Prototipo aplicación web para la promoción de cultura de paz en la comunidad Unadista

Giovany Gáfaro.

Universidad Nacional Abierta y a Distancia Unad.
Escuela de Ciencias Básicas, Tecnología e Ingeniería (ECBTI).
Ingeniería de sistemas
Pamplona
2016

Giovany Gáfaro.

Director
Jorge Enrique Ramirez Montañez.

Universidad Nacional Abierta y a Distancia Unad.
Escuela de Ciencias Básicas, Tecnología e Ingeniería (ECBTI).
Ingeniería de sistemas
Pamplona
2016

Para Alix, su amor y apoyo, brindan la total libertad ante una pantalla oscura, para jugar con unos y ceros.

Agradecimientos

A los profesionales pertenecientes a la Unad, quienes, de manera anónima, día a día y tomando el tiempo de sus hijos y familia, se entregan de manera responsable e idónea en el acompañamiento académico de sus estudiantes, enalteciendo su labor de tutores.

A la profesora Mabel G. y demás sicólogos y sociólogos de nuestra universidad quiénes muy amablemente me orientaron en la comprensión del campo de la resolución de conflictos.

Resumen

La nación colombiana actualmente desarrolla procesos de búsqueda de la paz estable y duradera. Actualmente no existen aplicaciones web enfocadas en resolución de conflictos. El documento describe el proceso de desarrollo de un prototipo mediante metodología Ágil con Visual Paradigm para manejo de herramientas del conflicto. Logrando el prototipo de una web App para la promoción de cultura de paz en la comunidad Unadista a partir del diagnóstico sobre las funcionalidades e historias de usuario para la especificación de líneas de tiempo del conflicto y finalmente ejecutar el despliegue a producción del aplicativo. De la misma manera se ilustra la programación desarrollada mediante el framework laravel para el lenguaje Php, mostrando las bondades en el uso de frameworks como la escalabilidad y la disponibilidad de funcionalidades previamente configuradas, entre estas enrutado, autenticación y gestión de protocolo Http. El autor concluye que el trabajo planteado permitirá a la Universidad Nacional Abierta y a Distancia iniciar el liderazgo único en aplicaciones en el campo descrito, ampliando el conjunto de herramientas y la apertura de nuevas líneas de investigación, fortaleciendo las actividades que la universidad desarrolla dentro del contexto de la Ley 1732 01 de septiembre de 2014 y el impacto social ya liderado a nivel nacional.

Tabla de Contenidos vii

Introducción 1
Objetivos 2
Objetivo General 2
Objetivos específicos 2
Planteamiento del problema 3
Definición del problema 3
Formulación del problema 3
Justificación 4
Marco de referencia 6
Antecedentes De La Investigación 6
Marco teórico 7
Cultura de paz. 7
Conflictos y violencia. 7
Análisis del conflicto. 8
Desarrollo de aplicaciones Agile 9
Aplicaciones Web o WebApps. 12
Servidores, máquinas virtuales y control de versiones 12
Modelo de datos. 14
Características de compatibilidad y requerimientos de la aplicación. 14
Automatización de pruebas y depuración 14
Marco conceptual 15
Marco Legal 15
Marco Contextual 15
Desarrollo de aplicativos y prácticas de diseño 17
Metodología 19
Tipo de investigación. 19
Línea de investigación. 19
Alternativa de trabajo de grado 19
Población y muestra. 20

Instrumentos. 20
Etapas. 21
Cronograma 22
Desarrollo del proyecto 23
Determinación de requerimientos 23
Objetivo 23
Requerimientos técnicos. 23
Análisis de requerimientos. 25
Modelo de datos 38
Desarrollo del prototipo 41
Configuración del entorno 41
Configuración librerías externas 42
Configuración control de Versiones 43
Sprint 1/2 44
Sprint 2/2 50
Ejecución de pruebas 65
Despliegue a entorno de producción 70
Conclusiones 73
Recomendaciones 74
Bibliografía 75

Lista de tablas

Tabla 1.Cronograma. 19
Tabla 2.Historias de usuario. 23
Tabla 3 Tabla relacional prototipo 39

Lista de figuras x

Figura 1 Card para historia de usuario. 10
Figura 2 Conversation en Visual Paradigm. 10
Figura 3 Confirmación en Visual Paradigm. 10
Figura 4 Actor, enlace y caso de uso 11
Figura 5 Organigrama Unad. Fuente autor. 17
Figura 6 Diagrama casos de usos prototipo 25
Figura 7 Editar Herramienta 26
Figura 8 EliminarHerramienta 27
Figura 9 Crear nuevo usuario 29
Figura 10 Editar cuenta de usuario 30
Figura 11 Eliminar cuenta de Usuario 31
Figura 12 Términos y condiciones 32
Figura 13 Detalles herramienta 34
Figura 14 Usuario signup 36
Figura 15 Crear nuevo conflict 38
Figura 16 Tabla relacional prototipo 39
Figura 17 Ccontrol de versiones en gitlab 43
Figura 18 Portada prototipo 49
Figura 19 Gestión de usuario en prototipo 49
Figura 20 Cuenta de usuario en prototipo 50
Figura 21 Migraciones prototipo 53
Figura 22 Menú conflictos y herramientas en prototipo 62
Figura 23 Gestión de conflictos en prototipo 63
Figura 24 Gestión lineas de tiempo 64
Figura 25 Gestión hechos del conflicto en prototipo 64
Figura 26 Hechos y visor línea de tiempo interactivo 65
Figura 27 Testing del prototipo 70
Figura 28 Homepage Hostgator 71
Figura 29 Configuración dase de datos a producción 71

Figura 30 Despliege de prototipo a producción 72

Introducción

En la actualidad el estado colombiano se encuentra cada vez más cerca a tiempos de postconflicto. Presente, el cual requiere de cada uno de los colombianos cambiar actitudes de que alguien pierda para que otro pueda ganar, donde cada uno de los espacios es de relación intra e interpersonal sean desplazados desde la concepción de victoria o derrota hacia la resolución pacífica de problemas y conflictos.

En el ámbito de la Universidad Nacional Abierta y a Distancia ya se vienen liderando trabajos en relación con la paz como lo es el programa Campo Unad sembrando un país en red para la paz, como estrategia de movilidad formativa y productiva para la población rural colombiana, contexto en el cual se pretende contribuir con este trabajo para el desarrollo de un Prototipo aplicación web para la promoción de cultura de paz en la comunidad Unadista, que permita la adhesión de tecnologías software al manejo de herramientas para la resolución de conflictos, específicamente líneas de tiempo.

La exposición de contenidos de este trabajo comienza con la descripción de los objetivos y el desarrollo del problema donde se plantea el requerimiento detectado frente a la resolución de conflictos y el uso de tecnologías. Seguido se presenta el marco teórico, conceptual y contextual del proyecto. Se continúa con el detalle del proceso de construcción del prototipo mediante una variante de la metodología Ágil para el desarrollo de software con herramienta visual paradigm, desglosando el trabajo en dos sprint. Finalmente se observará de manera general la aplicación de pruebas y despliegue a producción del prototipo alcanzado y las respectivas conclusiones.

Objetivos

Objetivo General

Desarrollar un prototipo de aplicación Web para la promoción de la cultura de paz en la comunidad académica de la UNAD

Objetivos específicos

- Realizar un diagnóstico sobre las funcionalidades deseables en una aplicación web para la promoción de la paz en la universidad Nacional Abierta y a Distancia Unad.
- Registrar las historias de usuario de una aplicación web para la promoción de la paz en la comunidad universitaria de la Unad.
- Configurar las características de despliegue a producción de la aplicación web para la promoción de la paz en la universidad Nacional Abierta y a Distancia Unad.

Planteamiento del problema

Definición del problema

La Asamblea de las naciones unidas en su declaración sobre una cultura de paz, reconoce la importante función que sigue desempeñando la educación, la Ciencia y la Cultura en la promoción de una cultura de paz, planteada como un estilo de vida basado en el respeto a la vida, el fin de la violencia y la promoción y la práctica de la no violencia por medio del diálogo y la cooperación. (Naciones Unidas, 1999)

Para el logro de dicho propósito recomienda la adopción de medidas como la ampliación las iniciativas en favor de una cultura de paz emprendidas por instituciones de enseñanza superior de diversas partes del mundo y el fortalecimiento de las actividades de las entidades pertinentes destinadas a impartir capacitación y educación, en las esferas de la prevención de los conflictos y la gestión de las crisis, el arreglo pacífico de las controversias y la consolidación de la paz después de los conflictos(Naciones Unidas, 1999)

El estado colombiano con el fin de garantizar la creación y el fortalecimiento de una cultura de paz en el país, a través del congreso de la república aprueba la ley 1732 por la cual se establece la cátedra de la paz en todas las instituciones educativas del país. De igual manera el decreto 1038 dicta: las instituciones educación superior desarrollarán la Cátedra de la paz en concordancia con sus programas académicos y su modelo educativo, para lo cual podrán definir las acciones educativas que permitan a la comunidad contar con espacios de aprendizaje, reflexión y diálogo para la vivencia de la paz.

Formulación del problema

La promoción de cultura de paz, desde el ámbito de análisis de conflictos puede ser desarrollado mediante herramientas como el árbol de conflicto, la dona o cebolla, las pirámides y líneas de tiempo. Ante la falta de un medio innovador tradicionalmente estas

herramientas son desarrolladas por los individuos participantes de manera manual con papel y lápiz, siendo necesario el avance hacia la gestión de estas herramientas mediante APPs (Fisher, 2000).

En este contexto y frente al amplio uso de las Tecnologías de la Información y la comunicación Tics que se desarrolla en la Universidad abierta y a distancia ¿Cómo pueden participar los diferentes actores académicos de la Universidad Abierta y a Distancia Unad, en la promoción de cultura de paz?, junto al aprovechamiento de un medio innovador que permite la interactividad, el registro y la participación que las aplicaciones proporcionan, desde los diferentes medios tecnológicos en red.

Justificación

La página web de El Ministerio de Tecnologías de la Información y las Comunicaciones se señala como objetivos "presentar los programas, iniciativas y proyectos que vienen acercando las TIC a la gente, proporcionándole beneficios y oportunidades" lo cual constituye una clara invitación y oportunidad para desde el campo del desarrollo software, generar plataformas digitales en diversos ámbitos, incluido la promoción de la paz (MinTic, 2012)

Es un hecho, en la actualidad en Colombia se desarrolla un diálogo directo entre el gobierno y las FARC. Requiriendo el compromiso de las instituciones de carácter público como las instituciones universitarias en la promoción de la paz, siendo tecnologías web un medio idóneo, de bajo costo y alto impacto para el desarrollo de propuestas relacionadas, en concordancia con las intencionalidades del ministerio Tics Colombiano (Gobierno de Colombia, 2012)

El producto de este trabajo de grado de igual manera puede ser un componente en el impacto de la proyección social de la Universidad Abierta y a Distancia UNAD y en la misma medida de acuerdo a la ley 1732, donde se establece la Cátedra de la Paz como de obligatorio cumplimiento en todas las instituciones educativas del país, abordar el inicio de desarrollo de aplicaciones para la paz de manera innovadora y creativa, en el programa de ingeniería de sistemas de la Universidad

Finalmente, la formación específica al campo profesional del programa ingeniera de software en la Unad, abarca un componente básico en programación web, por lo cual el desarrollo del proyecto constituye desde el punto de vista teórico, un espacio de indagación más especializado sobre el conocimiento existente en desarrollo de software enfocado a la divulgación de la paz a través de plataformas virtuales, desde las tendencias de desarrollo de aplicaciones web. Desde el punto de vista metodológico, constituye la oportunidad para la ejecución de métodos modernos de desarrollo ágil de software para la construcción de productos informáticos con calidad y profesionalismo, aplicado en el desarrollo de un prototipo de aplicación web para la paz.

