UNIVERSIDAD AUTÓNOMA DE CIUDAD JUÁREZ

Instituto de Ingeniería y Tecnología

Departamento de Ingeniería Eléctrica y Computación

PROTOTIPO DE RED SOCIAL: MKWISH

Reporte Técnico de Investigación presentado por:

Jesús Ayala Díaz De León 118031

Irving Joshua Muñoz Zepeda 121094

Requisito para la obtención del título de

INGENIERO EN SISTEMAS COMPUTACIONALES

Asesor

Mtro. Saúl González Campos

Ciudad Juárez, Chihuahua Mayo de 2018

Ciudad Juárez, Chihuahua a 19 de mayo de 2018

Asunto: Liberación de Asesoría

Ing. Jesús Armando Gándara Fernández

Jefe del Departamento de Ingeniería

Eléctrica y Computación

Presente.-

Por medio de la presente nos permitimos comunicarle que después de haber realizado las asesorías correspondientes al reporte técnico **Prototipo de red social MkWish**, de los alumnos **Irving Joshua Muñoz Zepeda** y **Jesús Ayala Díaz de León,** de la Licenciatura en Ingeniería en Sistemas Computacionales, consideramos que lo han concluido satisfactoriamente, por lo que pueden continuar con los trámites de titulación intracurricular.

Sin más por el momento, reciba un cordial saludo.

Atentamente

Mtro. Saúl González Campos

Profesor Investigador IIT

UACJ
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA Y COMPUTACIÓN

Ccp. Coordinador del programa de sistemas computacionales

Alumnos

Archivo

Ciudad Juárez, Chihuahua a 19 de mayo de 2018

Asunto: Autorización de impresión

C. Irving Joshua Muñoz Zepeda

Presente.-

En virtud de que cumple satisfactoriamente los requisitos solicitados, informo a usted que se autoriza la impresión del proyecto de **Prototipo de red social MkWish,** para presentar los resultados del proyecto de titulación con el propósito de obtener el título de Licenciado en Ingeniería en Sistemas Computacionales.

Sin otro particular, reciba un cordial saludo.

M. en C. José Fernando Estrada Saldaña

Profesor Seminario de Titulación II

Ciudad Juárez, Chihuahua a 19 de mayo de 2018

Asunto: Autorización de impresión

C. Jesús Ayala Díaz de León

Presente.-

En virtud de que cumple satisfactoriamente los requisitos solicitados, informo a usted que se autoriza la impresión del proyecto de **Prototipo de red social MkWish,** para presentar los resultados del proyecto de titulación con el propósito de obtener el título de Licenciado en Ingeniería en Sistemas Computacionales.

Sin otro particular, reciba un cordial saludo.

M. en C. José Fernando Estrada Saldaña

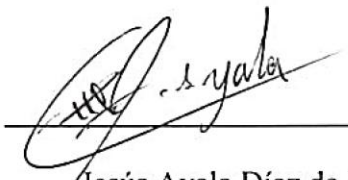
Profesor Seminario de Titulación II

Declaración de Originalidad

Nosotros, Irving Joshua Muñoz Zepeda y Jesús Ayala Díaz de León, declaramos que el material contenido en esta publicación fue generado con la revisión de los documentos que se mencionan en la sección de Referencias y que no ha sido utilizado para obtener otro título o reconocimiento en otra institución de educación superior.

Irving Muñoz

Irving Joshua Muñoz Zepeda



Jesús Ayala Díaz de León

Agradecimientos

Se agradece a la empresa Koala Workshop, lugar de donde se aprendieron múltiples conocimientos laborales que ayudaron a hacer posible la realización de este proyecto bajo la administración del Ing. Jorge Arturo Galicia Ramos; las enseñanzas de los hermanos e ingenieros Alejandro Muñoz y Omar Muñoz.

Se agradece al equipo de proyectos del departamento de CTG de la empresa Robert Bosch planta Zaragoza, por ayudarnos a mejorar en el uso de diversas herramientas utilizadas en la elaboración de este proyecto.

Se agradece también la asesoría y enseñanza del Ing. Miguel Alfredo Valdés Arias quien aconsejó alternativas y rumbos para este proyecto.

Se agradece la colaboración profesional del Lic. Adrián Torres Díaz de León, quién colaboró con el diseño visual del proyecto.

Se agradece la asesoría del Mtro. Saúl González Campos, persona que supervisó el rumbo del proyecto y la realización del presente documento.

Se reconoce a estas personas con la intención de agradecer por compartir su conocimiento en las disciplinas y ramas necesarias para concluir el proyecto de titulación.

Dedicatoria

A los familiares ya no presentes. Aunque quizás sí en espíritu, muy en especial al Sr. Padre de familia Jesús Ayala Aguilar, quien probablemente hubiese atesorado el presente documento.

A los familiares: La Sra. Madre de familia María Concepción Díaz de León Cortez, hermanos, Ing. Pedro Alberto Torres Díaz de León, Lic. Adrián Torres Díaz de León y hermana; futura Lic. Estefanía Navarro Trespalacios.

Para la familia Muñoz Hidalgo, Muñoz Loya y Muñoz Muñoz por siempre apoyar este proyecto desde su inicio.

Para Rubén Rodríguez, Erik Vargas y Daniel Armendáriz por su amistad y las esperanzas que proporcionaban.

Para Luz Barrón y Cesar Carrillo por ser siempre modelos a seguir y consejeros cuando las cosas parecían difíciles.

Índice de Contenido

Índice de figuras

Resumen

El prototipo de red social propuesto a continuación busca ayudar a aquellos usuarios que les molesta encontrar publicaciones que no les dejan nada provechoso en sus vidas, aquellas publicaciones que ocupan el espacio donde puede estar la respuesta a algún problema que pueden llegar a tener en el futuro o de aquellas que ayuden a obtener un conocimiento nuevo, por muy pequeño que sea. Se propuso una metodología que apoye a disminuir este problema y como resultado se obtuvo un algoritmo capaz de clasificar correctamente cuáles son las publicaciones que están teniendo un impacto positivo en sus espectadores y cuáles es mejor dejar en segundo plano, ya que su existencia se puede considerar no muy relevante para los demás usuarios.

Introducción

Este proyecto nace después de ser usuarios de las múltiples redes sociales que han gozado de popularidad en los años más recientes y encontrar que, a pesar de que cada una tiene su estilo y una dinámica en la que pueden interactuar las personas que tienen una cuenta en ella, es muy raro que una plataforma de este estilo busque que las publicaciones hechas tengan una verdadera finalidad y no sólo sea cualquier ocurrencia aleatoria del usuario (a excepción de que ese sea el verdadero motivo por el que se creó la red social).

Con el prototipo de red social expuesto en este documento, se espera poder dar una pauta para que las personas que tengan planeado crear una red social tengan en cuenta las propuestas que a continuación se describirán, para que puedan mejorar los conceptos o tomar aquellos que les convenga y tengan una plataforma con mayor calidad.

Es importante mencionar que el propósito general de este proyecto no es tener solamente una red social común, que sólo sea para interactuar y que se vea "bonita" a los ojos del usuario, ya que para lograr esto se necesita invertir mucho tiempo en temas que se pueden desviar del objetivo real del prototipo. Lo importante a fin de cumplir las metas que se propusieron es que la aplicación funcione correctamente de forma interna, dando también una vista algo simple pero agradable al usuario.

En el primer capítulo se plantea el problema tratando los antecedentes, los objetivos y las preguntas de investigación; en el segundo capítulo se aborda el marco conceptual con los términos y las teorías que sustentan el presente documento; continuando en el tercer capítulo se explica la metodología empleada; la siguiente sección contiene resultados obtenidos, con diferentes pruebas realizadas; para concluir, se habla de conceptos que pudieron ser aplicados y a la vez con la intensión de sugerir posibles rumbos para el lector, quien pudiese encontrar este documento como punto de partida si busca usarlo como pieza de investigación.

I. Planteamiento del problema

1,1 Antecedentes

El surgimiento de las redes sociales no está datado con exactitud, pero puede ser una consecuencia ante el logro de mandarse correos de forma electrónica, es decir una comunicación en correspondencia entre 2 computadoras, una hazaña lograda en los años 70s.

En esos años surgió la necesidad de compartirse información entre usuarios de una misma compañía, es así como surge HTML atribuido a Tim Berners Lee, persona a quien se le denomina ser el creador del Internet. En los años 90s ya existían aplicaciones muy desarrolladas que ayudaban a usuarios del Internet, basados en una práctica surgida en los años 70s con la red llamada SNDMSG, usada en ARPANET (precursora del Internet). Al mandarse cierta información de forma electrónica con personas que pertenecían a una misma red, a partir de SNDMSG, el uso del "@" fue para referirse a un servidor donde los usuarios estarían conectados, dando paso a la nueva cultura del e-mail y en los años 90s el surgimiento de los metadatos al tener el interés de compartir algo más que texto en un correo [1].

En los años 90s se tenían ya entonces dos opciones para compartir información, ya no entre usuarios de una misma compañía, si no entre personas que estuvieran colocadas en distantes puntos geográficos a nivel mundial. Estas opciones eran por medio de una página web o la interconexión entre servidores o dominios. En el caso de las páginas web surge el uso de foros, páginas aún existentes donde una persona empieza un tema de discusión o interés y otras posteriormente escribirán sus opiniones con respecto al tema, formando entonces una página con un historial desplegable de mensajes; es así como se formarían localidades virtuales en la red para hablar sobre diversos temas.

En el caso del uso de dominios o servidores, al usar aplicaciones de software, permiten a un usuario tener uso de una cuenta, donde se recopila todo un buzón personal. El uso de este tipo de mensajería no presencial empezó a tener muchas ambiciones, como: asegurarse que el destinatario realmente recibiera el mensaje, tener una notificación de que

la persona ya atendió el mensaje, tener una respuesta más inmediata, poder contactar de forma segura a la persona deseada con eficiencia, entre otros.

En el caso de los foros, pueden pasar horas indefinidas para recibir una respuesta y a las personas pudiesen conocerlas o no conocerlas; en el caso de los dominios sigue usándose de una forma muy personal, los usuarios prefieren sentirse libres de leer y responder correspondencia cuando a ellos les plazca, la mayoría de las veces son personas que no se conocen pero surge la necesidad de poseer cierta habilidad e intención de memorizar direcciones de correo electrónico para identificar a los contactos.

Ya para finales de los 90s el uso de *Hotmail* y *Yahoo!* era bastante común cuando se trataba de mandar un correo electrónico. El uso de foros era bastante útil para resolver preguntas recurrentes sobre cualquier tema. Sin embargo, estas herramientas TIC no tenían aún el ingrediente suficiente para que los usuarios interactuaran lo suficiente, los mensajes seguían teniendo una muy larga espera para ser respondidos [2].

Ante esto, los dominios evolucionaron del uso de correspondencia por e-mail a chateo. El chateo ofrece que dos personas o más hablen entre sí al momento de establecer una conexión. Una vez hecho esto, las personas pueden interactuar cuantas veces quieran. Desde los primeros dominios de chateo se les incorporó vista de estado de mensajería: se podía ver si el destinatario recibía el mensaje, si lo había visto, e incluso saber cuándo estaba escribiendo. Los dominios más populares fueron *Hotmail* y *Yahoo!*, pero también hubo foros que evolucionaron e implementaron el chat, sin embargo, en estos grupos la identidad de la persona sigue siendo desconocida.

Cabe destacar que ni una de estas prácticas ha dejado de usarse hoy en día, además, desde sus comienzos se quiso implementar el envío de multimedia. En un principio no era tan sencillo mandar un audio o video porque los formatos eran muy pesados a pesar de no tener la mejor calidad, cabe decir que por esto se tuvo la iniciativa de crear formatos más ligeros, con mejor calidad, para poder así enviar multimedia sin problemas; siempre y cuando el formato no sobrepasara unos pocos megabytes.

Se estima que el inicio de las redes sociales se dio en el año de 1995 con una red social llamada "classmates", creada por Randy Conrads, que ayudaba a contactar a los antiguos compañeros, ya sea de alguna escuela o de trabajos pasados sin importar en que parte del mundo se encontraban. A partir de esta creación se fueron dando gran variedad de sitios de este estilo, conocido como círculos de amigos y no red social como actualmente se nombra a este tipo de páginas o sitios web [3].

Empezando la década de los 2000s fue cuando se vio el enorme potencial que se tenía, ya que eran proyectos muy rentables que crecían en popularidad, especialmente en el año 2013 cuando es creada una de las redes sociales más conocidas a nivel mundial y actualmente una página poco visitada: Myspace, que gracias a su facilidad de subir archivos de diferentes tipos ayudó a los grupos de música menos conocidos a difundir sus canciones a nivel mundial para obtener seguidores y llamar la atención de las grandes productoras [4].

