_notificacion" => $this->NotificacionModel->asignarID($id_usuario),
        "tipo" => 3,
        "url" => base_url()."user/transaccion/Venta/".$id_usuario."/".$id_publicacion."/".$id_transaccion,
        "visto" => "n"
      );
      if($this->RegistroTransaccionModel->setSolicitud($data)){
        $this->NotificacionModel->setNotificacion($data_notificacion);
        redirect(base_url()."user/transaccion/compra/".$id_usuario."/".$id_publicacion."/".$id_transaccion);
```

```
    }
  }
}

public function Ventas(){
  $data = array(
    "ventas" => $this->RegistroTransaccionModel->getVentasUsuario($this->session->id)
  );
  $this->load->view("user/layout/header",$this->dataheader);
  $this->load->view("user/layout/aside");
  $this->load->view("user/mistransacciones/listadoventas",$data);
  $this->load->view("user/layout/footer");
}
public function Compras(){
  $data = array(
    "ventas" => $this->RegistroTransaccionModel->getComprasUsuario($this->session->id)
  );
  $this->load->view("user/layout/header",$this->dataheader);
  $this->load->view("user/layout/aside");
  $this->load->view("user/mistransacciones/listadocompras",$data);
  $this->load->view("user/layout/footer");
}

public function Compra($id_usuario,$id_publicacion,$id_transaccion){
  $transaccion = $this->RegistroTransaccionModel->getTransaccion($id_usuario, $id_publicacion, $id_transaccion);
  $publicacion = $this->PublicacionModel->getPublicacion($id_publicacion, $id_usuario);
  $usuario = $this->UsuarioModel->getDatosPerfil($id_usuario);

  if(!empty($transaccion)){
    if($transaccion->id_usuario_comprador != $this->session->id){
      redirect(base_url());
    }
    $data = array(
      "publicacion" => $publicacion,
      "transaccion" => $transaccion,
      "usuario" => $usuario
    );
    $this->load->view("user/layout/header",$this->dataheader);
    $this->load->view("user/layout/aside");
    $this->load->view("user/mistransacciones/compra",$data);
    $this->load->view("user/layout/footer");
  }else{
    redirect(base_url());
  }
}

public function venta($id_usuario,$id_publicacion,$id_transaccion){
  if($id_usuario != $this->session->id){
    redirect(base_url());
  }
  $transaccion = $this->RegistroTransaccionModel->getTransaccion($id_usuario, $id_publicacion, $id_transaccion);
  $publicacion = $this->PublicacionModel->getPublicacion($id_publicacion, $id_usuario);

  if(!empty($transaccion)){
    $usuario = $this->UsuarioModel->getDatosPerfil($transaccion->id_usuario_comprador);

    $data = array(
      "publicacion" => $publicacion,
      "transaccion" => $transaccion,
      "usuario" => $usuario
    );
```