Marco de referencia

Antecedentes De La Investigación

Frente a la promoción y la cultura de la paz desde los ámbitos académicos varias son las experiencias encontradas. Giraldo, 2015 menciona entre ellas, la universidad Autónoma con la emisión de ‘Basta ya’ formato de televisión y la divulgación del proceso de paz y perspectivas del posconflicto. La Universidad del Valle ofrece cursos de pregrado como Estudios Políticos y Resolución de Conflictos. Mientras que la Universidad Libre permite que programas de psicología y enfermería asuman un rol fundamental en la educación para la posguerra con la rehabilitación psíquica de las víctimas.

Desde el campo del desarrollo software en relación con áreas del conflicto y la paz, Vitimalz es una App para comprender la Ley de Víctimas (1448 de 2011), InmoVic ayuda a ciudadanos y funcionarios que brindan orientación a las víctimas frente a Fondo para la Reparación de las Víctimas (Redacción El tiempo, 2014). Un sondeo sobre las tiendas de Android y Apple también muestra que no existe ninguna aplicación a la fecha relacionada directamente a herramientas para el análisis de conflictos.

La universidad Abierta y a Distancia Unad en relación con la paz y resolución de conflictos ha desarrollados actividades de investigación y promoción entre ellas, las siguientes:

- Conversatorio "La paz en Colombia, retos e implicaciones: una mirada desde lo local a lo internacional", evento realizado por la Universidad Nacional Abierta y a Distancia -UNAD-, Centro de Educación Abierta y a Distancia -CEAD- Medellín "Miguel Antonio Ramón Martínez", en alianza con la Escuela de La Paz, con sede en Grenoble, Francia, que integra la Red Franco Colombiana Solidaridad (Unad, 2016)
- CAMPOUNAD, sembrando un país en red para la paz, estrategia de movilidad formativa y productiva para la población rural colombiana, mediante el uso adecuado de las Tecnologías de la Información y

Comunicación, con impacto en la vida social, productiva y económica del campesino y su familia. (Unad,2016)

Marco teórico

Cultura de paz.

Según la Unesco, es la presencia de valores, actitudes y conductas para el manejo de conflictos mediante el diálogo y la negociación, evitando la violencia y garantizando a todos los individuos el ejercicio de principios de libertad, justicia y democracia para su participación en el desarrollo y reconstrucción de las sociedades. De acuerdo con Salamanca (2008), la Cultura de paz busca:

1. Asegurar que los conflictos inherentes a las relaciones humanas se resuelven sin violencia.
2. Asumir que la paz y los derechos humanos son indivisibles y todo el mundo la preocupación.
3. Emprender una tarea multidimensional que requiere la participación de las personas en todos los niveles.

4. Contribuir al fortalecimiento de los procesos democráticos.

5. Garantizar que se desarrolle dentro de un proyecto de movilización completa de todos los medios de la educación, tanto formal como no formal, y de la comunicación.

6. Aprender y usar nuevas técnicas para la gestión pacífica y resolución de conflictos.

7. Cooperar con el desarrollo sostenible, el desarrollo endógeno, humano equitativo; no puede ser impuesta desde el exterior.

Conflictos y violencia.

Los conflictos hacen parte de la dinámica de las interacciones humanas sean intrapersonales, organizacionales o de carácter social y global. En general se establece claramente la separación de conflicto y violencia (Fisher, 2000):

● Conflicto es una relación entre dos o más partes (individuos o grupos) que tienen o consideran que tienen objetivos incompatibles.

● La violencia consiste en acciones, palabras, actitudes, estructuras o sistemas que provocan daño físico, psicológico, social o ambiental y/o que limitan que las personas alcancen todo su potencial humano.

Análisis del conflicto.

De acuerdo a Fisher, el análisis del conflicto comprende el conjunto flexible de técnicas, prácticas y herramientas de carácter práctico que permite el entender la dinámica, tendencias y antecedentes del conflicto desde la variedad de perspectivas de los involucrados. Las siguientes son algunas de las siguientes herramientas que el autor propone:

- Líneas de tiempo del conflicto
- Mapeo Del Conflicto
- El Triángulo Acc del conflicto
- La Dona del conflicto
- El Árbol De Conflicto
- La pirámide del conflicto
- Líneas del tiempo.

De acuerdo a Guzmán (2014), la herramienta para el análisis de conflicto Línea de tiempo, constituye una gráfica que muestra los hechos en relación con el tiempo, de igual manera menciona sus propósitos y usos:

1. Propósito:
    - Mostrar diferentes visiones de un conflicto;
    - Clarificar las diferentes perspectivas;
    - Identificar los hechos más importantes para las partes.
2. ¿Cuándo utilizarla?
    - Al inicio del proceso;
    - Más tarde en el proceso para definir estrategias;
    - Cuando hay diferencias y desconocimiento de los hechos;

- Ayudar a entender las diferentes perspectivas.

De manera tradicional la construcción de líneas de tiempo se desarrolla mediante paleógrafo o láminas de hojas y marcadores siguiendo de manera aproximada la siguiente secuencia (Fundación Unir, 2008):

1. Identificar el periodo de análisis para armar la gráfica.
2. Documentar los eventos más importantes del conflicto
3. Discutir las diferentes visiones y percepciones sobre las causas de los eventos y efectos importantes.
4. Incluir los momentos y sucesos que los actores consideren relevantes (Fechas).
5. Marcar y subrayar las acciones que detonaron el conflicto (Hechos).

Desarrollo de aplicaciones Agile

Marco de trabajo y herramienta que facilita el desarrollo, diseño e implementación de sistemas de información bajo el un método de desarrollo ágil para cada equipo de desarrollo, mediante de técnicas ágiles como historia de usuario, sprint, wireframes.

En la figura 1, las historias de usuario se plasman en notas que captura lo que el usuario hace o necesita hacer como parte de su trabajo. Cada historia consiste en una corta descripción desde el lenguaje y punto de vista del usuario, donde se expresa claramente su rol, el objetivo y el valor de negocio de su historia. Los tres aspectos principales para la construcción idónea de historias de usuario, Card (Tarjeta): Historias de usuario como tarjetas. Conversation (Consideraciones, figura 2): Los requerimientos hallados a través del contacto permanente con el cliente y Confirmation (Lista de chequeo, figura 3): los criterios de aceptación de la historia de usuario.

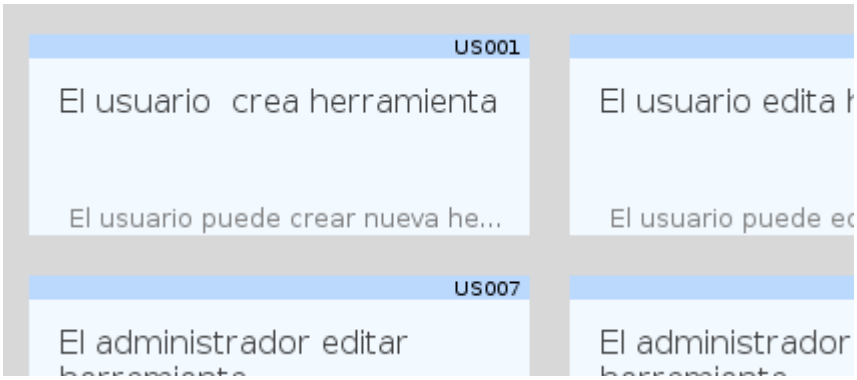

Figura 1. Card para historia de usuario. Fuente: Autor

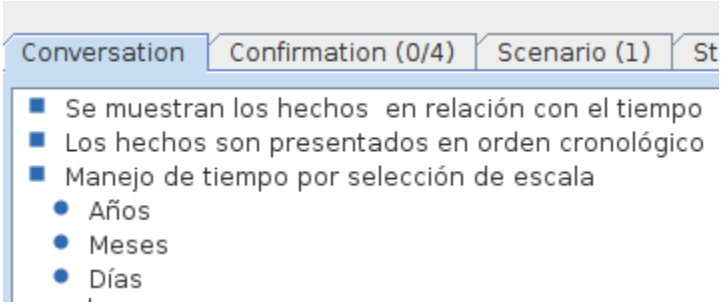

Figura 2. Conversation en Visual Paradigm. Fuente: Autor

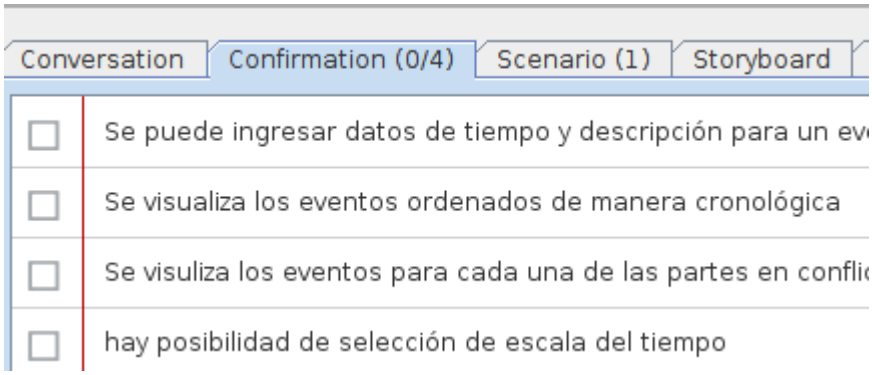

Figura 3. Confirmación en Visual Paradigm. Fuente: Autor

En el caso de los Wireframe, corresponde a bosquejos de pantallas del sistema. Son creados con el ánimo de presentar y explicar las funcionalidades e ideas de diseño al cliente. Los wireframes muestran justo la suficiente información asociada a una característica de interés, cada uno de ellos corresponde a un elemento gráfico que presenta los diferentes componentes y sus posiciones en la pantalla, mostrando como los

contenidos serán organizados y como interactuará con ella el usuario. Sin hacer énfasis en los detalles de cómo lucirá al final las deferentes pantallas.

Los casos de uso constituyen una herramienta para identificar la lógica de negocio del sistema, ayudando a definir su alcance y encontrar los requerimientos que permitirán dar valor a la estrategia y necesidades de negocio. Como se muestra en la figura 4, Los casos de uso pueden ser visualizados mediante diagramas de caso de uso UML. Los casos son mostrados dentro de elipses, de igual manera se dibujan dos elementos más: actor, un rol que interactuará con el sistema y enlace, correspondiente a la interacción o asociación entre el rol y el caso. (Visual Paradigm, 2016).