Ese mismo año se fundó otra red social conocida como Hi5, que inicialmente se creó para que la gente se contactara entre sí, pero terminó como un sitio centrado en juegos, ganándose a los usuarios *gamer* (anglicismo para referirse a una persona que le gustan los videojuegos). Nelson Solorzano, expresa en su *blog* que la popularidad de este sitio se dio porque "Hi5 ofrece una experiencia de diversión, entretenimiento interactivo a millones de visitantes de todo el mundo mediante la combinación de una plataforma social robusta con contenido de primera calidad, incluyendo los juegos sociales, bienes virtuales, avatares y mucho más." [5].

Actualmente se cuenta con Facebook, sin lugar a dudas la líder a nivel mundial, con miles de usuarios nuevos cada día. Fue creada en el año de 2004 como un servicio para los estudiantes de Harvard, expandiendo sus servicios por varios institutos más hasta llegar a ser abierta para el público en general. Su fácil navegación y el famoso botón "me gusta" fueron factores para que las personas optaran por tomar este sitio como su principal comunicación con sus conocidos [6].

El uso de estas redes sociales es muy sencillo y cualquier persona es capaz de interactuar en estas plataformas, pero en usuarios poco expertos en el tema suele pasar el

hecho de no saber cómo buscar algún tema que les agrade o darle seguimiento a historias que le llamen la atención.

1.2 Definición del problema

En las redes sociales actuales se generan miles de publicaciones al día y es común que se encuentren por orden cronológico, por lo tanto, es muy fácil que se pierda de vista una publicación que atrae y suele ser muy tedioso tener que buscarla de nuevo, resultando una pérdida de tiempo el encontrarla otra vez.

Además, el contenido publicado es de cualquier índole y no se encuentran espacios para tocar temas particulares donde se pueda obtener o compartir información. Otro caso que también se presenta es tener algún problema o algún plan que mostrar y no tener un sitio especial donde se dé la oportunidad de recibir apoyo y consejos de los demás usuarios, o servir como ejemplo para los que tienen alguna situación similar.

1.3 Objetivos

El objetivo general de este proyecto es crear un prototipo de red social que ayude a dar seguimiento a las metas o deseos específicos que comparten los usuarios, mediante un control dinámico de la popularidad de las publicaciones y la eliminación de las menos interesantes.

Algunos de los objetivos específicos son los siguientes:

- Seleccionar herramientas de programación web, de preferencia de uso libre, que sean adecuadas para el desarrollo de sitios web respaldados por bases de datos.

- Diseñar una base de datos eficiente, que cumpla con la normalización para evitar redundancia en los datos y asegurar la integridad de los mismos.

- Implementar conceptos de usabilidad y restricciones en publicaciones para evitar actividades que generen un efecto negativo entre usuarios.

1.4 Preguntas de Investigación

¿Qué ofrecerá por novedad el proyecto con respecto a las demás redes sociales en cuanto a publicaciones?

¿Es correcto eliminar publicaciones con poco seguimiento?

¿Qué tipos de deseos o metas deben de publicarse?

1.5 Justificación

La utilidad que tendrá el proyecto es que se ayudará a disminuir el tiempo que se tarda en encontrar una publicación y se buscará que el seguimiento hacia las historias sea más dinámico y más fácil para los usuarios.

Además, se busca que los usuarios puedan escoger qué tipo de publicación quiere ver, quitando algunas que puedan desagradarle. Por lo tanto, se busca beneficiar al público en general, brindando una experiencia agradable al tener una cuenta en el prototipo de red social propuesto y motivar a los participantes a publicar sus metas en la vida, sabiendo que pueden recibir apoyo o retroalimentación por parte de terceros, si llega a ser popular su historia.

II. Marco Referencial

2.1 Marco conceptual

2.1.1 Redes sociales

Como comentan S. Lara y C. Naval, en participación en la sociedad del conocimiento y redes sociales, "Las redes sociales son aplicaciones web para comunicación e interactividad entre usuarios. Donde los grupos compuestos de personas probablemente entre ellas conocidas, en función de los mismos intereses, gustos, entornos e incluso de la misma generación; desde luego muchas más variables. La red social puede facilitar el hallar a personas con las que se comparte algo en común, entonces poder entablar un tipo de comunicación, muchas veces un reencuentro después de muchos años" [7].

M. Zamora, respecto a redes sociales en internet, comenta que: "Las Redes son formas de interacción social, definida como un intercambio dinámico entre personas, grupos e instituciones en contextos de complejidad. Un sistema abierto y en construcción permanente que involucra a conjuntos que se identifican en las mismas necesidades y problemáticas y que se organizan para potenciar sus recursos. Una sociedad fragmentada en minorías aisladas, discriminadas, que ha desvitalizado sus redes vinculares, con ciudadanos carentes de protagonismo en procesos transformadores, se condena a una democracia restringida. La intervención en red es un intento reflexivo y organizador de esas interacciones e intercambios, donde el sujeto se funda a sí mismo diferenciándose de otros." [8]

2.1.2 Aplicaciones web 2.0

S. Lara y C. Naval explican que "Existen una gran cantidad de herramientas que facilitan la interacción; algunas de las más populares son las redes sociales, *blogs*, *wikis*, *podcast*, páginas de fotografías, etiquetados, entre otras muchas" [7]. Básicamente trata de todas aquellas aplicaciones web o páginas que sirven para que una comunidad de usuarios

interactúe, en este caso no todas las aplicaciones web de la generación 2.0 entran en la categoría de red social, punto que se retomará más adelante.

Las aplicaciones web 2.0 ayudan a establecer un punto terminal (referencia común) para que una comunidad de usuarios que comparten un interés parecido pueda acceder a ello. Sin embargo, no garantiza la interactividad entre usuarios, como es el caso de las *Wikis* (bibliotecas virtuales) o páginas para edición de archivos tales como imágenes, ya que en ellas se puede agrandar o editar la imagen, pero no es un punto de interactividad entre personas (usuarios).

2.1.3 HTML y CSS

Lenguajes de programación necesarios para el desarrollo de páginas web: HTML (*HyperText Markup Language*), éste se utiliza para elaborar entornos web y CSS (*Cascading Style Sheets*), el cual ayuda a programar un estilo predeterminado para un conjunto de páginas, haciendo innecesario adornar cada una de ellas. El primero, el HTML, cuenta ya con 5 versiones, adoptándose como estándar la 5.0 desde finales del 2014, aunque la versión 4 fue estándar durante muchos años y puede seguir usándose. CSS 3 es utilizado desde 1999 y es enteramente compatible con conceptos de CSS 2.1 para su uso [9].

2.1.4 *MVC* (*Model View Controler*)

Modelo Vista Controlador, es una filosofía y patrón para diseño de aplicaciones web y móviles. Las cuales están respaldadas por una base datos [10]. Se describe de la siguiente manera:

- Modelo: Contiene el núcleo de la funcionalidad (dominio) de la aplicación, encapsula el estado de la aplicación y no sabe nada (independiente) del Controlador y la Vista.
- Vista: Es la presentación del Modelo, puede acceder al modelo, pero nunca cambiar su estado y puede ser notificada cuando hay un cambio de estado en el Modelo.
- Controlador: Reacciona a la petición del Cliente, ejecutando la acción adecuada y creando el modelo pertinente.

2.1.5 *Front-end y back-end*.

El *front-end* es todo aquello que se mostrará al invitado de la vista del sitio o página web, siendo de éste más propio de las vistas en *MCV*. De esta forma los cambios que se hagan aquí no afectarán a su contraparte *back-end* (operaciones propias del servidor) y a la vez, los cambios que se hagan en este último no afectarán al *front-end*, pues el *back-end* es propio del *MVC* en operaciones en los controladores, que son programaciones utilizadas en los datos del servidor para fines tales como: consultas, búsqueda, validaciones, operaciones, conteo, etc. Es bastante recomendable que el visitante no tenga nada que ver con el *back-end*, propio del administrador o de los administradores. Ambos (*front-end* y *back-end*) tienen interacción con el modelo en el *MVC* [11].

2.1.6 *Host online*

Son servicios de *Host* en la red WWW, los cuales consisten en ser una conexión de computadoras, estos pueden ser puestos en línea desde un *router* (dispositivo de red para *redireccionamiento*) propio o desde uno *online*. Este caso se refiere a un servicio de paga, los cuales tienen cierto soporte, pueden garantizar un alcance considerable y gran estabilidad del servicio para que el sitio web se mantenga en línea. Algunos de estos servicios son tan avanzados que ya no cuentan con un servidor en físico si no en la *nube* (servidores virtuales-en línea), servidores propuestos para una mayor estabilidad y velocidad de los sitios [12].

2.2 Marco teórico

2.2.1 Teoría de los seis grados de separación

Esta teoría se menciona por primera vez en una historia del húngaro Frigyes Karinthy, llamada *Chains*, en el año de 1929. Explica que en el planeta una persona puede ser conectada con cualquier otra por no más de cinco intermediarios. “Propuse un problema más difícil: encontrar, utilizando uno de mis contactos, la vinculación con un anónimo trabajador en la compañía Ford Motor - y lo logré en tan sólo cuatro pasos. El trabajador conoce a su capataz, quien conoce al Señor Ford, quien es buen amigo del director general del imperio publicitario *Hearst*. Yo tengo un amigo cercano el Señor Árpád Pásztor, quien

recientemente habría entablado amistad con el director de la publicidad *Hearst*. Yo podría pedirle como favor a mi amigo que enviara un telegrama al director de *Hearst* pidiéndole contactar al señor Ford, quien entraría en contacto con el capataz quien le solicitaría al trabajador ensamblar un nuevo automóvil, el cual estoy necesitando." [13] Este es un fragmento del libro donde se da un claro ejemplo de cómo se usa esta teoría para contactar a una persona al azar a través de los contactos conocidos.

Esta teoría también fue plasmada por Duncan J. Watts en su libro "Six Degrees: The Science of a Connected Age" [14], donde se explica que una persona conoce en promedio a unas 100 personas, entre amigos, familiares y compañeros de sus actividades. Si cada uno de esos amigos o conocidos cercanos se relaciona con otras 100 personas, cualquier individuo puede pasar un recado a 10,000 personas más tan sólo pidiendo a un amigo que pase el mensaje a sus amigos. Este alcance aumenta de forma exponencial, hasta llegar a los seis pasos donde, con las tecnologías disponibles, se podría enviar un mensaje a cualquier individuo del planeta.

2.2.2 Teoría del intercambio social

Fue propuesta por Homans (1961), Thibaut y Kelley (1959) y finalmente Blau (1964), quien contribuyó a hacerla más fácil de entender. Homans afirma que toda la conducta humana social (la que se da entre dos personas que interactúan espontáneamente), es un intercambio: una relación entre dos o más personas se da si ambas quieren obtener algo a cambio de la otra y se va a mantener si sus esperanzas se confirman.

Aunque puede ser ilimitado el número de actividades que se pueden intercambiar, lo más importante en el intercambio social es una característica que todas poseen: el "valor", que se puede definir como el grado de esfuerzo que se deriva de una actividad y puede ser bueno o malo. Félix R, en su libro "Análisis de las redes sociales", define al intercambio social como "Los procesos de intercambio que son el resultado de los intentos de los actores (personas involucradas) por realizar y satisfacer sus necesidades" [15], por lo que para exista un intercambio, los participantes deberán hacer un análisis de costo/beneficio.

2.2.3 Teoría de la cola larga (*long tail*)

Fue propuesta por Chris Anderson en 2004, es un enfoque de negocios nuevo en Internet que busca aprovechar que con la tecnología actual se quitaron las limitaciones físicas y geográficas que tienen los modelos antiguos de negocios. El autor señala que "Solo ofertar los éxitos no es lo mejor" [16], gracias a la reducción del costo de almacenamiento y distribución que permiten las nuevas tecnologías, hace que ya no sea necesario enfocar el negocio en vender unos cuantos productos que generen mucha venta y se abrió un nuevo mercado poco conocido, que puede llegar a ser igual de redituable que el mercado anterior.

Esta teoría va en contra de la ley de Pareto (20/80) [17] que especifica que debes de centrar la venta en el 20% de los productos para obtener el 80% de los ingresos, válida en negocios físicos. Se puede buscar tener la acumulación de las pocas ventas que pueden tener muchos productos, dando grandes ganancias al aplicar este conocimiento. Existen negocios en Internet que actualmente utilizan este modelo, por ejemplo 'Amazon' (sitio web), que genera entre el 25% y el 57% de sus ingresos por vender libros al ofertar títulos que habitualmente no se encuentran en una librería común.