```
•
•       $this->load->view("user/layout/header",$this->dataheader);
•       $this->load->view("user/layout/aside");
•       $this->load->view("user/mistransacciones/venta",$data);
•       $this->load->view("user/layout/footer");
•
•     }else{
•       redirect(base_url());
•     }
•   }
•
•   public function confirmarVenta(){
•     if($this->input->post()){
•       $id_usuario = $this->input->post("id_usuario");
•       $id_publicacion = $this->input->post("id_publicacion");
•       $id_transaccion = $this->input->post("id_transaccion");
•       $calificacion_vendedor = $this->input->post("calificacion_vendedor");
•       $cantidad = $this->input->post("cantidad");
•       $transaccion = $this->RegistroTransaccionModel->getTransaccion($id_usuario, $id_publicacion,
$id_transaccion);
•       $publicacion = $this->PublicacionModel->getPublicacion($id_publicacion, $id_usuario);
•       if($publicacion->stock >= $cantidad){
•         $data = array(
•           "calificacion_vendedor" => $calificacion_vendedor,
•           "cantidad" => $cantidad
•         );
•         $this->RegistroTransaccionModel->updateTransaccion($id_usuario, $id_publicacion, $id_transaccion,$data);
•         $data_notificacion = array(
•           "id_usuario" => $transaccion->id_usuario_comprador,
•           "id_notificacion" => $this->NotificacionModel->asignarID($transaccion->id_usuario_comprador),
•           "tipo" => 5,
•           "url" => base_url()."user/transaccion/compra/".$id_usuario."/".$id_publicacion."/".$id_transaccion,
•           "visto" => "n"
•         );
•         $this->NotificacionModel->setNotificacion($data_notificacion);
•         $this->session->set_flashdata("mensaje","Venta confirmada por vendedor");
•         redirect(base_url()."user/transaccion/venta/".$id_usuario."/".$id_publicacion."/".$id_transaccion);
•       }else{
•         $this->session->set_flashdata("error","La cantidad que se quiere vender es menor al stock de la publicación");
•         redirect(base_url()."user/transaccion/venta/".$id_usuario."/".$id_publicacion."/".$id_transaccion);
•       }
•     }
•   }
•
•   public function confirmarCompra(){
•     if($this->input->post()){
•       $id_usuario = $this->input->post("id_usuario");
•       $id_publicacion = $this->input->post("id_publicacion");
•       $id_transaccion = $this->input->post("id_transaccion");
•       $calificacion_comprador = $this->input->post("calificacion_comprador");
•       $transaccion = $this->RegistroTransaccionModel->getTransaccion($id_usuario, $id_publicacion,
$id_transaccion);
•       $publicacion = $this->PublicacionModel->getPublicacion($id_publicacion, $id_usuario);
•       if($transaccion->cantidad > $publicacion->stock){
•         $data = array(
•           "estado_transaccion" => "c",
•           "fecha_hora" => date_format(date_create(),'Y-m-d H:i:s'),
•           "observacion" => "Cancelada vendedor no tiene suficiente stock"
•         );
•         $this->RegistroTransaccionModel->updateTransaccion($id_usuario, $id_publicacion, $id_transaccion,$data);
•         $this->session->set_flashdata("mensaje","Venta Cancelada por falta de stock");
•         redirect(base_url()."user/transaccion/compra/".$id_usuario."/".$id_publicacion."/".$id_transaccion);
•         $data_notificacion = array(
```

```
•          "id_usuario" => $id_usuario,
•          "id_notificacion" => $this->NotificacionModel->asignarID($this->input->post("id_usuario")),
•          "tipo" => 4,
•          "url" => base_url()."user/transaccion/venta/".$id_usuario."/".$id_publicacion."/".$id_transaccion,
•          "visto" => "n"
•        );
•        $this->NotificacionModel->setNotificacion($data_notificacion);
•      }
•      $data = array(
•        "calificacion_comprador" => $calificacion_comprador,
•        "estado_transaccion" => "f",
•        "fecha_hora" => date_format(date_create(),'Y-m-d H:i:s'),
•        "observacion" => "Transaccion Concretada exitosamente"
•      );
•      $this->RegistroTransaccionModel->updateTransaccion($id_usuario, $id_publicacion, $id_transaccion,$data);
•      $data_notificacion = array(
•         "id_usuario" => $id_usuario,
•         "id_notificacion" => $this->NotificacionModel->asignarID($this->input->post("id_usuario")),
•         "tipo" => 6,
•         "url" => base_url()."user/transaccion/venta/".$id_usuario."/".$id_publicacion."/".$id_transaccion,
•         "visto" => "n"
•      );
•      $this->NotificacionModel->setNotificacion($data_notificacion);
•      if($publicacion->stock == $transaccion->cantidad){
•        $data_pub["estado_publicacion"] = "c";
•        $data_notificacion = array(
•          "id_usuario" => $id_usuario,
•          "id_notificacion" => $this->NotificacionModel->asignarID($this->input->post("id_usuario")),
•          "tipo" => 9,
•          "url" => base_url()."user/MisPublicaciones/reponer/".$id_publicacion,
•          "visto" => "n"
•        );
•        $this->NotificacionModel->setNotificacion($data_notificacion);
•        //cancelar otras reservas
•                                                $otrastransacciones = $this->RegistroTransaccionModel-
>getTransaccionesPublicacion($id_usuario,$id_publicacion);
•        foreach ($otrastransacciones as $ot) {
•          $data_ot = array(
•            "estado_transaccion" => "c",
•            "observacion" => "Publicacion cancelada por cierre de publicación",
•            "fecha_hora" => date_format(date_create(),'Y-m-d H:i:s'),
•          );
•                  $this->RegistroTransaccionModel->updateTransaccion($ot->id_usuario, $ot->id_publicacion, $ot-
>id_transaccion,$data_ot);
•          $data_notificacion = array(
•            "id_usuario" => $ot->id_usuario_comprador,
•            "id_notificacion" => $this->NotificacionModel->asignarID($ot->id_usuario_comprador),
•            "tipo" => 4,
•                    "url" => base_url()."user/transaccion/compra/".$ot->id_usuario."/".$ot->id_publicacion."/".$ot-
>id_transaccion,
•            "visto" => "n"
•          );
•          $this->NotificacionModel->setNotificacion($data_notificacion);
•        }
•      }
•      $data_pub["stock"] = $publicacion->stock - $transaccion->cantidad;
•      $this->PublicacionModel->updatePublicacion($id_usuario,$id_publicacion,$data_pub);
•
•      //recalcular calificaciones
•      $data_vendedor = array(
•        "promedio_venta" => round($this->RegistroTransaccionModel->promedioVenta($id_usuario))
•      );
•      $data_comprador = array(
```