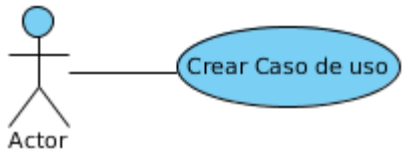

Figura 4. Actor, enlace y caso de uso Fuente: Autor

La recomendación generalizada para la formulación de casos de uso obedece a la estructura verbo + sustantivo, por ejemplo: registrar cuenta, ordenar pedido, crear línea de tiempo. Por último, queda claro que un caso de uso corresponde al alcance de una característica, mientras que una historia de usuario captura una necesidad dentro de dicha característica.

Los Sprint corresponde a un espacio de tiempo. Cada Sprint tiene una fecha de inicio y una fecha de terminación, en las que una historia de usuario debe ser terminada y confirmada. Al inicio del sprint, los stakeholders y los desarrolladores seleccionan y priorizan las historias de usuario que serán ejecutados en cada sprint. La duración de cada sprint debe guardar un equilibrio entre la motivación del programador y la calidad de cada una de las entregas, usualmente comprendido entre un par de semanas a un mes. (Visual Paradigm, 2016).

El Ciclo de vida historia de usuario está definido por seis estados que son:

1. Pendiente: Comprende la comunicación entre cliente y desarrollador para encontrar las historias.

2. Por hacer: Se priorizan las historias y los sprint.

3. En discusión: El cliente confirma los criterios de confirmación de los requerimientos.

4. En desarrollo: comprende la implementación/codificación de las características.

5. Confirmado: La historia es ejecutada por el usuario y la característica.

6. Finalizado: Característica está implementada y se inicia el ciclo para otra historia de usuario

Finalmente, dentro del desarrollo ágil mediante visual paradigm el Kanban Board organiza el desarrollo de características de una aplicación o programa en el cual se dispone las diferentes historias de usuario mediante notas, distribuidas en un tablero de acuerdo al ciclo de vida de las historias de usuario. El trabajo del desarrollador se verá reflejado en el flujo de las notas o cards sobre cada columna.

Aplicaciones Web o WebApps.

Dentro de las categorías de software informático se encuentran las aplicaciones web, definida de la siguiente manera (Luna, 2014):

> WebApp proviene de la conjunción de las palabras en inglés Web Application (Aplicación web). Este tipo de aplicaciones son accedidas mediante la web o una red intranet. Para acceder a ellas, el requisito esencial es contar con un navegador web que permita ejecutarlas. De este modo, una WebApp puede categorizarse como un programa informático, con la diferencia de que se ejecuta desde un browser web. Su estructura está conformada mayoritariamente por HTML, CSS, Javascript, y/o algún otro lenguaje de programación que trabaje del lado del servidor (PHP, ASP.net, Python, Ruby, CGI, Perl).

Servidores, máquinas virtuales y control de versiones

De cara el desarrollo o programación de aplicaciones desde el lado de servidores, es amplísima la oferta de lenguajes entre los principales se encuentran: ASP, JSP, Perl, Php y .Net.

Php es software libre, multiplataforma y de licencia Open Source. Permite la configuración sobre servidores web basados en sistema operativo Linux. Igualmente cuenta con una gran comunidad activa y constituye un lenguaje de amplias capacidades que contiene numerosas librerías y funciones que permiten realizar desde manejo de sesiones a operaciones dinámicas sobre gráficos y documentos PDF.(Yank, 2001).

Otro aspecto interesante de PHP es la amplia disposición de frameworks tales como Cake PHP, Symfony, CodeIgniter, Zend, Yii y Laravel. Este último surge el año 2011 y es desarrollado y liderado por Taylor Otwell, actualmente la versión de soporte LTS (long-term support) corresponde a la versión 5.1, con requerimientos de PHP versión mínima 5.5. (Pecoraro, 2015). Entre las principales características de Laravel se encuentran

- Enrutado: definición de rutas o direcciones de la aplicación mediante URI y funciones closure.
- Middleware: filtrado de peticiones HTTP a la aplicación, además middleware para mantenimiento y manejo de CSRF.
- Controller: permite el manejo lógico de las peticiones a la aplicación mediante clases controller.
- Request: manejo sencillo de peticiones HTTP por inyección de dependencias.
- Responses: libraría para la devolución de respuestas HTTP sean vistas, json o archivos.
- Views: permiten la separación de la lógica y la interfaz de presentación de la aplicación.
- Authentication: pre configuración de sistema para la autorización y autenticación de usuarios ante la aplicación.
- Eloquent ORM: provee la implementación simple y clara de ActiveRecord para el trabajo con base de datos.
- Testing: soporte para PhpUnity métodos Helper para especificación de test expresivos de la aplicación.

Las máquinas virtuales o entornos sobre plataformas software, permiten crear y configurar de manera ágil, reproducible y portable para entornos de desarrollo. Realizando el setup, donde se descarga e instala la plataforma (como vagrant) sobre el sistema operativo huésped, la configuración; de tipo y necesidades a ser instaladas y finalmente la codificación o trabajo mediante ejecución de sencillos comandos como vagrant up, para "encender" la máquina (HashiCorp, 2016).

Modelo de datos.

El modelo de datos o modelamiento de datos es el primer paso en el diseño de bases de datos actúa como un puente entre los objetos del mundo real y la base de datos que reside en el computador o servidor, reduciendo la complejidad de diseño de base de datos para entenderse con mayor facilidad abstracciones que definen las entidades y las relaciones entre ellos. (Coronel, Morris, 2010)

Características de compatibilidad y requerimientos de la aplicación.

Para le edición del código se utiliza el IDE PhpStorm versión demo, el cual permite trabajar con frameworks como Symfony, Drupal, WordPress, Zend Framework, Laravel, Magento, Joomla, CakePHP. Provee soporte para lenguaje php, asistente de código, indicación de errores, soporte para HTML5, CSS y JavaScript, de igual manera íntegra gestor de tareas para control de versiones, Vagrant y Composer (JetBrains, 2016).

Para el desarrollo del prototipo se recomiendan un servicio de web hosting con mínimo las siguientes características: soporte para un dominio, integración de base de datos Mysql versión 5.0, integración de Cpanel, soporte php versión 5.3 y 10 Gigas de almacenamiento. Ya que el prototipo corresponde a una WebApps por lo cual solo requiere como características de compatibilidad navegadores de internet con soporte HTML5, soporte canvas y CSS3

Automatización de pruebas y depuración

Generalmente aplicaciones tienen un conjunto de "partes "que se integran para formar el producto final, siendo necesario las pruebas automatizadas para asegurar que el comportamiento deseado de una aplicación y su comportamiento real son consistentes durante su vida útil. Hay varios tipos de pruebas, cada una dirigida a un aspecto específico de una aplicación. Para depurar un programa el primer paso se denomina

pruebas unitarias, lo más común, mediante PHPUnit, donde se ejecutan Test Cases, clases que contienen la lógica a testear de otras clases y se ejecutan mediante comando "php unit casoxTest" (Lorna, 2011).

Marco conceptual

Marco Legal

El decreto ley 1732 de 1 de septiembre de 2014, establece la cátedra de la paz en todas las instituciones educativas del país colombiano. Entre sus principales aspectos se resalta el propósito de fortalecimiento de la cultura de paz, la creación de espacios de reflexión y diálogo sobre la cultura de la paz para mejorar la calidad de vida de los ciudadanos, todo ello en cumplimiento de los artículos 22 y 47 de la constitución nacional de Colombia.(Presidencia República de Colombia, 2014).

Marco Contextual

En el sitio web oficial de la universidad Unad, se describe (Unad, 2016):

> La Universidad Nacional Abierta y a Distancia, (UNAD) es un Proyecto Educativo que nació con el nombre de Unidad Universitaria del Sur de Bogotá, UNISUR durante el gobierno de Belisario Betancur.
>
> Surgió, mediante la Ley 52 de 1981, como un establecimiento público del orden nacional adscrito al Ministerio de Educación Nacional y transformada por el Congreso de la República mediante la Ley 396 del 5 de agosto de 1997 en la Universidad Nacional Abierta y a Distancia UNAD.
>
> Se creó con el objeto de diseñar e implementar programas académicos con la estrategia pedagógica de la educación a distancia, que fuesen pertinentes con las necesidades locales, regionales, nacionales e internacionales y acordes con los retos y las demandas de una sociedad democrática, participativa y dinámica afines con modelos científicos, sociales y culturales que contextualizan al siglo XXI.
>
> Junto al sistema funcional y operacional, figura 5, la Misión de La Universidad Nacional Abierta y a Distancia (UNAD) es contribuir a la educación

para todos a través de la modalidad abierta, a distancia y en ambientes virtuales de aprendizaje, mediante la acción pedagógica, la proyección social, el desarrollo regional y la proyección comunitaria, la inclusión, la investigación, la internacionalización y las innovaciones metodológicas y didácticas, con la utilización de las tecnologías de la información y las comunicaciones para fomentar y acompañar el aprendizaje autónomo, generador de cultura y espíritu emprendedor que, en el marco de la sociedad global y del conocimiento, propicie el desarrollo económico, social y humano sostenible de las comunidades locales, regionales y globales con calidad, eficiencia y equidad social.

La Visión de la Unad, ser una organización líder en Educación Abierta y a Distancia, reconocida a nivel nacional e internacional por la calidad innovadora y pertinencia de sus ofertas y servicios educativos y por su compromiso y aporte de su comunidad académica al desarrollo humano sostenible, de las comunidades locales y globales.

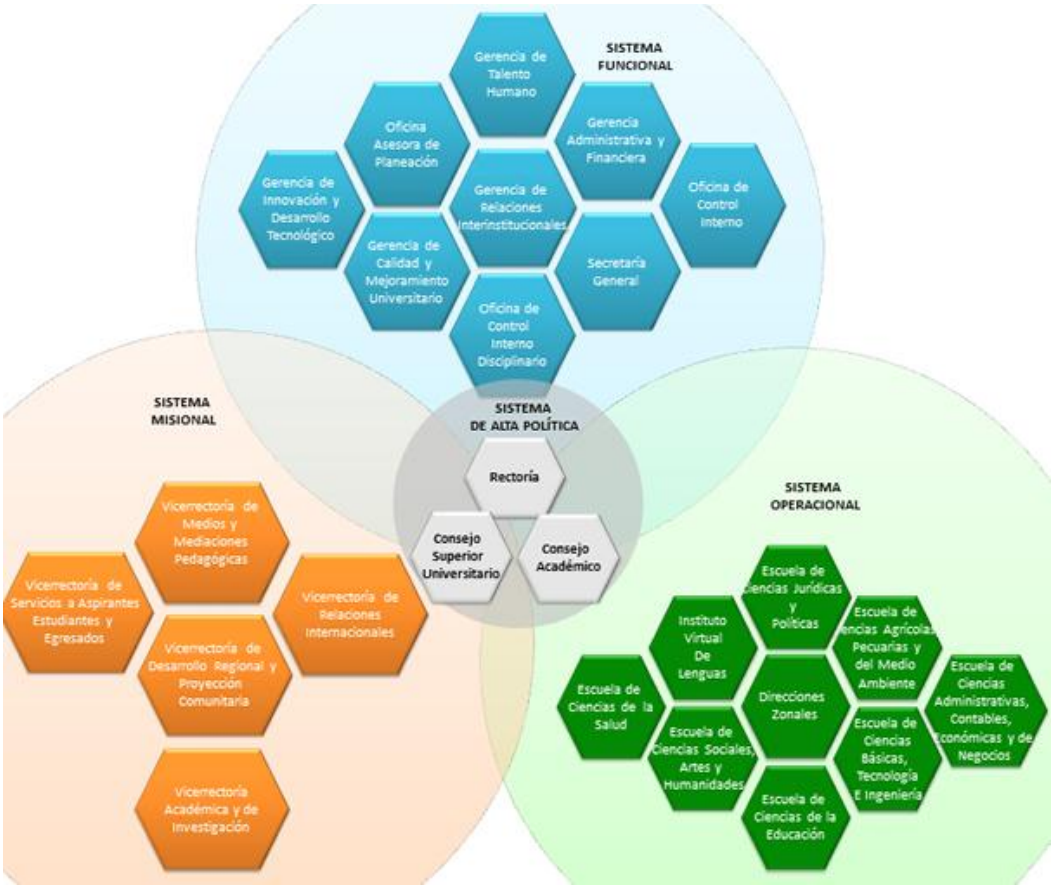

Figura 5. Organigrama Unad. Fuente: Universidad Nacional Abierta y a Distancia.