2.2.4 Popularidad y su medición

La palabra popularidad, que tiene su origen en la palabra popular, se define como un adjetivo que señala que pertenece o que es relativo al pueblo, según el diccionario. La popularidad es la notoriedad social que se imprime cuando se cree que una persona o un producto tienen cierta importancia entre un considerable número de personas. Por lo anterior, Kotler [18] enfoca la popularidad hacia el marketing en su libro "Fundamentos del *Marketing*", donde explica desde una perspectiva de cómo vender un producto, qué estrategias son necesarias y a qué público va dirigido. Un ejemplo de ello es que en una compañía se requiere un control de marketing. Éste consiste en un proceso de medir y evaluar los resultados de estrategias, planes de marketing y tomar medidas correctivas para mejorar el proceso.

También explica que en el proceso del marketing las necesidades del consumidor son una pieza clave para medir el nivel de popularidad de un producto en una compañía. Así que consiste en segmentación de mercados, es decir, en hacer pequeños grupos que definirán a sus clientes por su ubicación geográfica, su demografía, sus costumbres y tradiciones, entre otros. Entonces, el vendedor se pregunta si el producto o servicio cubre las necesidades del grupo ya segmentado de clientes a quienes se les pretenda vender y si el producto llena esas expectativas para que puedan ser consumidores leales a la marca. Entonces, la lealtad de la marca que los consumidores tienen del producto es una forma de medición del nivel de popularidad del producto.

Además, Kotthler, si bien dice que los ingresos de la compañía se pueden medir sabiendo si un producto es popular y atractivo para el consumidor, también se puede al medir otros aspectos, por ejemplo, desde cómo saberlo vender y qué estrategias se pondrán en práctica para obtener los objetivos de la compañía.

También Kotler habla en su libro acerca de cómo la clave del buen *marketing* es crear relaciones con el cliente, que le produzcan satisfacción. Para muchas compañías, la administración de las relaciones con el cliente depende no sólo de los empleados que interactúan con los clientes, sino también del software.

2.2.5 *Rating*

Rating se refiere a la cantidad de personas que está poniendo su atención en algún medio, normalmente se usa para un programa de televisión o de radio. A mayor *rating*, mayor cantidad de gente consumiendo el medio de comunicación en cuestión.

En el libro “Medición de audiencias de televisión en México” [19] escrito por Alejandro Garnica y Rubén Jara, presenta a IBOPE AGB México, que fue el primer sistema de medición de *ratings* de televisión en México, en 1991. Este modelo consiste en un panel para hacer muestreo en los cálculos del *rating* según los segmentos de audiencias que busca medir. Se comienza por la recolección de datos, que se organiza de la siguiente manera: la tele hogar seleccionada recibe una invitación a participar en el panel. Después de ser aceptada, se diagnostica el tipo de televisión que se tiene. En cada tele hogar se instala

peoplemeter (microcomputadora conectada al televisor) hasta hacer una red entre ellos. Así se forma la red que funciona como un sistema maestro y los otros como equipos esclavos. Después, cada noche, el centro de acopio de datos se comunica con el *peoplemeter* para extraer la información que se ha guardado durante el día.

Así que el sistema transforma los datos, valída y los convierte en parámetros internacionales para la edición de datos. Con ello se pueden obtener los niveles de *rating*. Es decir, se elige una prueba para hacer un muestreo y así se define por este modelo los niveles de audiencia que tiene un programa de televisión transformados en porcentajes, los cuales ayudan a los directivos de la empresa a evaluar si sus programas han tenido el resultado previamente presupuestado o si se requieren cambios para que sean del agrado del televidente, pues son ellos quienes mantienen los niveles de *rating* y quienes mantienen al aire los programas de su preferencia.

2.2.6 Usabilidad

La usabilidad es la capacidad por la cual un producto puede ser usado por usuarios específicos para conseguir los objetivos que se plantearon con efectividad, eficiencia y satisfacción. Steve Krug [20] en su libro "No me hagas pensar", ofrece muchos consejos para darle a las aplicaciones web la capacidad de poder ser utilizadas por cualquier tipo de usuario con comodidad. Como lo menciona el título del libro, entre menos tiempo gaste un usuario en aprender a utilizar la aplicación, mejor satisfacción tendrá al finalizar su uso y mayor será la cantidad de veces que utilizara el servicio que se le está brindando.

Krug menciona que "el uso más corriente de la web viene motivado por el deseo de ahorrar tiempo y, por ello, los usuarios de la web tienden a actuar como verdaderos tiburones: o se mueven constantemente o mueren". Por eso es importante conocer la forma en que un usuario promedio interactúa al navegar por Internet. Según estudios, la gente no suele leer por completo la información que se ingresa en el sitio, le da una hojeada al contenido y selecciona las opciones que le llaman la atención, dejando de lado mucha información que puede serle útil, pero que no tomó en cuenta. Para llamar la atención del usuario son necesarios estos cinco puntos:

- Creación de una jerarquía visual clara en cada página.
- Aprovechamiento y uso de las convenciones.
- División de las páginas en zonas claramente definidas.
- Dejar bien claro sobre lo que se puede hacer clic.
- Minimizar el ruido.

2.2.7 Red social

Barnes [21] en 1954 dijo que la red es "Un conjunto de puntos que se conectan a través de líneas". Los puntos de una imagen son personas, y a veces grupos, y las líneas indican las interacciones entre esas personas y/o los grupos" Es decir, una red social es como las relaciones humanas que tienen un impacto duradero en la vida de un individuo.

Existen clasificaciones de características que conforman una red social según el libro: "Las redes sociales se clasifican por sociales: Un concepto con importantes implicaciones en la intervención comunitaria", por Cristina Villalba Quesada, donde declara que se dividen en estructurales e interaccionales. Primero por su tamaño, distribución, densidad y dispersión, [22]. Después por multiplicidad, contenido transaccional, direccionalidad, duración, intensidad, frecuencia y características de apoyo social de las redes sociales, entre otras.

En la actualidad, las redes sociales continúan siguiendo el mismo patrón. Una red social, como se mencionaba anteriormente por Cristina Villalba en su libro Redes Sociales, [22]; es utilizada como un medio de comunicación que mantiene comunicadas a las personas con vínculos, empieza desde la familia y amigos hasta cerrar negocios por medio de ellas. Entonces, las redes sociales son una nueva oferta de mediación, relación y contenidos post-mediáticos que comienzan a constituir una nueva forma de hacer publicidad y atraer clientes o expresar una opinión.

Las redes sociales actualmente son un mundo de posibilidades que las personas tienen, donde se mantiene información actualizada y que permite a una persona o grupos estar informados para tomar decisiones. Es importante la relación interpersonal a lo largo

de la vida y esto acentúa la idea de que el individuo crece y madura rodeado de personas cercanas e importantes para él o ella.

Esto significa que se pueden definir las redes sociales como conexiones entre personas, las cuales se conectan por un sentido de pertenencia que tiene el ser humano para interactuar y crear relaciones.

2.3 Marco tecnológico

2.3.1 Uso de servidor local WampServer o Servidor local Xampp

Software recomendado para trabajar con desarrollo de aplicaciones web a nivel servidor local por medio de Apache hasta la versión 2.4.18 y lenguaje de programación PHP en la versión 5.6.19. Posee varias extensiones para el desarrollo del trabajo, entre ellas se mencionan algunas utilizadas en el presente proyecto, por ejemplo: *phpmyadmin, msql, mysqli,* y XML. Estos programas son recomendables para diseñar y manipular bases de datos, ya que se pueden realizar consultas en lenguaje SQL y MySQL, así como establecer una gran cantidad de idiomas en los que pueden estar estructurados los datos. Cada uno de estos programas cuenta con su sitio oficial y documentación oficial para su uso y además son gratuitos [23] [24].

2.3.2 HTML 5.0, CSS.

Es recomendable usar HTML 5.0 y utilizar *plugins* que ayuden al concepto estético y contemporáneo en las páginas web. También, el uso de diferentes *frameworks* ayudará en el uso de esta versión de HMTL, por ejemplo: Bootstrap 3.3.7, *Font Awesome* 4.6.3 y CodeIgniter. Estos *frameworks* son enteramente compatibles con HTML 5.0 y ayudan a estilizar los entornos de hipertexto, siendo algunos de ellos extensiones del CSS, como es el caso de Bootstrap *y* Font Awesome [25].

CSS es una herramienta recomendable para establecer un concepto de diseño a todas las vistas que surjan en un conjunto de páginas web, y de esta manera construir un sitio más limpio y con una vista uniforme [9].

2.3.3 Bootstrap versión 3.3.7 y Font Awesome versión 4.6.3

El uso de los *frameworks*: Bootstrap 3.3.7 [26] y Font Awesome 4.6.3 [27] es ideal para facilitar el uso de CSS, pues cuentan con estilos ya prestablecidos para iconos, entradas para formularios, barras navegadoras y botones. A la vez, posee numerosas extensiones para usar comandos de JavaScript y con esto construir sitios web dinámicos, algo bastante utilizado en los entornos de usuarios actuales. Para el uso de estos *frameworks* es recomendable utilizar conceptos de programación orientada a objetos, para tener una cantidad necesaria y posiblemente justa en líneas de código. Cada uno de estos entornos tiene su propia página oficial donde están las documentaciones oficiales para facilitar su uso.

Cabe mencionar que se pueden utilizar al mismo tiempo estas herramientas junto con otros *frameworks* adicionales, que permitan complementar el diseño del sitio. Se cuenta con un sitio y documentación oficial para cada uno de ellos, los cuales son mencionados en la bibliografía, así como su respectivo URL.

2.3.4 CodeIgniter

Cuando se cuenta únicamente con conceptos básicos acerca de la programación de páginas *web* y sobre todo cuando éstas implican el uso de bases de datos, es normal tener una gran cantidad de código, lo cual dificulta el mantenimiento del entorno. Para ello, es necesario tener una herramienta que facilite el manejo del código y le dé estructura. Esto es deseable y se facilita por medio de una herramienta como CodeIgniter, ya que se puede dar mayor prioridad a mejorar el aspecto y la dinámica del sitio web que a los detalles del diseño del código, pues resulta simple en Codeigniter predefinir una estructura ya automatizada.

CodeIgniter es un *framework* que funciona con el modelo *MVC*, el cual, como se mencionó anteriormente, permite dividir la estructura de un sitio en dos términos *front-end* (lo que el usuario ve) y *back-end* (controles y parámetros con el servidor, que el usuario normalmente no ve). Estos conceptos están definidos en el marco conceptual [28].

2.3.5 Entornos de programación PHPstorm y Notepad++

Existen diversos editores de texto en entornos de programación, sin embargo, PHPstorm y Notepad++, fueron seleccionados dados sus conceptos de usabilidad. Brindan apoyo al programador tal como dar sugerencias de palabras; copiar, pegar y buscar o remplazar texto; marcar palabras o sintaxis posiblemente errónea, entre otras; además de contar con navegación entre carpetas, lo cual ahorra mucho tiempo [29] [30].

2.3.6 Servicio de *hosting.*

Existen numerosos servicios de *hosting* de sitios web disponibles. Usualmente se recomienda utilizar alguno que sea de paga, ya que, según la tarifa, ofrecen mejor alcance y prestaciones para mantener en línea un sitio propio. Gracias a este servicio, el sitio web no tiene que depender de un servidor físico, al menos no propio; al pagar por tal servicio se tiene la posibilidad de hacer valer garantías con tal de no tener 'caídas' (tiempos muertos) en el servicio del sitio web. Se pueden mencionar, entre otros, a Hostgator, PCMag, Bluehost y Dreamhost, sin embargo, para este proyecto, se decidirá la mejor opción de acuerdo a las condiciones y precios, sobre todo considerando que la red propuesta es un prototipo solamente. Desde luego que existen *host* gratuitos, pero normalmente no se contaría con ciertas garantías de uso, usualmente tienen poco alcance y su soporte será escaso [31].

2.3.7 *Jquery*.

Es una biblioteca de JavaScript que ayuda a simplificar las acciones que deben de realizarse mientras el usuario interactúa con el documento HTML que se proporciona. En el caso de este proyecto, se usa junto con la técnica de desarrollo web *AJAX* y el formato de intercambio de datos JSON para cargar dinámicamente los datos que el usuario pueda requerir mientras navega en la red social, ya sea necesario consultar la base de datos o apoyar en el manejo de la información del "deseo" que se está actualizando o creando.

III. Metodología

3.1 Bosquejo de la idea de la aplicación.

El primer paso que se realizó fue un bosquejo rápido de las vistas que serían necesarias y a las que tendría acceso el usuario al ser parte de la red social. En esta etapa se concluyó que serían necesarias cuatro páginas principales:

- El registro, donde el nuevo usuario puede darse de alta para ser parte de la red social.
- El módulo de inicio y cierre de sesión, para que el usuario previamente registrado pueda autenticarse y usar la red social.
- La pantalla principal del usuario, donde pueda ver los avances de sus deseos, modificar, borrar o ingresar uno nuevo.
- La pantalla People, donde se pueden ver los deseos que los otros usuarios subieron, poder observar las fotografías que subieron y dar su aprobación o desaprobación al deseo.