```
•                "promedio_compra"   =>   round($this->RegistroTransaccionModel->promedioCompra($transaccion-
>id_usuario_comprador))
•      );
•      print_r($data_comprador);
•      $this->UsuarioModel->updateUsuarioGlobal($id_usuario,$data_vendedor);
•      $this->UsuarioModel->updateUsuarioGlobal($transaccion->id_usuario_comprador,$data_comprador);
•
•      $this->session->set_flashdata("mensaje","Venta concretada");
•      redirect(base_url()."user/transaccion/compra/".$id_usuario."/".$id_publicacion."/".$id_transaccion);
•     }
•   }
•
•   public function cancelar(){
•     if($this->input->post()){
•      $id_usuario = $this->input->post("id_usuario");
•      $id_publicacion = $this->input->post("id_publicacion");
•      $id_transaccion = $this->input->post("id_transaccion");
•              $transaccion   =   $this->RegistroTransaccionModel->getTransaccion($id_usuario,   $id_publicacion,
$id_transaccion);
•      $data = array(
•        "estado_transaccion" => "c",
•        "fecha_hora" => date_format(date_create(),'Y-m-d H:i:s'),
•      );
•      if($id_usuario == $this->session->id){
•        $data["observacion"] = "Cancelada por el vendedor";
•
•        $data_notificacion = array(
•           "id_usuario" => $transaccion->id_usuario_comprador,
•           "id_notificacion" => $this->NotificacionModel->asignarID($transaccion->id_usuario_comprador),
•           "tipo" => 4,
•           "url" => base_url()."user/transaccion/compra/".$id_usuario."/".$id_publicacion."/".$id_transaccion,
•           "visto" => "n"
•        );
•        $esvendedor = true;
•      }else{
•        $data["observacion"] = "Cancelada por el comprador";
•        $data_notificacion = array(
•           "id_usuario" => $id_usuario,
•           "id_notificacion" => $this->NotificacionModel->asignarID($id_usuario),
•           "tipo" => 4,
•           "url" => base_url()."user/transaccion/venta/".$id_usuario."/".$id_publicacion."/".$id_transaccion,
•           "visto" => "n"
•        );
•        $esvendedor = false;
•      }
•
•      $this->NotificacionModel->setNotificacion($data_notificacion);
•      $this->RegistroTransaccionModel->updateTransaccion($id_usuario, $id_publicacion, $id_transaccion, $data);
•      if($esvendedor){
•        $this->session->set_flashdata("mensaje","Venta Cancelada");
•        redirect(base_url()."user/transaccion/venta/".$id_usuario."/".$id_publicacion."/".$id_transaccion);
•      }else{
•        $this->session->set_flashdata("mensaje","Compra Cancelada");
•        redirect(base_url()."user/transaccion/compra/".$id_usuario."/".$id_publicacion."/".$id_transaccion);
•      }
•     }
•   }
•  }

•  <?php
•  defined('BASEPATH') OR exit("No direct script access allowed");
•
•  class RegistroTransaccionModel extends CI_Model{
```