Desarrollo de aplicativos y prácticas de diseño

La calidad del diseño de una aplicación garantiza interfaces limpias, eficientes y fáciles de usar, a continuación, algunas recomendaciones (Linowski, 2016):

- Contenidos preferiblemente a una columna
- Agrupar funciones similares en la aplicación
- Hacer énfasis mediante repetición de las acciones principales o primarias
- Aplicar estilos para elementos seleccionables y "clics" claramente diferenciados
- Mostrar mensajes directos sin divagaciones y/o indecisiones
- Manejar pocos elementos en formularios, evitando la saturación
- Mostrar las opciones siempre que sea posible y no ocultarlas
- Mantener el foco de atención y no abusar del uso de enlaces
- Aplicar botones de manipulación directa, evitando menús contextuales

- Usar bordes con prudencia para evitar perder la atención del usuario
- Para elementos, aplicar convenciones: anterior, siguiente, arriba, entre otros, disminuyendo el esfuerzo de aprendizaje
- Aplicar jerarquías: indexado, fontsize, padding, espacios, que ayuden a separar los elementos importantes de aquellos que no lo son.

Metodología

Tipo de investigación.

Para el presente trabajo se asume una investigación formativa. La cual permite a cada uno de los entes académicos el desarrollo de capacidades cognoscitivas como la analítica y la solución de problemas (Restrepo, 2003).

De esta menara se realiza consulta de expertos, construcción de modelos conceptuales, ensayo de prototipos, identificación de necesidades de comunidades o grupos específicos y la aplicación de conocimientos adquiridos en áreas como programación web, estructura de datos y base de datos durante la formación como ingeniero de sistemas

Por ello, el desarrollo del prototipo se realiza mediante el movimiento ágil, donde se busca alternativas a la gestión tradicional de proyectos software. El enfoque ágil ayuda a los equipos responden a través de imprevisibilidad, cadencias de trabajo iterativos incrementales y la retroalimentación empírica. Constituye una alternativa a la rigidez de metodologías como cascada, o desarrollo secuencial tradicional. (Agile team, 2016)

Línea de investigación.

El proyecto aquí desarrollado se enmarca en la línea de investigación Gestión de sistemas–Ingeniería del software, establecida en la Universidad Nacional Abierta y a Distancia UNAD en su escuela de Ciencias Básicas Tecnología e Ingeniería ECBTI y se plantea el desarrollo de un Prototipo aplicación web para la mediación y promoción de resolución de conflictos dentro de la misma comunidad. (Unad, 2011)

Alternativa de trabajo de grado

El tratado de este documento se ajusta a un proyecto aplicado. El cual, según reglamento estudiantil de la Unad, corresponde a la actividad de transferencia investigativa que le permite al estudiante diseñar proyectos de desarrollo empresarial, tecnológico y social comunitario para contribuir de manera novedosa a la solución de problemas focalizados (Unad, 2013, p 23).

Población y muestra.

El desarrollo del prototipo dentro de la metodología ágil se fundamenta en el desarrollo de entrevista con el o los stakeholders para conocer el flujo de trabajo real y sus preocupaciones, frente a la lógica de negocio que se desea sistematizar. La metodología consiste en tomar notas de lo que dicen, especificándolo en casos de uso Notas. Estas notas conforman los escenarios y los requerimientos de la aplicación,

De manera la selección de muestra corresponde a uno o más profesionales del área de humanidades de la Universidad Nacional Abierta y a distancia, preferiblemente del área de la sicología y/o la rama de la sociología, los cuales permitirán la captura de los requerimientos del objeto del trabajo de grado planteado. Los cuales se reúnen con el analista para identificar los objetivos de la aplicación o el sistema, y para identificar los requerimientos de información que surgen a partir de estos objetivos (Kendall& Kendall, 2011).

Instrumentos.

Dentro del marco de trabajo para el desarrollo de prototipos software de manera ágil los instrumentos a utilizar comprenden:

- Diagrama casos de uso: Durante los encuentros con los stakeholders, se describen historias como una corta descripción desde el lenguaje y punto de vista del usuario, expresando rol, objetivo del requerimiento respecto a la herramienta para solución de conflictos.
- Wireframes: en correspondencia con las historias de usuario permitirán la diagramación de bosquejos de pantallas del sistema para cada una de las

funciones del prototipo. Los wireframes trasmitirán de manera instantánea y fácil las funcionalidades e ideas de diseño al cliente, mostrando cada elemento gráfico que presenta los diferentes componentes, tales como acciones para crear, editar, eliminar, menús de navegación y sus posiciones en la pantalla o interfaz del aplicativo rescon.

Etapas.

Dentro del marco de trabajo para el desarrollo de prototipos software de manera ágil se plantea como etapas:

- Recolección historias de usuario: durante las conversaciones con los stakeholders se realiza la captura de manera espontánea de las necesidades con respecto al prototipo,
- Diagramación caso de usos: Se plantea de manera gráfica los actores, casos de usos y las relaciones entre estas para la aplicación resolución de conflictos
- Análisis de requerimientos: Se elabora un esquema simulando “tarjetas”, las cuales describen aspectos principales de la Conversación, un checklist y los respectivos wireframes.
- Diseño de wireframes: Para facilitar la transmisión de ideas sobre el diseño de realiza en forma de bosquejo la posible distribución de los diferentes elementos de la interfaz de usuario, así como sus funciones.
- Codificación de requerimientos por sprint: Se procede a la codificación o “programación” de las diferentes historias de usuario, para lo cual se plantea de manera general dos sprint, el primero de ellos abordando los requerimientos de gestión de usuarios y del sistema, el segundo sprint se “codifica” los requerimientos relacionados propiamente con la herramienta de resolución de conflictos

- Socialización de sprint con stakeholders: se comunica a los stakeholders los resultados y avance alcanzados en fase de codificación y/o programación de los objetivos del sprint.
- Si aplica se repita ciclo desde la recolección de historias: En caso de requerir algún ajusto por parte del usuario o ante errores o bugs detectados se repite el ciclo anteriormente mencionado

Cronograma

Para la codificación y programación, como se mencionó anteriormente los sprint a desarrollar corresponden a dos. El primero implementara las funciones generales de gestión del sistema y de usuarios, tales como servicios de autenticación y autorización de usuarios ante la aplicación y gestión de cuentas de usuarios, el sprint 2 pretende desarrollar la gestión de la herramienta línea del tiempo, implementando las funciones de creación, edición, actualización y eliminación de conflictos, líneas de tiempo y hechos del conflicto. Para el desarrollo ágil del prototipo software, se contempla como fechas orientadoras de las actividades:

Tabla1

Cronograma

Actividad	Fecha
Sprint 1	19 de septiembre al 15 de octubre de 2016
Sprint 2	16 de octubre al 1 de noviembre de 2016

Desarrollo del proyecto

Determinación de requerimientos

Objetivo.

Cuando se presenta un conflicto, el análisis del conflicto permite determinar las orientaciones a seguir. El análisis se puede realizar de manera individual o la participación grupal de varios individuos (Mason, 2005).

La delimitación del conflicto a través de su análisis, puede ser realizado mediante herramientas para el análisis de conflictos (Fisher, 2000). La aplicación de estas herramientas generalmente se desarrolla de manera manual sobre papel o láminas de cartulina. El objetivo planteado para el análisis de requerimientos está enfocado en la ejecución de la herramienta Líneas de tiempo mediante la aplicación prototipo. El levantamiento y análisis de requerimientos se realizará mediante la herramienta Visual Paradigm.

Requerimientos técnicos.

Las historias de usuario detectadas, corresponden a las indicadas en la tabla:

Tabla2.

Historias de usuario.

Historias de usuario	
El usuario crea herramienta	El usuario puede crear nueva herramienta de resolución de conflicto
El usuario edita herramienta	El usuario puede editar una herramientas de resolución de conflicto ya existente y de la cual tiene permisos correspondientes
El usuario elimina herramienta	El usuario puede eliminar una herramienta de resolución de conflicto ya existente y de la cual tiene permisos correspondientes
El administrador crea herramienta	El administrador puede crear una herramienta de resolución de conflicto

El administrador edita herramienta	El administrador puede editar cualquier herramienta de resolución de conflicto existente en el sistema
El administrador elimina herramienta	El administrador puede eliminar cualquier herramienta de resolución de conflicto ya creada en el sistema
El administrador crea nuevo usuario	El administrador puede crear cualquier cuenta de usuario
El administrador edita cuenta de usuario	El administrador puede editar cualquier cuenta de usuario ya existente
El administrador elimina cuenta de usuario	El administrador puede eliminar cualquier cuenta de usuario contenida en la aplicación
El administrador ve términos y condiciones e información aplicación	El administrador puede ver ventana de información de términos y condiciones y ventana sobre la aplicación
El usuario ve términos y condiciones e información aplicación	El usuario puede ver ventana de información de términos y condiciones y ventana sobre la aplicación
El visitante ve términos y condiciones e información aplicación	El visitante puede ver ventana de información de términos y condiciones y ventana sobre la aplicación
El usuario ve detalles de herramienta	El usuario puede ver en detalle los datos de la herramienta de resolución de conflicto para la cual tiene permisos respectivos
El administrador ver detalles de herramienta	El administrador puede ver en detalle los datos de la herramienta de resolución de conflicto ya creada en el sistema
El administrador edita términos y condiciones e información aplicación	El visitante puede editar el contenido de las ventana de información sobre la aplicación y/o términos/condiciones

Obsérvese en la figura 6, el diagrama de casos de uso correspondientes al prototipo:

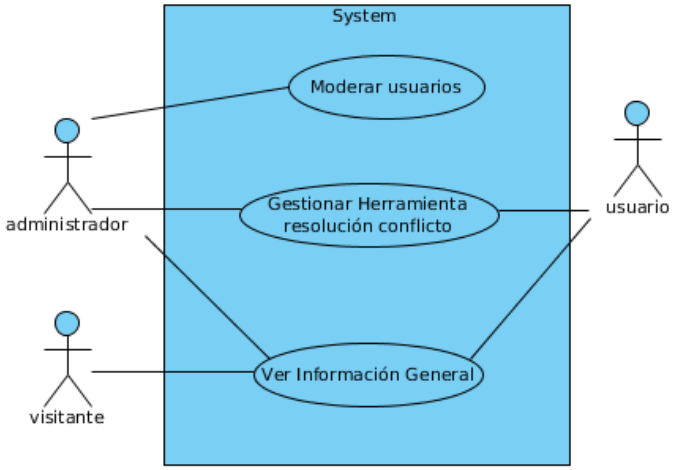

Figura 6. Diagrama casos de usos prototipo. Fuente: Autor.