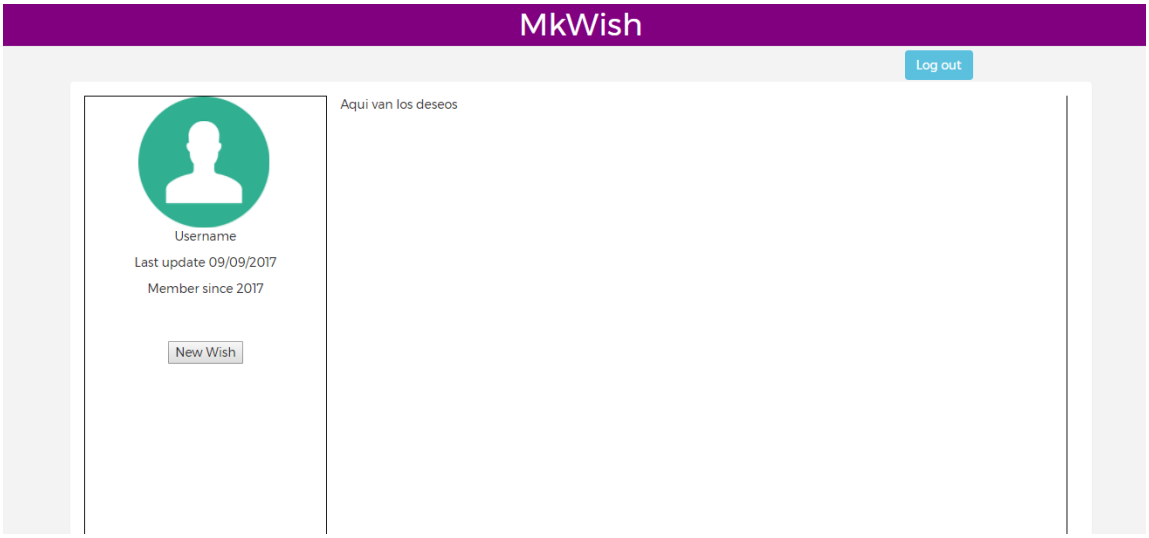


Ilustración 1 Primer bosquejo vista Home

3.2 Diseño de la base de datos.

El sistema de gestión de bases de datos que se utilizó es MySQL, en este mismo fue donde se diseñaron las tablas necesarias para poder llevar un control de los datos que se manejarán en la propuesta de red social. En todo momento se buscó cumplir con las reglas de normalización para tener una base de datos coherente y eficiente.

Las tablas necesarias, así como su composición son las siguientes:

Tabla *users*: Se guardan los datos básicos de un usuario, los cuales fueron enviados directamente del formulario al momento de crear su cuenta. Además, tiene los campos necesarios para soportar la activación de su cuenta, recuperar la contraseña en caso de que se olvide y tiene ciertos campos que ayudan a ver la actividad que tiene regularmente al ingresar en la aplicación.

Tabla *users_groups*: Esta tabla se genera con la relación entre la tabla *users* y la tabla *groups*, ya que un usuario puede tener varios roles y no por eso es necesario tener varias veces ingresado al mismo con diferentes roles.

Tabla *groups*: Se guardan los distintos tipos de usuario con los que se cuenta.

Tabla *login_attempts*: Esta tabla se utiliza para saber si un usuario está intentando entrar en la aplicación, si alguien externo es quien quiere acceder, se guarda la dirección *ip* para posteriormente investigar que ocurrió.

Tabla *user_save_wish*: Esta tabla se genera con la relación de la tabla *users* y la tabla *wish* para saber los deseos que tiene guardado un usuario.

Tabla *wish*: Se guardan los datos básicos que se ingresaron de un deseo, además de tener los elementos necesarios para su correcta clasificación, el estatus actual y el tipo de deseo. Es importante señalar que los *likes* y *dislikes* se guardan en esta tabla a través de un JSON, dejando toda información de un deseo en un solo renglón.

Tabla *photo*: Esta tabla ayuda a administrar las fotografías de los diferentes deseos que se tienen.

Tabla *type*: Esta tabla ayuda a administrar los diferentes tipos de deseo que se pueden ingresar.

El modelo entidad relación final se muestra en la ilustración 2

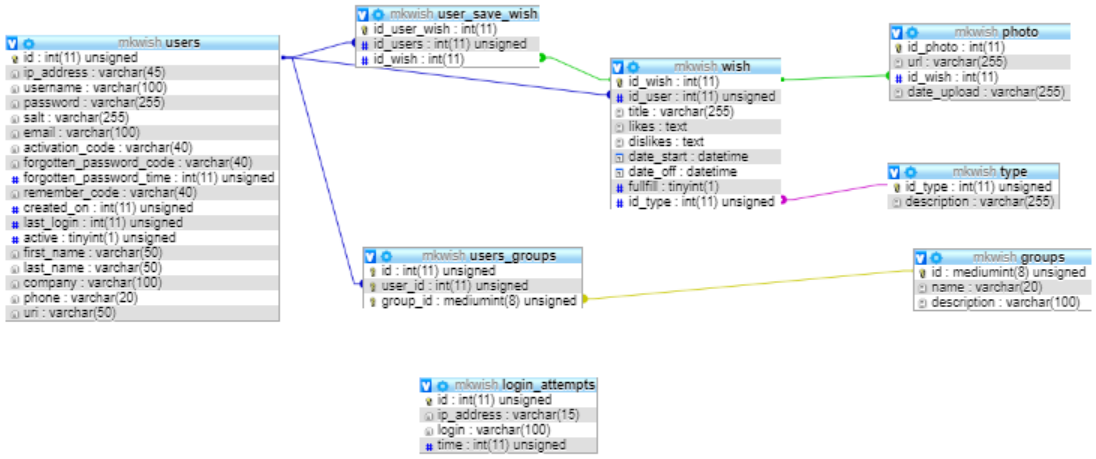


Ilustración 2 Modelo entidad relación de la base de datos

3.3 Implementar el registro a la red social.

Para tener acceso a la aplicación y tener la oportunidad de interactuar con los demás usuarios que ya se encuentran registrados, es necesario contar con un usuario y contraseña con el que se pueda identificar. Se implementó un pequeño formulario para que el usuario ingrese los datos esenciales que se necesitan para generarle un *uri*, la cual es una cadena única de caracteres generados automáticamente por el sistema tomando como base el nombre de la persona, ésta será de ayuda para distinguirlo en todo momento.

En el formulario creado para dar de alta a un usuario (ver Ilustración 3), sólo es necesario ingresar el nombre, apellido, correo y contraseña, evitando pedirles a las personas información sensible que pudiera ser utilizada de manera inadecuada. Como medidas de seguridad, las contraseñas son guardadas encriptadas, evitando que si algún intruso ingresara a la base de datos, pueda fácilmente observar esa información, además de usar *re-captcha* para confirmar que es un humano el que está creando una cuenta.

Una vez que el usuario ya fue registrado, se crea una carpeta personal (cuyo nombre es el *uri* asignado), inicialmente solo con una foto de perfil provisional, que puede ser cambiada en cualquier momento. Esta carpeta será la base para guardar todas las imágenes que el nuevo usuario utilice para darle mayor visibilidad a los deseos que llegue a subir.

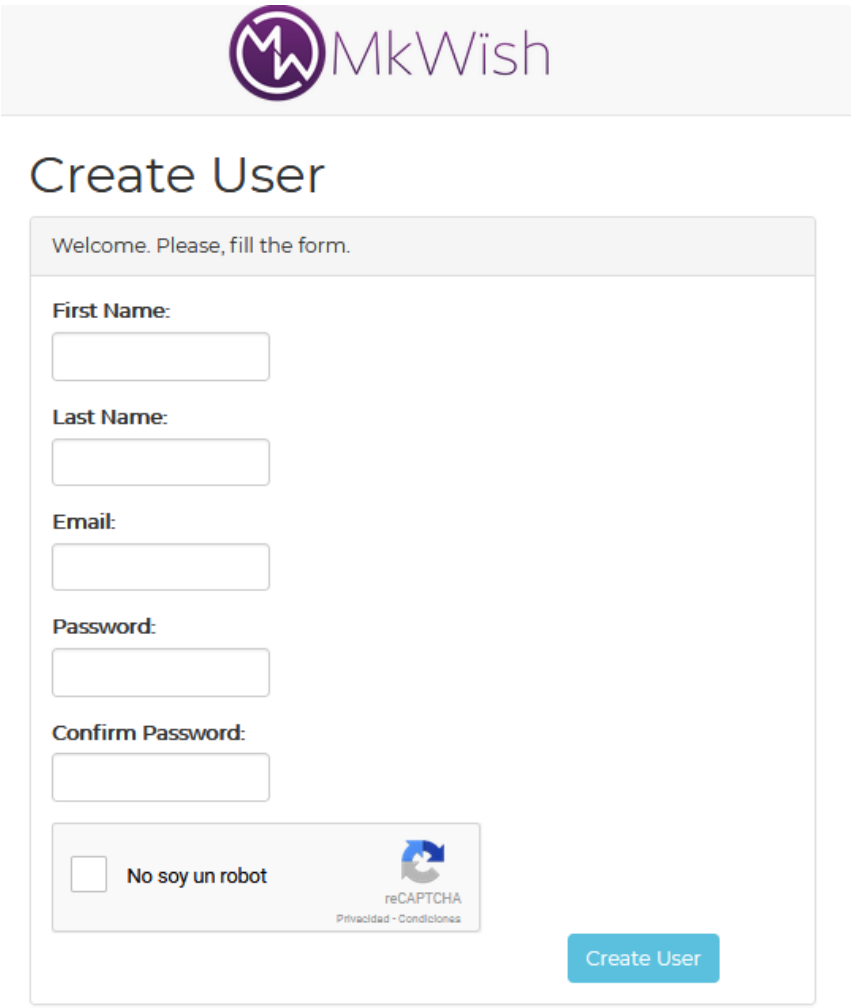


Ilustración 3 Formulario para crear cuenta.

3.4 Módulos para iniciar y cerrar sesión.

Cada usuario contará con un perfil donde tendrá acceso a la información referente a él mismo, tendrá una bitácora de publicaciones hechas por el propio usuario, además de otras actividades referentes a su perfil. Para lograr la implementación se utilizó CodeIgniter,

Al conocer el identificador personal de un usuario, o *uri*, se puede ingresar al perfil de esa persona simplemente agregándolo a la URL principal de la página. Pero al intentar ingresar al perfil de alguien más, a la vista People, o al propio inicio, sin haber iniciado sesión previamente, el sistema posicionará a la persona directamente en la página de inicio de sesión para que se pueda autenticar y navegar sin preocupaciones. Si se presenta el caso

de olvido de contraseña, el sistema cuenta con el sistema de recuperación de contraseña, para no perder la cuenta.

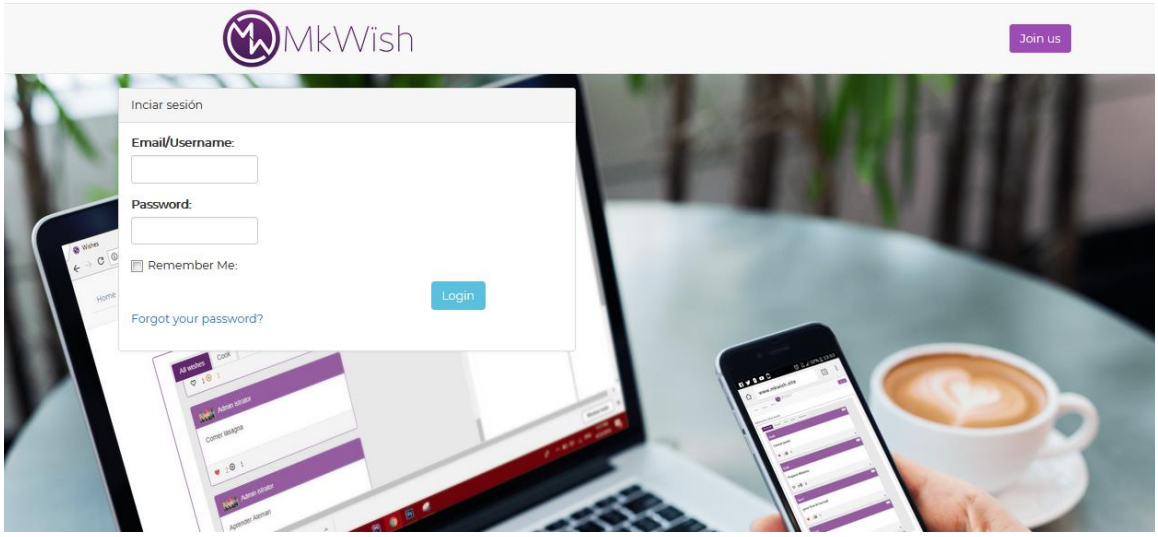


Ilustración 4 Vista inicial de la aplicación.