```
•
•   public function getVentasUsuario($id){
•   $this->db->select("registro_transaccion.id_usuario,id_transaccion, id_usuario_comprador,registro_transaccion.id_publicacion, registro_transaccion.fecha_hora, registro_transaccion.estado_transaccion, publicacion.nombre_publicacion");
•     $this->db->from("registro_transaccion");
•   $this->db->join("publicacion","registro_transaccion.id_usuario = publicacion.id_usuario and registro_transaccion.id_publicacion = publicacion.id_publicacion","inner");
•     $this->db->where("registro_transaccion.id_usuario",$id);
•     $resultado = $this->db->get();
•     return $resultado->result();
•   }
•
•   public function getComprasUsuario($id){
•   $this->db->select("registro_transaccion.id_usuario,id_transaccion,registro_transaccion.id_publicacion, registro_transaccion.fecha_hora, registro_transaccion.estado_transaccion, publicacion.nombre_publicacion");
•     $this->db->from("registro_transaccion");
•   $this->db->join("publicacion","registro_transaccion.id_usuario = publicacion.id_usuario and registro_transaccion.id_publicacion = publicacion.id_publicacion","inner");
•     $this->db->where("registro_transaccion.id_usuario_comprador",$id);
•     $resultado = $this->db->get();
•     return $resultado->result();
•   }
•
•   public function getTransaccion($id_usuario, $id_publicacion, $id_transaccion){
•     $this->db->where("id_usuario",$id_usuario);
•     $this->db->where("id_publicacion",$id_publicacion);
•     $this->db->where("id_transaccion",$id_transaccion);
•     $resultado = $this->db->get("registro_transaccion");
•     if ($resultado->num_rows() > 0){
•       return $resultado->row();
•     }else{
•       return false;
•     }
•   }
•
•   public function getTransaccionesPublicacion($id_usuario,$id_publicacion){
•     $this->db->where("id_usuario",$id_usuario);
•     $this->db->where("id_publicacion",$id_publicacion);
•     $this->db->where("estado_transaccion","v");
•     $resultado = $this->db->get("registro_transaccion");
•     return $resultado->result();
•   }
•
•   public function asignarID($id_usuario,$id_publicacion){
•     $this->db->select_max("id_transaccion");
•     $this->db->where("id_usuario",$id_usuario);
•     $this->db->where("id_publicacion",$id_publicacion);
•     $resultado = $this->db->get("registro_transaccion");
•     if($resultado->row()->id_transaccion > 0){
•       return $resultado->row()->id_transaccion + 1;
•     }else{
•       return 1;
•     }
•   }
•
•   public function updateTransaccion($id_usuario, $id_publicacion, $id_transaccion, $data){
•     $this->db->set($data);
•     $this->db->where("id_usuario",$id_usuario);
•     $this->db->where("id_publicacion",$id_publicacion);
•     $this->db->where("id_transaccion",$id_transaccion);
•     return $this->db->update("registro_transaccion");
•   }
```

```
•
•   public function setSolicitud($data){
•     return $this->db->insert("registro_transaccion",$data);
•   }
•
•   public function promedioCompra($id_usuario){
•     $this->db->select_avg('calificacion_vendedor');
•     $this->db->where("id_usuario_comprador",$id_usuario);
•     $resultado = $this->db->get("registro_transaccion");
•     return $resultado->row()->calificacion_vendedor;
•   }
•
•   public function promedioVenta($id_usuario){
•     $this->db->select_avg('calificacion_comprador');
•     $this->db->where("id_usuario",$id_usuario);
•     $resultado = $this->db->get("registro_transaccion");
•     return $resultado->row()->calificacion_comprador;
•   }
•
•   public function contarReservas($id_usuario,$id_publicacion){
•     $this->db->where("id_usuario",$id_usuario);
•     $this->db->where("id_publicacion",$id_publicacion);
•     $this->db->where("estado_transaccion","v");
•     $this->db->from("registro_transaccion");
•     return $this->db->count_all_results();
•   }
•
•   public function contarVentas($id_usuario,$id_publicacion){
•     $this->db->where("id_usuario",$id_usuario);
•     $this->db->where("id_publicacion",$id_publicacion);
•     $this->db->where("estado_transaccion","f");
•     $this->db->from("registro_transaccion");
•     return $this->db->count_all_results();
•   }
•
•   public function tieneReserva($id_usuario,$id_publicacion,$id_comprador){
•     $this->db->where("id_usuario",$id_usuario);
•     $this->db->where("id_publicacion",$id_publicacion);
•     $this->db->where("id_usuario_comprador",$id_comprador);
•     $this->db->where("estado_transaccion","v");
•     $resultado = $this->db->get("registro_transaccion");
•     if($resultado->num_rows()>0){
•       return true;
•     }else{
•       return false;
•     }
•   }
•
•   public function cancelarVentasUsuario($id,$data){
•     $this->db->set($data);
•     $this->db->where("id_usuario",$id);
•     $this->db->where("estado_transaccion","v");
•     return $this->db->update("registro_transaccion");
•   }
•   public function cancelarComprasUsuario($id,$data){
•     $this->db->set($data);
•     $this->db->where("id_usuario_comprador",$id);
•     $this->db->where("estado_transaccion","v");
•     return $this->db->update("registro_transaccion");
•   }
• }
```