Análisis de requerimientos.

El usuario crea herramienta - El administrador crea herramienta

- **Identificación**

Descripción: El usuario puede crear nueva herramienta de resolución de conflicto

Estado: Pending

- **Consideraciones**

Se muestran los hechos en relación con el tiempo. Los hechos son presentados en orden cronológico. Manejo de tiempo por selección de escala: Años, Meses, Días

- **Lista chequeo**

Se puede ingresar datos de tiempo y descripción para un evento. Se visualiza los eventos ordenados de manera cronológica. Se visualiza los eventos para cada una de las partes en conflicto hay posibilidad de selección de escala del tiempo

- **Escenario**

1. Abrir página herramienta RC
2. Seleccionar Nuevo
3. Mostrar Formulario de herramienta
4. Ingresar datos relacionados con herramienta

5. SYSTEM señala campos faltantes o incorrectos
6. if usuario selecciona nuevo ítem
   6.1. Ir a paso 3
7. elseif usuario selecciona finalizar
   7.1. Mensaje Herramienta guardada con éxito

endif

- **Wireframes:**

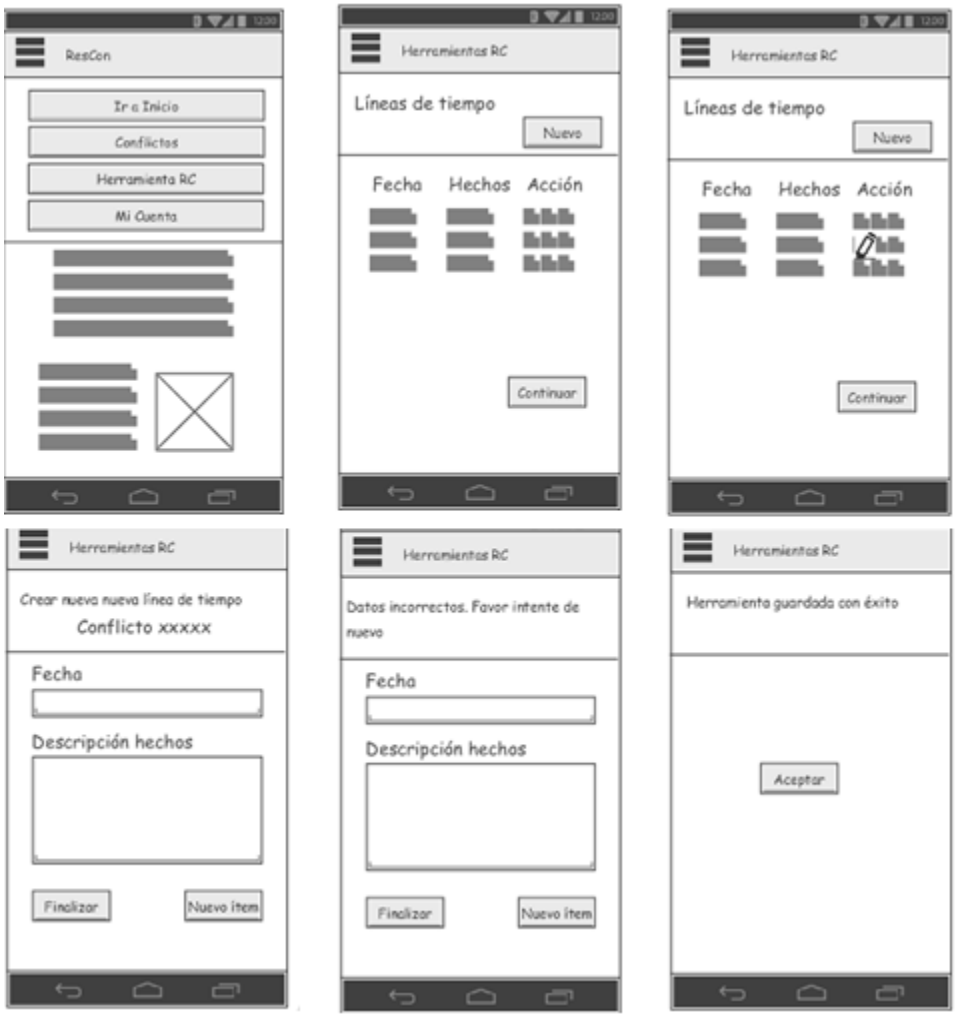

Figura 7. Editar Herramienta. Fuente: Autor

El usuario elimina herramienta - El administrador elimina herramienta

- **Identificación**

Descripción: El usuario puede eliminar una herramienta de resolución de conflicto ya existente y de la cual tiene permisos correspondientes

Estado: Pending

- **Escenario**

1. Abrir página herramienta RC
2. Seleccionar Línea de tiempo
3. Seleccionar Icono Eliminar (Columna Acción)
4. Mensaje Herramienta Eliminada con éxito

- **Wireframes:**

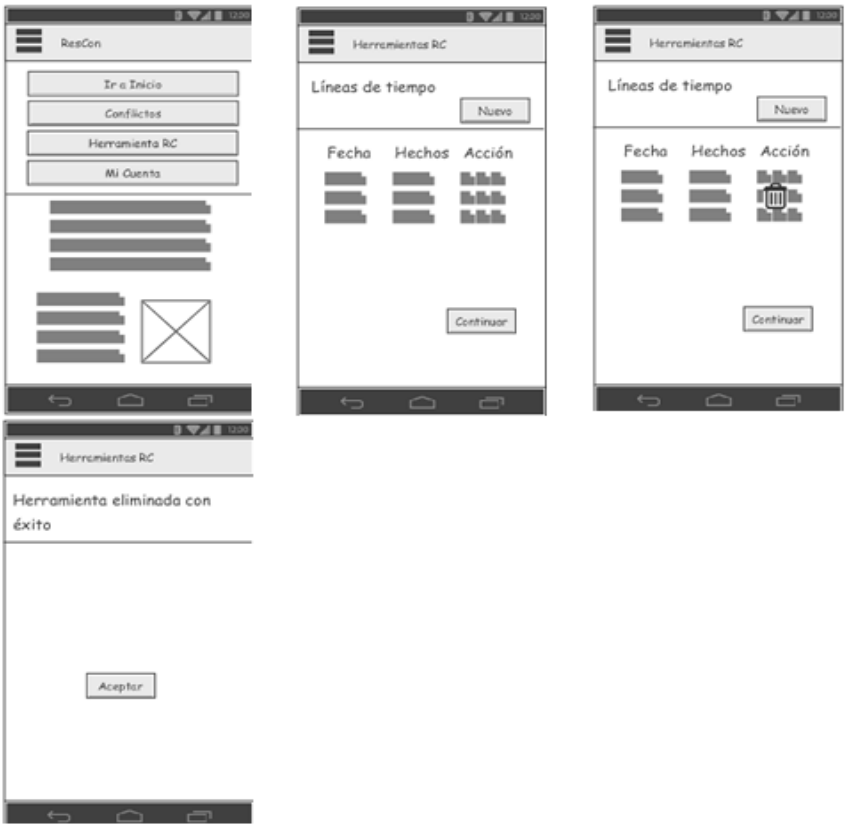

Figura 8. Eliminar Herramienta. Fuente: Autor

El administrador crea nuevo usuario

- **Identificación**

Descripción: El administrador puede crear cualquier cuenta de usuario

Estado: Pending

- **Consideraciones**

El administrador puede crear nuevas cuentas de usuario. Se desea disponer de datos básicos de identificación del usuario. Datos mínimos necesarios: Nombre de usuario, Correo electrónico, Password o contraseña

- **Lista chequeo**

Al crear nueva cuenta de usuario, aparece en el panel de administración. La base de datos de usuario contiene el nombre, password del nuevo usuario

- **Escenario**

1. Administrador selecciona Usuarios
2. SYSTEM Presenta Pagina de usuarios registrados
3. Administrador selecciona Nuevo
4. Se ingresa un nombre de usuario
5. Se ingresa cuenta de correo electrónico
6. Se ingresa password
7. Se confirma password
8. if Datos nuevo usuario correctos
   8.1. SYSTEM Nueva cuenta creada con éxito
9. elseif
   9.1. SYSTEM Señalar errores de datos

endif

- **Wireframes:**

Figura 9 Crear nuevo usuario. Fuente: Autor

El administrador edita cuenta de usuario

- **Identificación**

Descripción: El administrador puede editar cualquier cuenta de usuario ya existente

Estado: Pending

- **Escenario**

1. Administrador selecciona Usuarios
2. SYSTEM Presenta Pagina de usuarios registrados
3. Administrador selecciona Usuario
4. Administrador selecciona Icono Editar (Columna Acción)

5. Presenta formulario editar usuario

6. Administrador edita datos cuenta usuario

7. if Datos nuevo usuario correctos

   7.1. SYSTEM Mensaje Cuenta de usuario actualizada con éxito

8. elseif

   8.1. SYSTEM Señalar errores de datos

endif

- **Wireframes:**

Figura 10 Editar cuenta de usuario. Fuente: Autor

El administrador elimina cuenta de usuario

- **Identificación**

Descripción: El administrador puede eliminar cualquier cuenta de usuario contenida en la aplicación

Estado: Pending

- **Escenario**

1. Administrador selecciona Usuarios
2. SYSTEM Presenta Pagina de usuarios registrados
3. Administrador selecciona Usuario
4. Administrador selecciona Icono Eliminar (Columna Acción
5. SYSTEM Mensaje Cuenta de usuario Eliminada con éxito

- **Wireframes:**

Figura 11. Eliminar cuenta de Usuario. Fuente: Autor

El administrador y usuario ve términos y condiciones e información aplicación.