3.5 Creación de las vistas, modelos y controladores principales para el usuario.

3.5.1 Estructura básica del código

Como se mencionó en la sección 2.1.4, se utilizó el modelo *MVC* para elaborar la aplicación, por lo que fue necesario hacer las conexiones entre los archivos PHP que servirían como vistas, modelos y controladores. El *framework* Codeigniter es una herramienta diseñada para no empezar un proyecto PHP desde cero, utiliza la lógica del modelo *MVC* y los conceptos de programación orientada a objetos, tales como: clases, funciones, constructores, etc. Se implementó el uso de la técnica de desarrollo web *Ajax* con el fin de lograr un portal con actividad más dinámica e interactiva. Para desarrollar en Codeigniter se recomienda usar un servidor local, en este proyecto se utilizó *Wamp* y *Xam*p, ya que además de ser simple de usar, se tiene la ayuda de la documentación oficial publicada en las páginas web de cada uno, donde se explica dónde almacenar los archivos PHP para que sean interpretados por el servidor *HTTP Apach*e, que ayuda a interpretar los archivos que se ejecutan de manera local, sin necesidad de conexión a otros equipos.

La lógica del *framework* Codeigniter está en la configuración de las rutas, las cuales se deben de configurar y dar nombre para que señalen a un controlador por defecto. En el

proyecto Codeigniter se encuentra la ruta en el directorio *application/config/config.php*, dentro del cual se encuentra un arreglo en lenguaje PHP con toda la información que se necesita. Básicamente el orden semántico es: *$config[‘nombre de la ruta’] = ‘nombre del controlador’/’función dentro del controlador’*, al poner en el navegador, se reflejaría de la siguiente forma: *‘url del sitio’/’nombre de la ruta’ o ‘url del sitio’/’nombre del controlador’/’función* dentro del controlador’ ambas formas son correctas al momento de escribirse en el navegador. Al lado izquierdo del “=” se encuentra el nombre que se añadirá a la ruta, al lado derecho de tal signo se encuentra el controlador con el método o función dentro de la clase del controlador. Por lo tanto, no siempre es necesario establecer un nombre, mientras exista el controlador y el método dentro de este último, se está cumpliendo con la lógica del enrutamiento; el nombre generado para la ruta ayuda a no tener que escribirla en su totalidad. Se recomienda que las rutas establecidas bajo un nombre sean para señalar a una vista recurrente.

Para generar un controlador se crea un archivo PHP, el cual debe iniciar con mayúscula y debe encontrarse en el directorio *application/controllers*; el controlador debe tener una función inicial llamada Índex, su uso consiste en que cada vez que se llame al portal en el navegador; el controlador llamará a esta función, la señala como una vista, por lo que se le puede combinar código PHP y HTML. El navegador muestra entonces tal archivo con la lógica del lenguaje de hipertexto.

La vista que se suele configurar bajo esta lógica suele ser donde los usuarios inician sesión y en caso de que esté iniciada una sesión se dirija a otro archivo PHP y sea allí donde se muestre la sesión del usuario. Si la vista requiere de datos por parte de una consulta SQL en la base datos el controlador debe señalar a una clase modelo la cual también tiene funciones o métodos, en cada función se encuentra una consulta en la lógica *Active Record*, el cual consiste en un tipo de traducción del lenguaje SQL para ser interpretada por Codeigniter. En cada función de esta clase, se configura un arreglo con los resultados de la consulta; la ruta de directorios para un archivo modelo es *application/models*. La mayoría de las consultas son del tipo *“Select * ‘table’ from ‘Column’ ”*, por lo tanto daría como resultado todos los campos de la tabla. Se puede configurar una consulta con un resultado único de cierta columna en la tabla, sin embargo el concepto *MVC* para vistas es: obtener

una consulta generalizada de una tabla en la base datos solicitada por la clase del controlador, el controlador manda los datos a la vista establecida y en ella se escoge por medio de funciones (*if, for, foreach, switch ,etc*) qué ver y qué no ver, con el lenguaje PHP o de ser necesario con el lenguaje JavaScript. Si se quisieran mandar valores a la base datos, entonces sería la lógica en reversa: la vista por medio de un formulario que apunte al controlador o la llamada a una ruta del tipo*: ´url´/´Controlador´/´método* en el *Controlador´/´valor,* en *string´*; entonces el método del controlador recibiría los datos y al mandar llamar al método del modelo, la clase del modelo recibiría los datos con los parámetros necesarios en el constructor del método, en la forma: '*nombre del método'('Parámetro 1'…'Parámetro n')*. Se recomienda mandar un arreglo para usar la menor cantidad de parámetros posibles.

Para hacer una nueva vista, que no sería por defecto, se crea un nuevo método con diferente nombre al *Index*, donde apunte a otro archivo PHP. Los archivos PHP que serán parte de las vistas deben estar almacenados en la ruta de directorios *application/views*. En el proyecto que se describe en este documento se hicieron tres vistas principales: Home, Profile y People.

Para la realización del proyecto a describir en el presente documento se utilizaron dos controladores con los siguientes nombres: Admin y Auth. En el primero se establecen todas las funciones o métodos referentes al manejo lógico del portal, como el crear publicaciones o deseos, procesar *likes* o *dislikes*, consulta de publicaciones o de usuarios en la base datos, etc. En el caso de Auth, es por parte del *framework*, se obtiene un controlador, en éste se encuentran métodos para abrir y cerrar sesión; crear usuarios, activar usuarios, restablecer contraseñas, o solicitar cambio de contraseñas; todo lo necesario para el manejo de sesiones. Se logró hacer una combinación en el uso de ambos controladores, las vistas Home, People y Profile están manejadas por el controlados Admin. Una cosa a tener en consideración es que un controlador no puede mandar parámetros a otro por medio de un constructor o instancias propias de programación orientada a objetos.

Ion_Auth, es un *framework* que figura más como una librería, aunque es el conjunto de una librería, un controlador y vistas ya predeterminadas. Está diseñado para usarse con

Codeigniter. En éste se encuentra una vista ya hecha para el inicio de sesión, recuperación de contraseña, cambiar contraseña, etc. Su uso es muy variado, pero consiste en usar las funciones en el controlador que vienen en el archivo comprimido, obtenido al descargar de la página. También cuenta con unas tablas SQL para agregar a la base datos, una de ellas es la de *users*, ahí se encuentra ya una configuración establecida por este f*ramework*. En este proyecto se combinó el controlador Ion_Auth con el controlador Admin, aunque el que está por defecto es este último. Ion_auth se usó para el manejo de sesiones mientras que Admin para las consultas en la base datos del proyecto en concreto. Como se mencionó anteriormente, en el conjuro de carpetas y archivos que se agregan a un proyecto Codeigniter, cada carpeta tiene el nombre de las carpetas del proyecto en *back-end* indicando en qué carpeta debe ir cada archivo PHP de Ion_auth. Sin embargo, una de las carpetas tiene archivos SQL para agregar la base datos. El archivo SQL consiste en tres tablas para el manejo de usuarios, en la tabla Users se encuentran las columnas a las que el *framewor*k hace referencia de forma automática. Ion_auth encripta las contraseñas de los usuarios dentro del portal.

3.5.2 Active Record en clase modelo

Active Record es una estructura sintáctica para hacer una consulta SQL, funciona como una traducción simplificada del lenguaje SQL. Al hacer una consulta la mayoría tienden a ser del tipo "*Select * from table*" donde se esperarían varios renglones con todas las columnas por parte de la tabla como resultado; también se pueden hacer consultas de un solo renglón como: *"Select * from table, table1 where id_primary='1' and table.id_foranium=table.id_foranium*". En la semántica Active Record, la primera consulta luciría de la siguiente forma: *"$query = $this->db->get(table)->result();*", la terminación *result()* almacenará en la variable *$query* un array con todo los renglones de la consulta, mientras que en la segunda consulta luciría de la siguiente forma *"$query = $this->db->where('table.id_foranium=table1.id_foranium')->where('id', "1")->get('table,table1')->row();"*. La terminación *row()* guardará en la variable *$query* un arreglo conteniendo un solo renglón. Los ejemplos expuestos dan a notar cómo la estructura de SQL va de izquierda a derecha mientras que en Active Record es en sentido

inverso. Si se provoca un error ya sea intencional o no, Codeigniter puede mostrar el error por parte de la consulta mal estructurada; mostraría la consulta en la sintaxis SQL a la que se trata de aludir. Las inserciones, actualizaciones y el borrado de alguna tabla también tienen que estar en la sintaxis *Active Record*.

Para combinar Ajax con una vista es necesario crear un archivo JS en el directorio *assets/js* o si se prefiere, *asstets/pluggins/js*, queda a elección de quien desarrolla en Codeigniter escoger la lógica del directorio *assets*; normalmente allí es donde se almacena todo lo que tiene que ver con archivos CSS, JS y/o imágenes del proyecto. Al crear el archivo JS, las funciones adentro deben apuntar a una ruta con un método en el controlador, en este caso como se busca combinar con una vista, dicho método en el controlador debe corresponder al de una vista. Se recomienda agregar el archivo JS como *striming* en el *<footer>* según la lógica HTML de la vista para establecer en ella una variable que sería la ruta para apuntar en la misma y así poder usar esa variable en el formato *Ajax*. En el archivo *config.php* en el directorio *assets/config/* viene para establecer el *base_url*, por lo tanto, cada vez que se busque llamar a la ruta del proyecto, que ésta sea dinámica, así, si el proyecto se llama diferente en algún momento, al llamar al *base_url*, éste la adapte de forma dinámica en el proyecto.

3.5.3 Estructura *MVC* de la vista Home

El método Index del controlador Admin tiene una función *if* para revisar si hay una sesión abierta (ver ilustración 5), se hace una referencia a la librería Ion_Auth. Tal librería tiene funciones estructuradas, en este caso para las sesiones; *"if (!$this->ion_auth->logged_in())"*. Si no existe una sesión abierta, redirigirá el navegador a la vista para iniciar sesión en el portal *("redirect('auth/login', 'refresh');")* en cambio, si existe, la estructura dentro de la función *else*, es para extraer los datos del usuario que tiene abierta la sesión y guardarlo en la variable *$user*. A continuación se hace un arreglo *$data* para enviarlo a la vista con la función *render*, de la siguiente forma: *"$this->_render_page('auth/index', $data);"*; en el arreglo se envía también el nombre del archivo *Ajax* a usar en *index.js* para agregarlo en *<footer>* HTML.

```
public function index()
{
    if (!$this->ion_auth->logged_in()) {
        redirect( uri: 'auth/login', method: 'refresh');
    } else {
        $this->data['message'] = (validation_errors()) ? validation_errors() : $this->session->flashdata('message');
        $this->data['users'] = $this->ion_auth->users()->result();
        foreach ($this->data['users'] as $k => $user) {
            $this->data['users'][$k]->groups = $this->ion_auth->get_users_groups($user->id)->result();
        }
        $user = $this->ion_auth->user()->result();
        $data['js_to_load'] = "index.js";
        $data = array(
            $this->data,
            'user' => $user
        );
        $this->_render_page( view: 'auth/index', $data);
    }
}
```

Ilustración 5 Código para revisar sesión iniciada.

El siguiente método con el nombre Ajax es para hacer una vista diferente al Index, aunque estaría dentro de éste y es a donde el archivo *index.js* apunta. No es realmente necesario tener dos vistas diferentes; se eligió esta estructura para manejar por separado el código HTML de las funciones y las de la vista Index. Queda a criterio de quien desarrolla simplificarlo o no. El archivo *functions.php* es al que apunta el método *Ajax* para crear la vista, en este método se define una variable tipo *post, $selection* en *functions.php*. Hay una función PHP, *switch,* donde se manejará la variable *$selection* (ver ilustración 6).

Ajax bajo el evento del cursor cambia el valor de *$selection* y así cambia al mismo tiempo la vista de manera dinámica. Es así como se muestran los deseos por categorías de los deseos o metas. El *array $post* sirve para enviar todas las variables necesarias para la vista, entre ellas *$user* que es una consulta en el modelo Users, obteniendo de esta forma todos los deseos del usuario en sesión abierta. Es necesario enviar en el constructor el *id* del usuario consultado por separado en el modelo, de la siguiente forma: *"$id = $this->session->userdata['user_id']; $wishs = $this->Users->wish($id);".* Una vez que se tiene la información de cada deseo estos se muestran, además de incluir una opción para eliminarlo y otra para abrir el modal para administrar las imágenes del deseo (ver ilustración 7.

En el archivo *index.php* dentro del directorio *application/view/auth/* es a donde apunta el método Index del controlador Admin, ahí se recibe también el *array $wish* donde se almacenó la consulta de deseos del usuario en sesión por medio del *id*.

```php
public function ajax()
{
    $selection = $this->input->get('selection');
    $name = $this->input->post('name');
    $des = $this->input->post('des');
    $type = $this->input->post('type');
    $id = $this->session->userdata['user_id'];
    $wishs = $this->Users->wish($id);
    $user = $this->ion_auth->user()->result();
    $post = array(
        'selection' => $selection,
        'name' => $name,
        'des' => $des,
        'type' => $type,
        'user' => $user,
        'wishs' => $wishs
    );
    $this->load->view('functions.php', $post);
}
```

```php
<?php
$arr_length = count($wishs);
$type = array();
for ($i = 0; $i < $arr_length; $i++) {
    $type[] = $wishs[$i]->description;
}
$user_session = $user['0']->first_name . " " . $user['0']->last_name;
$email_session = $user['0']->email;
$user_llogin = $user['0']->last_login;
$user_llogin = unix_to_human($user_llogin);
$user_creaton = $user['0']->created_on;
$user_creaton = unix_to_human($user_creaton);
$user_image = site_url( uri: IMAGES_PATH . $user['0']->uri . '/profile.png');
switch ($selection) {
    case "filltabs":...
    case "fillnewwish":...
    case "filldivuser":...
    case "fillwishprofile":...
    case "filldivorderwish":...
    case "filldivsavewish":...
    case "filldivkitchenwish":...
    case "filldivsportswish":...
    case "filldiveducationalwish"...
    case "fillwishupdates":...
    case "savenewwish":
        break;
    default:
        echo("default");
        break;
}
?>
```

Ilustración 6 switch para seleccionar funciones

```php
<?php include "user_session.php"; ?>
<html>
<head>
    <?php include 'header.php' ?>
    <title>Welcome <?= $user_session ?></title>
</head>
<body>
<div class="navbar session navbar-default"...>
<div class="container-fluid"...>
</body>
<footer>
    <?php include 'modalpicprofile.php'; ?>
    <?
    $this->carabiner->js_string(<<<'EOT'
  $('[data-toggle="tooltip"]').tooltip();
EOT
        , 'jquery');?>
    <?php include 'footer.php'; ?>
    <script>
        var BASE_URL = '<?=base_url();?>'
    </script>
    <?$this -> carabiner -> display('jsl', 'js'); ?>
</footer>
```

Ilustración 7 Inclusión del modal para ver imágenes.

3.5.4 Estructura *MVC* de la vista People

Al igual que en la vista Home, People tiene un controlador que la precede, dentro del método People en el controlador, también está la misma estructura en código para revisar si existe alguna sesión abierta o no; si no la hay, redirige el navegador hacia la vista Login, pero si hay sesión abierta entonces prepara la vista People.

Se ideó una vista aparte para las funciones al igual que en Home con el método *ajax_people*, este método envía al archivo *functions_people.php* un arreglo *$data*. En este se envía de nuevo una variable $user con los datos del usuario en sesión y a la vez una consulta en el modelo Users sobre los deseos y se almacenan en la variable *$wishes*. La consulta en el modelo debe ser de los deseos incluyendo los datos de los usuarios. Se busca priorizar el orden en que serán mostrados. La consulta en *Active Record* es la siguiente: *"$query = $this->db->order_by('date_start', 'asc')->order_by('likes','desc')->join('users', 'wish.id_user = users.id')->join('type', 'wish.id_type = type.id_type')->from('wish')->get()->result();".* La consulta puede presentar cambios según los criterios de prueba, la finalidad como mínimo es priorizar las publicaciones con mayor *rating* y las más recientes.

Ilustración 8 Código para consultar deseos.

3.5.5 Estructura *MVC* de la vista Profile

La vista Profile, como se ha mencionado, es donde se muestran los datos de algún usuario en selección. Se estableció la siguiente ruta: *"$route['user/(:any)']='admin/profile/$1';"*. La ruta con el nombre User tiene una expresión regular *(:any)* para procesar un parámetro de tipo alfanumérico: *"dominio.com/user/url_usuario"*. La ruta *"admin/profile/$1"* a la que hace referencia al tener la terminación *$1*, hará que el método Profile (perteneciente al controlador Admin) reciba *"url_usuario"* ingresado en el navegador. Al igual que con las otras dos vistas, se usó un archivo aparte para las funciones, *functions_profile.php*, almacenado en la ruta de directorios "*application/views"*, con las funciones a donde hará referencia un archivo JS para usar *Ajax*. La vista con el archivo *functions_people.php* también está intercedida por un método en el controlador Admin, con el nombre *Ajax_people*. Se le configuró una ruta también para recibir el mismo parámetro "url_usuario" que se ingrese en el navegador, de la siguiente forma: *"$route['funct_profile/(:any)']='admin/ajax_profile/$1'*;". El método Profile en el controlador Admin se comunica con el método profile de la clase modelo Users, en éste se encuentra la siguiente consulta:" *$users = $this->db->get('users')->result();"*. De esta manera obtiene todos usuarios existentes en el controlador; en éste se utiliza una función *foreach* para guardar en el arreglo *$uri* a todos los *url* almacenados en la base datos en la tabla Users; ahí mismo se usa la función *if (in_array($name, $uri))*, donde $name es el "url_usuario" ingresado en el navegador. De esta manera se revisa si el usuario al que se intenta hacer referencia existe, si existe el método prepara la vista Profile y envía el "url_usuario" al método Ajax_profile. En cambio, si no existe, se redirige a otra vista para mostrar que dicho usuario no existe. El método Ajax_profile se comunica con el modelo función *Uri_profile* para hacer la siguiente consulta en Active Record: *"$profile = $this->db->where('uri', $uri)->get('users')->row();*". En esta consulta se usa la variable $uri que es el "uri_usuario". Éste ya fue revisado como existente en la base datos, por lo tanto la consulta no puede fallar y almacenaría en *array* Profile los datos del usuario para ahora sí ser enviado a la vista Profile del usuario. En esta vista no hace falta el candado de sesión abierta o cerrada, puesto que es para visita del público en general, no importa si alguien está suscrito al portal, mientras se sepa la ruta para llegar a la vista Profile de algún

usuario, se puede ver; aunque su mayor propósito consiste en que los usuarios puedan consultar los deseos de otros usuarios.

Ilustración 9 Código para mostrar vista Profile.

3.5.6 Uso de Bootstrap

Se usó Bootstrap como un conjunto de herramientas de código para darle un mejor diseño a las vistas. Para la pantalla de inicio y la pantalla de People se tiene el mismo encabezado con botones de navegación a la izquierda, el logo de la aplicación en el centro y el botón para cerrar sesión a la derecha. Las particularidades de cada vista se describen a continuación:

•Vista Home.- Es la vista que sale por default al momento de acceder con el usuario y contraseña. Aparece un mensaje de bienvenida con el nombre de usuario con sesión abierta debajo del encabezado, una sección vertical a la izquierda con la foto de perfil del usuario, la cual al hacer clic permitirá abrir un modal para poder cambiarla y debajo, la información del usuario con un botón para crear un deseo. Se tienen formularios ya definidos para tres tipos de deseos por el momento, con el fin de delimitar el concepto de

interacción, ayudando a que pudiera ser más fácil para los usuarios el poder ingresar o seguir las historias relacionadas con su tema de interés. Se puede comprender que no es lo mismo una meta de superación personal que otra, por ejemplo: aprender a cocinar cierto platillo, sacar buenas calificaciones, bajar/subir de peso, conseguir trabajo, graduarse o titularse de algún curso etc.

Finalmente se tiene la sección central donde aparecen los deseos ya publicados por el usuario (en pestañas), iniciando con una lista cronológica de los deseos publicados, seguido de los deseos ya cumplidos (con estatus *fullfill* y en dorado) y por último divididos los deseos por categorías. Es importante mencionar que en la sección de todos los deseos es donde se puede hacer la administración de los mismos, ya sea borrar, modificar o el portal cuenta con la posibilidad de retroalimentar la publicación con hasta tres fotografías (en formato .jpg o .png.) que pueden ser remplazadas por el usuario en todo momento mientras la publicación no haya vencido su vigencia. Al momento de retroalimentar la publicación por lo menos con una fotografía, la publicación será candidata a ser evaluada por el algoritmo del portal.

•Vista People: Esta es la pantalla donde el usuario, fuera de su perfil, puede ver publicaciones hechas por otros usuarios. La vista puede estar puesta en orden cronológico, pero también por orden de *rating* (popularidad de las publicaciones), medida obtenida por la aceptación manifestada por los usuarios hacia cada una de las publicaciones mediante porcentajes de aceptación. Así podrán aparecer las publicaciones que realmente tienen un seguimiento o finalidad por cumplir una meta, en teoría. También se pueden ver las imágenes subidas en cada deseo (Si cuenta con ellas.)

En el uso de Bootstrap sobre las vistas se tiene las clases CSS más comunes que este *framework* ofrece para el *front-end*. El uso de Bootstrap es por medio de la etiqueta *HMTL *; se debe agregar el nombre de la clase a utilizar de la siguiente forma: * *. Entre ellas: *panel*, sus derivadas *panel-heading* y *panel-body* para la muestra de cada uno de los deseos. En cada vista se tiene la clase *container*, utilizada para manejar los tamaños en el *display*, a dicho término se le conoce como *grid*. Una de

las principales finalidades de este tipo de *framework* es sustituir la vieja manera de dividir todo en las viejas etiquetas <*table*> <*tr*> y <*td*> para hacer tablas. La clase *container* actúa como si fuese la etiqueta <*table*>, mientras que para <*tr*> se encuentra la clase *row* y para <*td*> la clase *col* en sus distintas variantes. El *grid* bajo la clase *container* se verá dividido de izquierda a derecha en doce columnas predeterminadas, esto es controlado por medio de la clase *col*, la cual tiene diversas variantes, entre ellas las siguientes*: xs, sm, md y lg; extra-small, small medium, medium y extra large*, respectivamente. Las dos primeras para dispositivos móviles y tabletas, y las dos últimas para monitores de computadora. Para dar un ejemplo, si se requiere un *grid* centrado con diez columnas centradas, sería de la siguiente forma dentro de la etiqueta HTML *<body>: " "*; la clase *col* estaría ocupando diez columnas en *xs* y a la vez con la variante *offset* agregará una columna invisible en cada lado para centrar la clase *container*. Cabe mencionar que la variante *xs* es adaptable en todos los displays, esto puede facilitar al momento de empezar a desarrollar la vista en HTML, pero al momento de desarrollar para los diversos tipos de *display* será necesario usar la variante adecuada para cada dispositivo; se pueden usar las cuatro variantes simultáneamente en la clase de la etiqueta <*div*>, pero es recomendable usar pares; una para referirse a los dispositivos *xs, sm* y otra para los dispositivos md, lg. Si se trata de usar columnas de más, el *framework* no trabajará adecuadamente. Se recomienda usar el inspector de código CSS que ofrecen los navegadores para poder observar cómo se comporta el *grid.*

3.6 Diseño del algoritmo para clasificación de deseos.

Cada publicación cuenta con un tiempo de vigencia y la vez con un estatus como "cumplido" o "no cumplido". La publicación, al permanecer como "no cumplido" su vigencia puede vencer bajo ciertos parámetros y entonces darse de baja (eliminarse). El algoritmo del programa es programado para funcionar de manera automática y será éste quien se encargue de calificar la meta como "cumplida" o "no cumplida". Bajo este funcionamiento del algoritmo se tienen programadas otras funciones para cumplir acciones

de manera automática por parte del portal, tales como: pasar publicaciones vencidas a segundo plano, pasar publicaciones cumplidas al *blog* del usuario, catalogar publicaciones, etc. El algoritmo está programado para catalogar por medio del *rating* y la actividad. En resumen, el algoritmo califica y cataloga la meta o deseo como "cumplido(a)" si éste tiene el 60% del *rating* positivo, de lo contrario será catalogado como "no cumplido(a)". El *rating* se compone de la suma de *likes* y *dislikes*. El *rating* positivo parte del porcentaje de *likes* en el *rating*, por el contrario, el *rating* negativo parte de los *dislikes*. La actividad es la suma de *likes*, *dislikes* y fotografías que se usaron para retroalimentar. La finalidad del parámetro *actividad* es para el momento de mostrar las publicaciones al público suscrito al portal, los deseos o metas publicadas por todos los usuarios. Dichas publicaciones están ordenadas en el orden de las más recientes a las más antiguas y usando cómo factor la actividad para influir en el orden como segunda prioridad. De esta manera, las publicaciones más recientes y con mayor actividad serán las primeras en mostrárseles al público suscrito. Por medio del uso de la base datos y la programación del portal, la vista de las publicaciones se actualiza con cada *refresh* en el navegador.