- **Identificación**

Descripción: El administrador/usuario/visitante puede ver ventana de información de términos y condiciones y ventana sobre la aplicación

Estado: Pending

- **Escenario**

1. Ingresar Página términos y condiciones
2. Presentar contenido términos y condiciones

- **Wireframes:**

Figura 12. Términos y condiciones. Fuente: Autor

El usuario ve detalles de herramienta - El administrador ve detalles de herramienta

- **Identificación**

Descripción: El usuario/administrador puede ver en detalle los datos de la herramienta de resolución de conflicto para la cual tiene permisos respectivos

Estado: Pending

- **Consideraciones**

Se muestran los detalles de la línea de tiempo. La línea de tiempo corresponde a la creada por usuario. Los hechos son presentados en orden cronológico. Los

hechos/eventos del conflicto se muestran en lados opuestos en correspondencia a cada una de las partes involucradas

- **Lista de chequeo**

  Se permite el deslizamiento de ventana para navegar la línea de tiempo

- **Escenario**

1. Abrir página herramienta RC
2. Seleccionar Línea de tiempo
3. Seleccionar Icono Ver (Columna Acción)
4. SYSTEM Presentación de datos línea de tiempo del conflicto

- **Wireframes:**

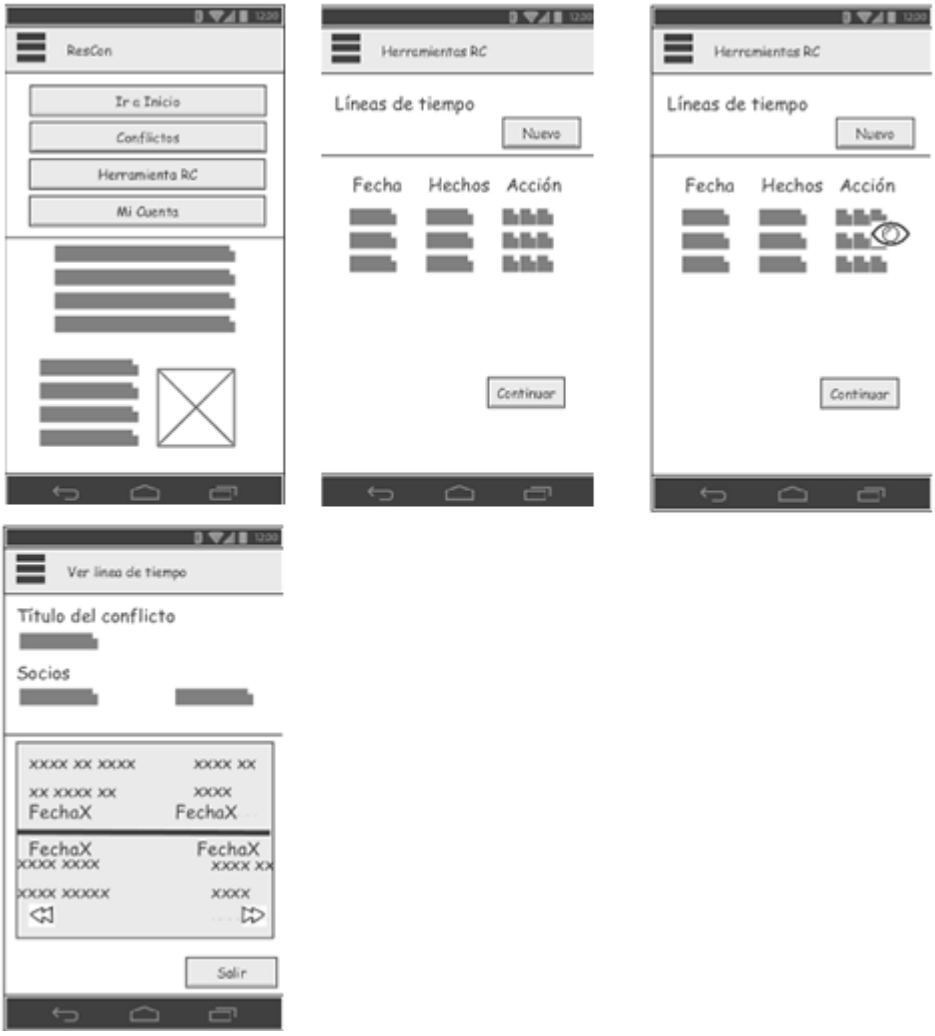

Figura 13. Detalles herramienta. Fuente: Autor

El usuario crea cuenta de usuario

- **Consideraciones**

Se desea disponer de datos básicos de identificación del usuario. Datos mínimos necesarios: nombre de usuario, correo electrónico, password o contraseña

- **Lista chequeo**

Al crear nueva cuenta, los datos de usuario son almacenados en la base de datos. La base de datos de usuario contiene el nombre, password del nuevo usuario

- **Escenario**

1. Usuario selecciona Registrarse

2. SYSTEM Presenta formulario de registro
3. Se ingresa un nombre de usuario
4. Se ingresa cuenta de correo electrónico
5. Se ingresa password
6. Se confirma password
7. if Datos nuevo usuario correctos
   7.1. SYSTEM Nueva cuenta creada con éxito
8. elseif
   8.1. SYSTEM Señalar errores de datos
endif

- **Wireframes:**

Figura 14. Usuario signup. Fuente: Autor

2. **El usuario crea nuevo conflicto**

- **Consideraciones**

Un conflicto representa una diferencia de intereses entre dos o más partes. Las partes involucradas se denominan socios. Datos de interés sobre un conflicto: Nombre o título, descripción del conflicto, Se debe garantizar la privacidad de las personas o socios

- **Lista chequeo**

Se puede ingresar datos de título, descripción y opción de visibilidad al crear nuevo conflicto. Se asocia de manera automática el usuario logeado al conflicto creado. Al crear nuevo conflicto, se almacena en la base de datos

- **Escenario**

1. Seleccionar página Conflictos
2. Seleccionar opción Nuevo
3. Presenta formulario nuevo conflicto
4. Ingresar datos nuevo conflicto
5. Clic Guardar
6. if Campos con error
   6.1. Mostrar mensaje datos incorrectos. Favor intentar de nuevo
7. else
   7.1. Mostrar mensaje: Conflicto guardado con éxito

endif

- **Wireframes:**

Figura 15. Crear nuevo conflicto. Fuente: Autor

Modelo de datos

Las entidades identificadas para el prototipo propuesto son: usuario, conflicto, línea de tiempo y hecho del conflicto, igualmente y solo con carácter ilustrativo se menciona entidades como mapa del conflicto, cebolla, árbol y pirámide de conflicto, que corresponden a otras herramientas de resolución de conflictos no abordadas.

Las asociaciones detectadas entre las entidades son:

1. Un Usuario puede tener uno o muchos conflictos.

2. Una herramienta de conflicto pertenece solo a un conflicto
3. Un Conflicto puede tener muchas líneas de tiempo
4. Los hechos pertenecen solo a una línea de tiempo
5. Las líneas de tiempo deben pertenecer solo a un conflicto

Se observa en la figura15, el diagrama relacional:

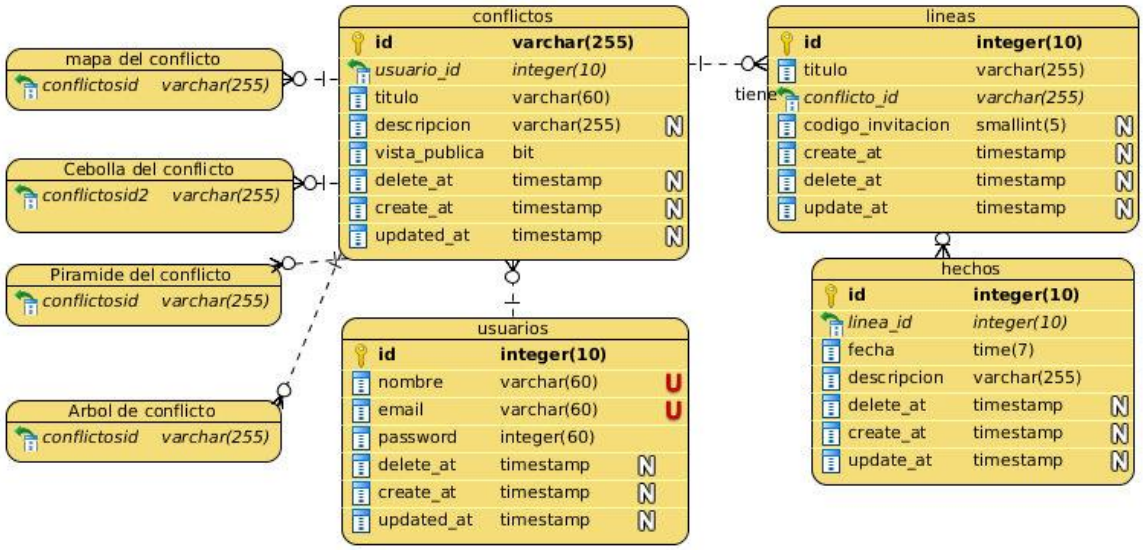

Figura 16. Tabla relacional prototipo. Fuente: Autor

Las tablas relacionales con su respectiva descripción de atributos por entidad:

Tabla3.

Tabla relacional prototipo

Nombre	Tipo	Longitud	Descripción
id	Integer	0	Identificador único de la entidad
nombre	Varchar	0	Nombre de usuario
email	varchar	0	Correo electrónico del usuario
password	integer	0	Contraseña del usuario
delete_at	timestamp		Marca de tiempo de eliminación de

			registro
create_at	timestamp		Marca de tiempo de creación de registro
updated_at	timestamp		Marca de tiempo de actualización de registro
id	integer	0	Identificador único de la entidad
linea_id	integer	0	Identificado de línea de tiempo
fecha	time		Fecha de ocurrencia del hecho
descripcion	varchar	55	Descripción del hecho de conflicto
delete_at	timestamp		Marca de tiempo de eliminación de registro
create_at	timestamp		Marca de tiempo de creación de registro
update_at	timestamp		Marca de tiempo de actualización de registro
id	integer	10	Identificador único de la entidad
titulo	varchar	55	Título para la línea de tiempo
conflicto_id	varchar	55	Identificador de conflicto
codigo_invitacion	smallint		Código de asociación de hechos/conflicto
create_at	timestamp		Marca de tiempo de creación de registro
delete_at	timestamp		Marca de tiempo de eliminación de registro
update_at	timestamp		Marca de tiempo de actualización de registro
id	varchar	55	Identificador único de la entidad
usuario_id	integer	0	Identificador del usuario involucrado en conflicto

titulo	varchar	0	Título del conflicto
descripcion	varchar	55	Descripción naturaleza del conflicto
vista_publica	bit		Opción visibilidad al público
delete_at	timestamp		Marca de tiempo de eliminación de registro
create_at	timestamp		Marca de tiempo de creación de registro
updated_at	timestamp		Marca de tiempo de actualización de registro

Desarrollo del prototipo

Configuración del entorno

El desarrollo de la aplicación se realiza sobre máquina virtual alojada en sistema operativo Linux y para ellos se realiza la configuración de las herramientas vagrant-Homestead, los cuales incluyen Ubuntu 14.04, Git, PHp 7, Mysql, MariaDB, Sqlite3, composer entre otros. De manera general el procedimiento seguir es(Laravel, 2015):

1. Descarga e instalación de VirtualBox 5.x. (https://www.virtualbox.org/wiki/Downloads)
2. Descarga e instalación de vagrant (https://www.vagrantup.com/downloads.html)
3. Descarga e instalación de Homestead, Comando (ventana de comandos):
    a. vagrant box addlaravel/homestead
4. Declaración de proveedor y carpetas compartidas entre host y máquina virtual dentro del archivo ~/.homestead/Homestead.yaml :
    a. provider: virtualbox
    b. folders
    c. map: ~/Code
        i. to: /micarpeta/proyecto/enhost
5. En el equipo Host en archivo /etc/hosts, agregamos dominio:
    a. 192.168.10.10 midominio.dev

6. La conexión a la base de datos por defecto 127.0.0.1 and port 33060 (MySQL), usuario y contraseña: homestead / secret
7. Finalmente, mediante comandos iniciamos la máquina y nos conectamos mediante protocolo ssh:
    a. vagrant up
    b. vagrant ssh
8. Sobre la máquina virtual, creamos un nuevo proyecto, mediante comando: composer create-project laravel/laravel rescon "5.1.*"