La dinámica en que los usuarios participarán es un juego, una clase de competencia que consiste en lograr guardar las metas cumplidas de cada uno en un *blog* de deseos personal; al tener varias metas realizadas ellos podrán mostrarlas como sus trofeos. Esto con el fin de lograr que los usuarios interactúen lo mayor posible con el portal o aplicación web. Los usuarios nunca sabrán el funcionamiento del algoritmo que usa el portal por parte de quien o quienes crean el portal; el usuario puede intuirlo poco a poco al experimentar o interactuar y con eso disfrutar la dinámica del juego. Se les podrá mostrar a los usuarios algunos ejemplos o sugerencias de cómo lograr un buen *rating* en sus publicaciones y de las actividades a promover en el portal, pero nunca se les dirá el funcionamiento del algoritmo.

3.7 Pruebas locales de la red social y subir proyecto al dominio.

En la etapa de validación de la red social se crearon múltiples cuentas en forma local, para detectar posibles errores que tuviera la aplicación.

Se subió la aplicación a la nube, montada en un dominio y *hosting* de un servicio pagado. El público en general se puede suscribir. Con el uso del *framework* Bootstrap mencionado previamente, se procuró lograr responsividad web en los diferentes tipos de dispositivos que puedan navegar en Internet, con la finalidad de lograr una vista agradable para quienes les pueda ser más cómodo entrar por medio de un dispositivo móvil.

Como el alcance del proyecto es llegar a un prototipo funcional, las validaciones realizadas fueron exclusivamente en formato local.

3.8 Consideraciones adicionales.

El portal no cuenta con controles para evitar que un usuario publique un concepto erróneo o fuera de lugar en cada categoría, pues el *rating* será el concepto que ayudará darle validez o finalidad a una publicación. Si un usuario publica algo fuera de la dinámica, es probable que vaya a tener un *rating* negativo. En dado caso de que la publicación haya sido apoyada por otros usuarios, que inclusive pueden conocer al usuario quien publicó para lograr un *rating* negativo, el control del portal podría calificarla como realizada, aunque esta publicación no haya tenido que ver con publicar una meta o deseo. En este caso no se considera ilegítimo, pues hubo apoyo por parte de otros para que esta publicación tuviera una relevancia o finalidad, lo cual es el concepto que se busca cumplir en una publicación.

Una de las finalidades del sitio es poder depurar publicaciones con la intención de reducir aquellas que no tengan contenido que realmente aporte a los usuarios, ya que, si no tienen un buen motivo para estar ahí, es posible que no tengan seguimiento. Mientras la publicación permanezca como ‘No cumplida’, ésta tendrá una vigencia, y si no tiene movimientos en ese tiempo, será depurada (o no visible). Cada panel de deseo está estilizado con iconos y números para representar los *likes* y *dislikes*.

El ideal de esta aplicación es evitar comentarios negativos por otros usuarios al leer las publicaciones, basándose en la teoría de que, si ya existen metas “comunes” entre usuarios, probablemente estos se sientan identificados con las metas. Entonces, los demás,

probablemente manifiesten su aceptación y así servirán de motivación hacia la persona que publicó su meta.

La aplicación es web y tiene parámetros para responsividad (compatibilidad con navegadores móviles), con esto podrá usarse en un móvil (celular) y tabletas. Se ha decidido hacerlo web, porque es más compatible con casi cualquier dispositivo, sin importar mucho si es *androi*d o *IOS*. Por lo tanto, no se tendrá una aplicación exclusiva para estas plataformas por el momento.

El prototipo sirve para formar una comunidad de usuarios, por lo tanto, no cabe la posibilidad de un cobro para entrar al sitio.

En ningún momento el prototipo de aplicación garantizará el cumplimiento de las metas o deseos.

IV. Resultados y Discusiones

Se hicieron múltiples simulaciones para ver cómo clasifica el portal por medio de los métodos dispuestos para lograr el objetivo de darle seguimiento a las publicaciones que realmente puedan dejar un buen mensaje a los demás usuarios y descartar aquellas que no aportan alguna novedad o que no fueron del agrado de los espectadores que la siguieron.

Se obtuvo el resultado esperado, ya que los deseos se califican de manera eficiente por medio del algoritmo de clasificación de deseos basado en el *rating*. Aquellos con vigencia terminada y una baja popularidad ya no se pueden observar en la vista People, depurando los deseos que no son necesarios promover entre los participantes, pasándolos a segundo plano.

También se puso a prueba el funcionamiento de los controles del portal, verificando que las reglas establecidas para la clasificación de forma automática y dinámica funcionaran correctamente. Esto se logró al modificar arbitrariamente con estatus 0 la columna *fullfill* de todos o algunos deseos en la base de datos, o modificando las fechas que tenían de entrada. El control automático del portal, de nuevo, logró calificarlas de la misma manera, mostrando que funciona correctamente.

Para lograr estos resultados se han creado múltiples cuentas de prueba para ver el funcionamiento y la interacción del portal. Los diferentes tipos de resultados que se pueden obtener al ingresar un deseo son los siguientes:

Deseo aprobado: el deseo fue aprobado por obtener 4 *likes* contra 1 *dislike*, contiene 3 fotografías, en la categoría “cocina”. De esta forma, tiene un rating positivo del 80%, siendo superior al 60% requerido y tiene más de 1 fotografía requerida; el deseo sería aprobado después de 3 días (ver ilustración 10).

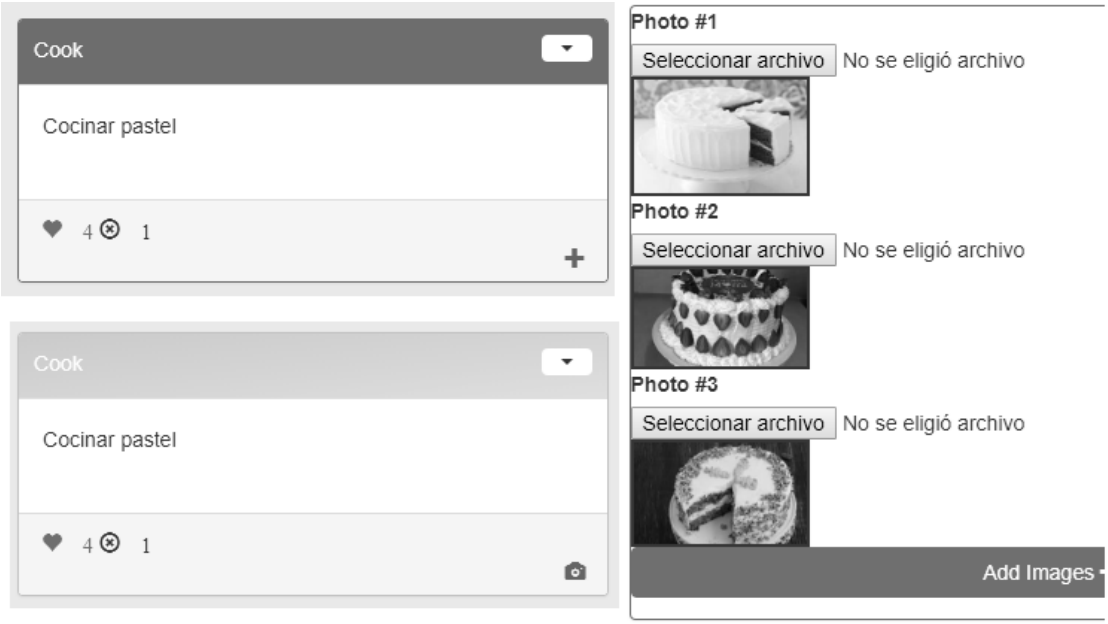


Ilustración 10 Ejemplo de deseo aprobado.

Deseo no aprobado: el deseo tiene 1 *like* contra 0 *dislikes*, no tiene fotografías, está en la categoría ‘‘educacional’’. Aunque posee 100%, no tiene una fotografía, es un deseo educacional y por lo tanto posee mucho tiempo para poder ser retroalimentado por una foto, ya que, si no la tiene, no será aprobado para cuando venza su vigencia.

Ilustración 11 Ejemplo de deseo no aprobado.

Deseo sin *rating*: El deseo no tuvo *likes* ni *dislikes*, pudo tener fotografía o no, pero lo más probable es que no. Es una publicación sin seguimiento por lo tanto el método *fullfill* la ignorará (ver ilustración 12).

Ilustración 12 Ejemplo de deseo sin rating.

Deseo con vigencia vencida: Los deseos con vigencia vencida no serán vistos en la *Vista People*, pues la vista está diseñada para mostrar los deseos más recientes, priorizando los de mayor *rating* positivo y a la vez los que tengan su fecha de caducidad superior a la fecha actual. El deseo podrá seguirse viendo en el *Home* y *Perfil* del usuario, pero ya no se le podrá dar *like* ni *dislike.* El usuario puede eliminarlo y volver a hacer un nuevo deseo con el mismo concepto, para que este nuevo deseo pueda ser calificado por otros usuarios. Si el deseo fue calificado como *fullfill* será mostrado como trofeo, este caso no lo fue.

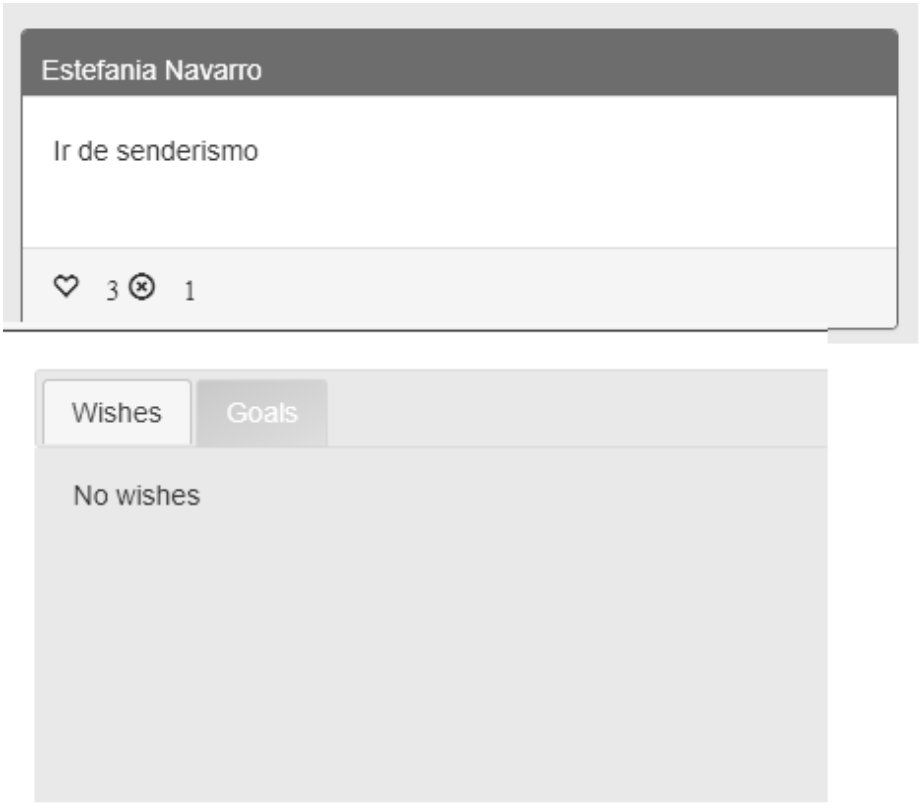


Ilustración 13 Ejemplo de deseo con vigencia vencida.

Deseo en categoría no adecuada: Se pudiera presentar el caso de poner un deseo de cocina en la categoría educacional, tener el *rating* adecuado con las fotografías, pero por ser un deseo de categoría de largo plazo, tardaría mucho en ser aprobado. Podría presentarse un deseo de deportes puesto en la categoría cocina, sin embargo, al necesitar más tiempo para realizarlo y retroalimentarse, su vigencia vencerá pronto, por lo tanto no le dará tiempo para ser calificado por otros, a no ser que suba las 3 fotografías dentro de los 3 días y la gente le dé el *rating* necesario.

Deseo con vigencia recortada: El deseo es de la categoría cocina, tiene 5 *likes* y 15 *dislikes*. Su factor *Actividad* es calculado en 20 y tiene 75% de *rating* negativo, por lo tanto, su tiempo de vigencia será reducido; en este caso es irrelevante si tiene imágenes en su retroalimentación.

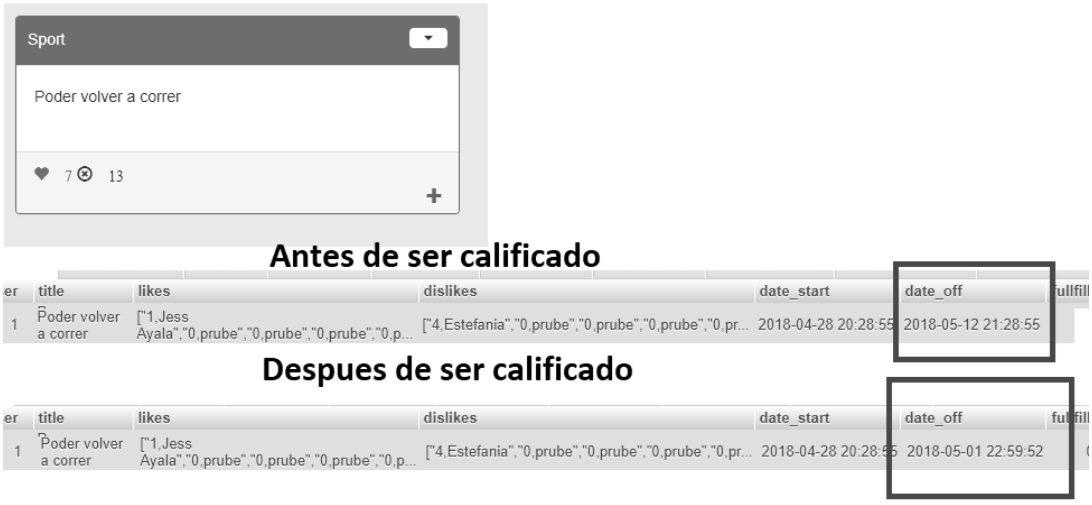


Ilustración 14 Ejemplo de deseo con vigencia recortada.

Además de lograr el objetivo principal, que es la correcta clasificación de los deseos ingresados a la aplicación, la interfaz ofrecida al público ha logrado agradar tanto a personas que no saben del concepto de lenguajes de hipertexto como algunos que trabajan con este tipo de lenguaje. Con esto se comprobó que a pesar de ser vistas muy simples, cuyo propósito sólo es comprobar que el algoritmo descrito funciona correctamente, se logró también aplicar conceptos de usabilidad aprendidos a lo largo de la carrera.

V. Conclusiones

5.1 Conclusiones

Cada publicación, al estar sujeta al *rating*, también está sujeta a un periodo de tiempo para ser vista, ser notada por la audiencia y si no se da a conocer el deseo después de ser publicado, podría ser difícil que otros la noten. En cierto modo, al implementar el sorteo de las publicaciones por medio del *rating* y la fecha de creación, ayuda a que los usuarios se pregunten dónde se encuentran ubicados, entre todos los demás publicados por la audiencia.

Al poner a prueba el algoritmo de procesamiento de los deseos, se contestó la primera pregunta de investigación, ya que la novedad que se puede dar a aquellas personas que ya crearon o que piensan crear una red social, es comprobar que sí se puede buscar tener un control sobre lo que se debería o no se debería publicar en una red social. Con esto se ayuda a los usuarios a durar menos tiempo en cada sesión, al no tener que buscar buenas historias entre actualizaciones que no le despiertan ningún interés, ya que las publicaciones con bajo *rating* no necesariamente se eliminan, pero sí se pasan a un segundo plano (contestando así la segunda pregunta de investigación). Por lo mismo, un usuario se puede ver motivado a tener una mayor cantidad de visitas a la red social para darle seguimiento a aquellas publicaciones que lo dejaron intrigado.

Los alcances y limitaciones del proyecto ingresados desde el inicio del mismo, a criterio de quienes desarrollan el presente documento, fueron correctamente definidos. Si bien, el desarrollo de este prototipo era comprobar que era correcto el funcionamiento del algoritmo propuesto, no se profundizó en temas que no fueran relevantes para comprobar el objetivo. Para obtener retroalimentación adicional, es necesario que haya actividad por parte de los usuarios en el portal, porque la aplicación no es autónoma, no tiene manera de generar contenido automático y de forma aleatoria y se necesitaría que los usuarios se sientan incentivados a publicar contenido y de esta manera sumarse a la dinámica del portal.

En cuanto a la última pregunta de investigación, para propósitos de las pruebas hechas, sí fue útil establecer categorías de deseos para poder llevar un control sobre ellos.

5.2 Trabajos futuros

Sería ideal dar a notar las nuevas publicaciones y sus cambios en el instante que ocurren. Una manera de hacerlo es agregando un método llamado *push notification*s, muy común en las redes sociales, el cual consiste en representar un número al lado de un mensaje o pestaña. Por ejemplo, podría ser: "People (5)", haciendo referencia a la vista People y donde el número daría a notar la cantidad de nuevas publicaciones que aparecieron desde el último ingreso a esa vista. Algo a tomar en cuenta en este tipo de notificaciones es poder darle al usuario la posibilidad de activarlas o desactivarlas, pues en ocasiones llegan a ser molestas. También se pueden usar para dar a conocer que una publicación por parte del usuario obtuvo un *like* y/o *dislike* y mostrarle un avance parcial del *rating* que tiene. En este caso, no sería del todo necesario agregar la opción de desactivar esta notificación pues es muy probable que a una persona le interese saber si su notificación tiene seguimiento por parte de otros. Otra aplicación podría ser, cuándo una publicación pasa a ser trofeo o con estatus *fullfill* o cuándo fue descartada por su mal *rating*.

Como parte del proyecto actual no se consideró aplicar el concepto de *autorefresh*, el cual consiste en que una página cargue o muestre nuevo contenido sin tener que hacer *refresh* de forma manual en el navegador. Al igual que el concepto anteriormente explicado, la posibilidad de notificar nuevo contenido podría ayudar a incentivar a la audiencia sumarse a la dinámica del portal. Esto dependería de una posible mejora de la metodología actual, e incluso sería necesario poner a prueba el funcionamiento de este concepto y del anterior subtema para poner a prueba diferentes variantes que puedan surgir, sacar análisis estadísticos y con ello dejar la variante con mejores resultados.

Los temas anteriormente explicados no se implementaron desde el inicio del proyecto porque no eran realmente necesarios para lograr el objetivo planteado, pero pueden contribuir a darle un *plus* a la aplicación.

Uno de los retos en este proyecto fue lograr hacer funcionar la metodología explicada, ponerla a prueba y con ello saber si la metodología fue funcional. Bajo las pruebas lo fue, sin embargo, el siguiente paso sería optimizarla. Se tiene un controlador que funciona de manera automática, dicho método funciona por medio de factores como el *rating* y la actividad en las publicaciones. Una manera en que esto pudiera funcionar es el

hecho de sustituir o sumar a este método con una red neuronal, donde sus entradas serían los mismos factores usados en el método aplicado. En el método *fullfill* se aplicaron porcentajes de manera arbitraria, no se hicieron pruebas para observar o estudiar si esos porcentajes son acertados o no.

Un sitio web, al igual que cualquier proyecto que requiera audiencia, está sujeto a ser actualizado cada cierto periodo. En la aplicación del proyecto que se describe en este documento se pueden presentar actualizaciones con la intención de invitar a audiencia real al portal y buscar que tenga un impacto a futuro. Sin embargo, se respetarían los conceptos expuestos en la metodología explicada, es decir no serían remplazados. La finalidad de estas posibles actualizaciones es apoyar el concepto de la metodología para obtener resultados óptimos.

Referencias

[1] Prime Publicaciones Electrónicas, «Internet y la World Wide Web,» 2008. [En línea]. Available: http://www.paralibros.com/passim/p20-tec/pg2050ci.htm. [Último acceso: 17 Septiembre 2016].

[2] D. Sanchez-Molina, «Historia de hotmail.com,» 24 Enero 2011. [En línea]. Available: http://descubrehotmail.net/historia-hotmail-com. [Último acceso: 17 Septiembre 2016].

[3] Google, «Google sities,» 28 Febrero 2008. [En línea]. Available: https://sites.google.com/site/listaredessociales/listaredessociales/classmates. [Último acceso: 17 Septiembre 2016].

[4] M. Izquierdo, «Historia de myspace,» 11 Abril 2013. [En línea]. Available: http://www.factorde.com/blog/historia-de-myspace. [Último acceso: 2016 Septiembre 2016].

[5] N. Solorzano, «Historia de HI5,» 13 Noviembre 2012. [En línea]. Available: http://nelsonsolorzano.blogspot.mx/2012/11/historia-de-hi5.html. [Último acceso: 17 Septiembre 2016].

[6] D. Fincher, Dirección, *The social network.* [Película]. Estados Unidos: Scott Rudin Production, 2010.

[7] S. Lara y C. Naval, «Participación en la sociedad del conocimiento y redes sociales,» 27 Octubre 2014. [En línea]. Available: http://dadun.unav.edu/bitstream/10171/36924/1/Participaci%C3%B3n%20en%20la%20sociedad%20del%20conocimiento.pdf. [Último acceso: 07 Octubre 2016].

[8] M. Zamora, «Redes sociales en internet,» 14 Noviembre 2016. [En línea]. Available: http://www.maestrosdelweb.com/redessociales/. [Último acceso: 08 Octubre 2016].

[9] W3Schools, «CSS Tutorial,» 2016. [En línea]. Available: http://www.w3schools.com/css/. [Último acceso: 08 Octubre 2016].

[10] Universidad Carlos III de Madrid, «Ejemplos TIW,» 2015. [En línea]. Available: http://www.lab.inf.uc3m.es/~a0080802/RAI/mvc.html. [Último acceso: 08 Octubre 2016].

[11] I. Alvarado, «¿Que es FrontEnd Y Backend en la programación web?,» 12 Abril 2014. [En línea]. Available: http://serprogramador.es/que-es-frontend-y-backend-en-la-programacion-web/. [Último acceso: 08 Octubre 2016].

[12] J. Malik, «Alien vault,» 03 Octubre 2016. [En línea]. Available: https://www.alienvault.com/products/ossim. [Último acceso: 08 Octubre 2016].

[13] F. Karinthy, Láncszemek (Chains), Budapest, 1929.

[14] D. Watts, Six Degrees: The Science of a Connected Age, New York: W. W. Norton, 2004.

[15] F. Santos, Analisis de redes sociales, Madrid: Centro de investigaciones sociologicas, 2006.

[16] C. Anderson, «The log tail,» 01 Octubre 2004. [En línea]. Available: www.wired.com/2004/10/tail. [Último acceso: 09 Octubre 2016].

[17] B. Guirren, La microeconomia, Paris: Universidad de Paris, 1998.

[18] P. Kotler, Fundamentos de marketing, México: Pearson, 2008.

[19] R. Jara y A. Garnica, Medicion de audiencias de television en México, México: Editorial Patria, 2014.

[20] S. Krug, No me hagas pensar, Madrid: Pearson, 2014.

[21] J. Barnes, Social network, Austrailia: Addison-Wesley , 1972.

[22] C. Villalba Quezada, Redes sociales: Un concepto con importantes implicaciones en la intervención comunitaria.

[23] R. Bourdon, «Wampserver,» 02 Octubre 2016. [En línea]. Available: http://www.wampserver.com/en/. [Último acceso: 11 Octubre 2016].

[24] Apache friends, «XAMPP Apache + MariaDB + PHP + Perl,» 29 Septiembre 2016. [En línea]. Available: https://www.apachefriends.org/es/index.html. [Último acceso: 11 Octubre 2016].

[25] M. Alvarez, «Que es HTML5,» 14 10 2009. [En línea]. Available: http://www.desarrolloweb.com/articulos/que-es-html5.html. [Último acceso: 11 Ocubre 2016].

[26] Bootstrap, «Gets bootstrap,» [En línea]. Available: http://getbootstrap.com/. [Último acceso: 12 Octubre 2016].

[27] D. Gandi, «Fontawesome,» 10 Octubre 2016. [En línea]. Available: http://fontawesome.io/get-started/. [Último acceso: 12 Octubre 2016].

[28] EllisLab, «Codeigniter,» 19 Septiembre 2016. [En línea]. Available: http://www.codeigniter.com/. [Último acceso: 12 Octubre 2016].

[29] Don Ho, «Notepad about,» 10 Octubre 2016. [En línea]. Available: https://notepad-plus-plus.org/. [Último acceso: 13 Octubre 2016].

[30] Jetbrains, «Phpstorm,» 03 Octubre 2016. [En línea]. Available: https://www.jetbrains.com/phpstorm/. [Último acceso: 15 Octubre 2016].

[31] F. Oberta, Estudio para la migración de Servidores a un Servicio Nube Híbrida, Cataluña: Universidad de Cataluña, 2014.

[32] Rankerizer, «Rankerizer,» 15 Diciembre 2011. [En línea]. Available: http://www.rankerizer.com/download.php. [Último acceso: 22 Octubre 2016].

[33] Majestic, «Majestic,» 11 Octubre 2016. [En línea]. Available: https://es.majestic.com/. [Último acceso: 22 Octubre 2016].

[34] Instituto Tecnologico de Matehuala, 18 Octubre 2016. [En línea]. Available: https://programacionwebisc.wordpress.com/2-5-metodologias-para-el-desarrollo-de-aplicaciones-web/. [Último acceso: 22 Octubre 2016].

[35] P. Farías, «Metodologias de desarrollo para aplicaciones web,» 05 Junio 2015. [En línea]. Available: http://documents.mx/documents/metodologias-de-desarrollo-para-aplicaciones-web.html#. [Último acceso: 22 Octubre 2016].