Configuración librerías externas

AdminLTE constituye una plantilla para webapps de carácter open source y desarrollada en HTML y con estilos CSS Bootstrap. Facilita la configuración de tableros de administrador y paneles de control demás componentes comunes para la parte administrativa y de interface de un proyecto. A continuación los pasos para su instalación y configuración(InfyomLabs, 2015):

1. Agregamos al archivo composer.json:
    a. "require": {
    b. "infyomlabs/laravel-generator": "5.3.x-dev",
    c. "laravelcollective/html": "dev-master",
    d. "infyomlabs/core-templates": "5.3.x-dev"
    e. }
2. Registramos como proveedor de servicio o librería (en archivo config/app.php):
    a. Collective\Html\HtmlServiceProvider::class,
    b. Laracasts\Flash\FlashServiceProvider::class,
    c. Prettus\Repository\Providers\RepositoryServiceProvider::class,
    d. \InfyOm\Generator\InfyOmGeneratorServiceProvider::class,
    e. \InfyOm\CoreTemplates\CoreTemplatesServiceProvider::class,
3. Para el uso realizamos el registro de sus alias (en config/app.php):
    a. 'Form' =>Collective\Html\FormFacade::class,
    b. 'Html' =>Collective\Html\HtmlFacade::class,
    c. 'Flash' =>Laracasts\Flash\Flash::class,

Configuración control de Versiones

Para el prototipo se usa la plataforma gitlab.com. Una vez creada una cuenta de usuario, se realiza desde el icono de nuevo proyecto, el registro de datos como título, descripción del proyecto y su visibilidad, como lo ilustra la siguiente figura:

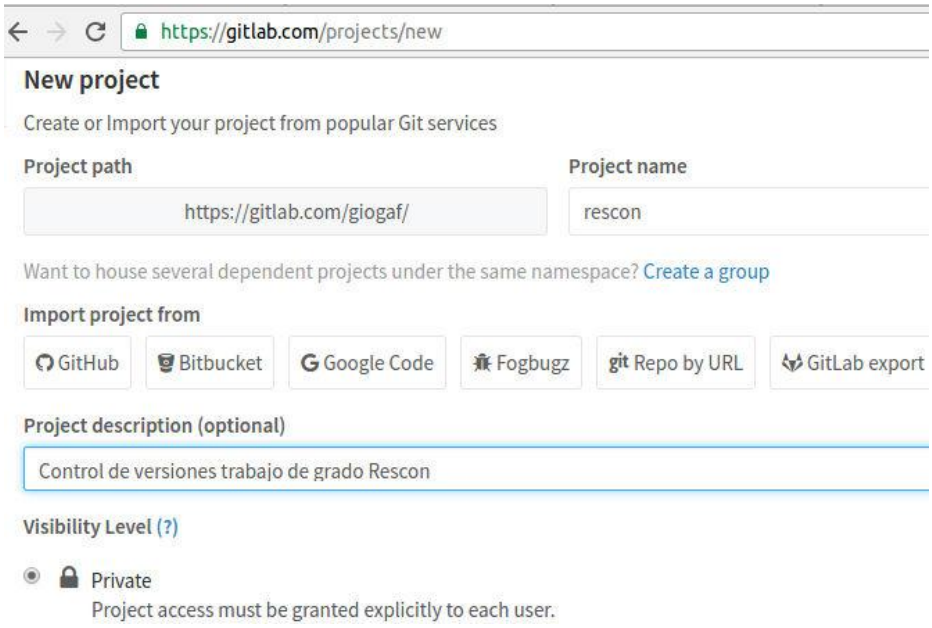

Figura 17. Control de versiones en gitlab. Fuente: Autor

Al crear el proyecto, como se muestra en la figura 16, el sitio le indicara los pasos finales:

1. Desde la máquina virtual configuramos acceso a nuestra cuenta en gitlab:
    a. gitconfig --global user.name "giogaf"
    b. gitconfig --global user.email "giogaf@gmail.com"
2. Para inicializar control de versiones con git y agregar el repositorio remoto, finalmente ejecutamos:
    a. cd rescon
    b. git init
    c. git remote add origin git@gitlab.com:giogaf/rescon.git
    d. git add.
    e. git commit –m “Esta es la versión inicial-base del proyecto”

f. git push -u origin master

Sprint 1/2

Durante la ejecución de esta etapa se implementará las historias de usuario relacionadas con la administración de usuarios y la gestión de cuentas del prototipo. Acciones a desarrollar:

Se registra los campos relacionados con la base de datos (archivo .env)

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_DATABASE=homestead
DB_USERNAME=homestead
DB_PASSWORD=secret
```

Se define de rutas para la aplicación (rescon/http/routes.php):

```
<?php
Route::get('/', function() {
        return redirect('/welcome');
    });
Route::resource('usuarios', 'usuarioController');
Route::resource('cuentausuario', 'cuentausuarioController',
                    ['only'=>['show','edit','update']]
);
Route::get('/welcome', 'HomeController@welcome')->name('welcome');
Route::get('/home', 'HomeController@home')->name('home');
Route::get('login', 'Auth\AuthController@getLogin')->name('login');
Route::post('login', 'Auth\AuthController@postLogin');
Route::get('logout', 'Auth\AuthController@getLogout')->name('logout');
Route::get('register', 'Auth\AuthController@getRegister')->name('registrarse');
Route::post('register', 'Auth\AuthController@postRegister');
Route::get('condiciones','HomeController@condiciones')->name('condiciones');
```

Ejecución de migraciones, la instalación por defecto viene configurada para realizar la migración de la tabla users, en la carpeta del proyecto, sobre la máquina virtual, se ejecuta:

```
vagrant@homestead:~/Code/rescon$ art  migrate
```

Creación de Scaffold, para la generación de operaciones de lectura, edición, actualización y borrado de usuarios, indicando el nombre del modelo, nombre de la tabla en la base de datos, se ejecuta el comando

```
vagrant@homestead:~/Code/rescon$ art infyom:scaffold usuarios --fromTable --tableName=users
```

Se crean los siguientes recursos:

```
Modelo creado:
usuario.php
Repositorio:
usuarioRepository.php
Manejo peticiones Request :
CreateusuarioRequest.php
Creación de Controller:
usuarioController.php
Vistas Generadas:
table.blade.php
index.blade.php
field.blade.php
create.blade.php
edit.blade.php
show_fields.blade.php
show.blade.php
```

Se realizan los ajustes en usuariocontroller.php, con el siguiente resultado:

```php
<?php
class usuarioController extends AppBaseController
{
```

```
    *
    * @paramRequest $request
    * @returnResponse
    */
public function index(Request $request)
    {
        $this->usuarioRepository->pushCriteria(new RequestCriteria($request));
        $usuarios = $this->usuarioRepository->all();
        return view('usuarios.index')
            ->with('usuarios', $usuarios);
}
public function create()
    {
        return view('usuarios.create');
}
public function store(CreateusuarioRequest $request)
    {
        $input = $request->all();
        $input['password']= bcrypt($input['password']);
        $usuario = $this->usuarioRepository->create($input);
        Flash::success('Usuario ha sido guardado.');
        return redirect(route('usuarios.index'));
}

public function show($id)
    {
        $usuario = $this->usuarioRepository->findWithoutFail($id);
        if(empty($usuario)) {
            Flash::error('Usuario no hallado');
        return redirect(route('usuarios.index'));
```

```
}
return view('usuarios.show')->with('usuario', $usuario);
}

public function edit($id)
    {
        $usuario = $this->usuarioRepository->findWithoutFail($id);
        if(empty($usuario)) {
            Flash::error('Usuario no hallado');
        return redirect(route('usuarios.index'));
        }
        return view('usuarios.edit')->with('usuario', $usuario);
}

public function update($id, UpdateusuarioRequest $request)
    {
        $usuario = $this->usuarioRepository->findWithoutFail($id);
        if(empty($usuario)) {
        Flash::error('Usuario ho hallado');
        return redirect(route('usuarios.index'));
        }
        $pass=$request->input('password');
        $usuario= $request->all();
        $usuario['password']= bcrypt($pass);
        $usuario = $this->usuarioRepository->update($usuario, $id);
        Flash::success('Usuario ha sido actualizado.');
        return redirect(route('usuarios.index'));
}
public function destroy($id)
    {
```

```
        if($id != 1)
        $usuario = $this->usuarioRepository->findWithoutFail($id);
        else{
        $usuario=null;
        Flash::error('Usuario Administrador no puede ser eliminado');
        return redirect(route('usuarios.index'));
        }
        if(empty($usuario)) {
        Flash::error('Usuario no hallado');
        return redirect(route('usuarios.index'));
        }
        $this->usuarioRepository->delete($id);
        Flash::success('Usuario ha sido borrado.');
        return redirect(route('usuarios.index'));
        }
}
```

Edición de vistas, incorporamos al menú:

```
<li class=""><a href="{{route('home') }}"><i class="fa fa-home"></i>Ir a
Inicio</a></li>
@if (Auth::user()->id==1)
<li class=""><a href="{{route('usuarios.index')  }}"><i class="fa fa-
users"></i>Usuarios</a></li>
@endif

<li class=""><a href="{{route('cuentausuario.show',Auth::user()->id)}}"><i class="fa
fa-user"></i>Mi cuenta</a></li>

<li class=""><a href="{{route('logout')  }}"><i class="fa fa-sign-out
"></i>Salir</a></li>
```

Las figuras 17, 18,19, detallan algunas imágenes de los resultados de esta fase:

Figura 18. Portada prototipo. Fuente: Autor

Figura 19. Gestión de usuario en prototipo. Fuente: Autor

Figura 20. Cuenta de usuario en prototipo. Fuente: Autor

Sprint 2/2

Durante la ejecución de esta etapa se implementa las historias de usuario relacionadas con la gestión de conflictos y herramientas de conflicto.

Definición de rutas:

```
Route::resource('conflictos', 'conflictoController',
[
      'names'=>[
      'index'=>'conflictos.index',
      'create'=>'conflictos.create',
      'destroy'=>'conflictos.destroy',
      'show'=>'conflictos.show',
      'update'=>'conflictos.update',
      'edit'=>'conflictos.edit',
      'store'=>'conflictos.store',
      ]
   ]);
Route::resource('lineas', 'lineaController',
[
```

```
        'names'=>[
        'index'=>'lineas.index',
        'create'=>'lineas.create',
        'destroy'=>'lineas.destroy',
        'show'=>'lineas.show',
        'update'=>'lineas.update',
        'edit'=>'lineas.edit',
        'store'=>'lineas.store',
        ]
        ]);
Route::resource('lineas.hechos', 'hechoController',
[
        'names'=>[
        'index'=>'lineas.hechos.index',
        'create'=>'lineas.hechos.create',
        'destroy'=>'lineas.hechos.destroy',
        'show'=>'lineas.hechos.show',
        'update'=>'lineas.hechos.update',
        'edit'=>'lineas.hechos.edit',
        'store'=>'lineas.hechos.store',
        ]
]);
```

Ejecución de migraciones, se editan las migraciones para los modelos conflicto, líneas de tiempo, hechos del conflicto:

```
Public function up()
{
Schema::create('conflictos', function(Blueprint $table) {
        $table->increments('id');
        $table->integer('usuario_id')->unsigned();
        $table->foreign('usuario_id')->references('id')->on('users');
```

```
        $table->string('titulo_conflicto',60);
        $table->text('descripcion_conflicto');
        $table->boolean('vista_publica')->default(false);
        $table->softDeletes();
        $table->timestamps();
});
}
```

Para la entidad líneas, se tiene:

```
Public function up()
{
Schema::create('lineas', function(Blueprint $table) {
        $table->increments('id');
        $table->string('titulo',60);
        $table->integer('conflicto_id')->unsigned();
        $table->foreign('conflicto_id')->references('id')->on('conflictos');
        $table->string('codigo_invitacion',10)->nullable();
        $table->softDeletes();
        $table->timestamps();
});
}
```

La entidad hechos, se describe, así:

```
Public function up()
{
Schema::create('hechos', function(Blueprint $table) {
        $table->increments('id');
        $table->date('fecha');
        $table->text('descripcion');
        $table->integer('linea_id')->unsigned();
        $table->foreign('linea_id')->references('id')->on('lineas');
        $table->softDeletes();
```

```
        $table->timestamps();
});
}
```

En seguida, se ejecuta comando:

```
vagrant@homestead:~/Code/rescon$ art  migrate
```

Desde phpMyAdmin se puede observar el resultado (figura 20):

Figura 21.Migraciones prototipo. Fuente: Autor

Creación de Scaffold, para la generación de operaciones de lectura, edición, actualización y borrado de conflictos y herramientas, se ejecuta el comando:

```
vagrant@homestead:~/Code/rescon$ art infyom:scaffold conflicto --fromTable --tableName=conflictos
vagrant@homestead:~/Code/rescon$ art infyom:scaffold hecho --fromTable --tableName=hechos
vagrant@homestead:~/Code/rescon$ art infyom:scaffold linea --fromTable --tableName=lineas
```

Editamos conflicto controller, para obtener:

```
<?php
namespace App\Http\Controllers;
class conflictoController extends AppBaseController
{
public function index(Request $request)
```

```
    {
        $nombre_usuario=Auth::user()->name;
        $usuario_model=User::where('name',$nombre_usuario)->first();
        $conflictos = conflicto::where('usuario_id', $usuario_model->id)->get();
        return view('conflictos.index')
            ->with('conflictos', $conflictos);
}

public function create()
    {
        returnview('conflictos.create');
}

public function store(CreateconflictoRequest $request)
    {
        $input = $request->all();
        $nombre_usuario=Auth::user()->name;
        $usuario=User::where('name',$nombre_usuario)->first();
        $input['usuario_id']=$usuario->id;
        $conflicto = $this->conflictoRepository->create($input);
        Flash::success('Conflicto guardado.');
        return redirect(route('conflictos.index'));
}

public function show($id)
    {
        $conflicto = $this->conflictoRepository->findWithoutFail($id);
        if(empty($conflicto)) {
            Flash::error('Conflicto no hallado');
        return redirect(route('conflictos.index'));
```

```
        }
        returnview('conflictos.show')->with('conflicto', $conflicto);
    }

    public function edit($id)
    {
        $conflicto = $this->conflictoRepository->findWithoutFail($id);
        if(empty($conflicto)) {
            Flash::error('Conflicto no hallado');
        return redirect(route('conflictos.index'));
        }
        return view('conflictos.edit')->with('conflicto', $conflicto);
    }

    public function update($id, UpdateconflictoRequest $request)
    {
        $conflicto = $this->conflictoRepository->findWithoutFail($id);
        if(empty($conflicto)) {
            Flash::error('Conflicto no hallado);
        return redirect(route('conflictos.index'));
        }
        $conflicto = $this->conflictoRepository->update($request->all(), $id);
        Flash::success('Conflicto updatedsuccessfully.');
        return redirect(route('conflictos.index'));
    }

    public function destroy($id)
    {
        $conflicto = $this->conflictoRepository->findWithoutFail($id);
        if(empty($conflicto)) {
```

```
            Flash::error('Conflicto no encontrado');
        return redirect(route('conflictos.index'));
        }
        $this->conflictoRepository->delete($id);
        Flash::success('Conflicto creado.');
        return redirect(route('conflictos.index'));
        }
}
```

Editamos líneacontroller, para obtener:

```
<?php
class línea Controller extends AppBaseController
{
public function index(Request $request)
    {
        $this->lineaRepository->pushCriteria(new RequestCriteria($request));
        $lineas= DB::table('lineas')
            ->leftJoin('conflictos','conflictos.id','=','lineas.conflicto_id')
            ->leftJoin('users','users.id','=','conflictos.usuario_id')
            ->where('conflictos.usuario_id',Auth::user()->id)
            ->get(['lineas.id','titulo','titulo_conflicto','codigo_invitacion']);
        return view('lineas.index')
            ->with('lineas', $lineas);
}

public function create()
    {
        $user_id= Auth::user()->id;
        $conflictos= conflicto::where('usuario_id',$user_id)->get(['id','titulo_conflicto']);
        for each($conflictos as $c){
```

```
        $c['titulo_conflicto']= str_limit($c['titulo_conflicto'], 20);
        }
        return view('lineas.create',['conflictos'=>$conflictos->pluck('titulo_conflicto','id')
->toArray()]);
}

public function store(CreatelineaRequest $request)
    {
        $input = $request->all();
        if($input['codigo_invitacion']=="")
        $input['codigo_invitacion'] = str_random(4);
        dd($input);
        Flash::success('Registro Linea de tiempo guardada..');
        return redirect(route('lineas.index'));
}

public function show($id)
    {
        $linea= $this->lineaRepository->findWithoutFail($id);
        if(empty($linea)) {
            Flash::error('Linea de tiempo no hallada');
        return redirect(route('lineas.index'));
        }
        return view('lineas.show')->with('linea', $linea);
}

public function edit($id)
    {
        $linea= $this->lineaRepository->findWithoutFail($id);
        $user_id= Auth::user()->id;
```

```
        $coleccion= conflicto::where('usuario_id',$user_id)->get(['id','titulo_conflicto']);
        for each($coleccionas $c){
        $c['titulo_conflicto']= str_limit($c['titulo_conflicto'], 20);
        }
        $filtro = $coleccion->pluck('titulo_conflicto','id')->toArray();
        if(empty($linea)) {
            Flash::error(Línea de tiempo no hallada);
        returnredirect(route('lineas.index'));
        }
        returnview('lineas.edit',['linea'=>$linea,'conflictos'=>$filtro]);
}

public function update($id, UpdatelineaRequest $request)
    {
        $linea= $this->lineaRepository->findWithoutFail($id);
        if(empty($linea)) {
            Flash::error(línea de tiempo no hallada);
        returnredirect(route('lineas.index'));
        }
        $linea= $this->lineaRepository->update($request->all(), $id);
        Flash::success('Linea de tiempo creada.');
        return redirect(route('lineas.index'));
}

public function destroy($id)
    {
$linea= $this->lineaRepository->findWithoutFail($id);
        if(empty($linea)) {
            Flash::error('Línea de tiempo no hallada');
        return redirect(route('lineas.index'));
```

```
        }
        $this->lineaRepository->delete($id);
        Flash::success('Linea de tiempo eliminada.');
        return redirect(route('lineas.index'));
        }
}
```

Se edita hecho controller, para obtener:

```
<?php
namespace App\Http\Controllers;
{
public function index(Request $request,$lineaId)
    {
        $hechos= DB::table('hechos')
            ->leftJoin('lineas','hechos.linea_id','=','lineas.id')
            ->leftJoin('conflictos','conflictos.id','=','lineas.conflicto_id')
            ->leftJoin('users','users.id','=','conflictos.usuario_id')
            ->where('hechos.linea_id',$lineaId)
            ->get(['hechos.id','hechos.fecha','hechos.descripcion']);
        $f=array_pluck($hechos,'fecha');
        $d=array_pluck($hechos,'descripcion');
        $t=array(collect($hechos)->count(),5);
        if($request->getContentType()=='json')
                return array_collapse([$t,$f,$d]);
        return view('hechos.index',['hechos'=>$hechos,'linea_id'=>$lineaId]);
}

public function create($lineaId)
    {
        $hecho= new hecho();
        return view('hechos.create',['hecho'=>$hecho,'linea_id'=>$lineaId]);
```

```
}

public function store(CreatehechoRequest $request,$lineaId)
    {
        $input = $request->all();
        $hecho = $this->hechoRepository->create($input);
        Flash::success('Hecho guardado.');
        returnredirect(route('lineas.hechos.index',['linea_id'=>$lineaId]));
}

public function show($lineaId,$hechoId)
    {
$hecho = $this->hechoRepository->findWithoutFail($hechoId);
        if(empty($hecho)) {
            Flash::error('Hecho de conflicto no hallado);
        return redirect(route('hechos.index'));
        }
        return view('hechos.show',['linea_id'=>$lineaId,'hecho'=>$hecho ]);
}

public function edit($lineaId,$hechoId)
    {
        $hecho = $this->hechoRepository->findWithoutFail($hechoId);
        if(empty($hecho)) {
            Flash::error('Hecho no hallado);
        return redirect(route('lineas.hechos.index',[$lineaId]));
        }
        return view('hechos.edit',['hecho'=>$hecho, 'linea_id'=>$lineaId]);
}
```

```
public function update($lineaId,$hechoId,UpdatehechoRequest $request)
    {
        $hecho = $this->hechoRepository->findWithoutFail($hechoId);
        if(empty($hecho)) {
            Flash::error('Hecho no hallado);
        return redirect(route('hechos.index'));
        }
        $hecho = $this->hechoRepository->update($request->all(), $hechoId);
        Flash::success('Hecho guardado.');
        return redirect(route('lineas.hechos.index',$lineaId));
}

public function destroy($id)
    {
        $hecho = $this->hechoRepository->findWithoutFail($id);
        if(empty($hecho)) {
            Flash::error('Hecho no hallada);
        return redirect(route('hechos.index'));
        }
        $this->hechoRepository->delete($id);
        Flash::success('Hecho eliminado.');
        return redirect(route('hechos.index'));
}
```

Creación de vistas, incorporamos al menú:

```
<li class=""><a href="{{route('conflictos.index')  }}"><i class="fa fa-exclamation-
triangle"></i>Conflictos</a></li>
<li class="treeview">
<a href="#">
<i class="fa fa-tasks"></i><span>Herramientas RC</span>
<spanclass="pull-right-container">
```

```
<i class="fa fa-angle-leftpull-right"></i>
</span>
</a>
<ul class="treeview-menu" style="display: none;">
<li class=""><a href="{{route('lineas.index') }}"><i class="fa fa-circle-o text-
yellow"></i>Líneas de tiempo</a></li>
<li class=""><a href="#"><i class="fa fa-circle-o"></i>Mapas de conflicto</a></li>
<li class=""><a href="#"><i class="fa fa-circle-o"></i>Árbol de conflicto</a></li>
<li class=""><a href="#"><i class="fa fa-circle-o"></i>Pirámide Conflicto</a></li>
<li class=""><a href="#"><i class="fa fa-circle-o"></i>Cebolla</a></li>
</ul>
</li>
```

Los resultados de esta etapa, se observan en las figuras 21, 22, 23,24 y 25:

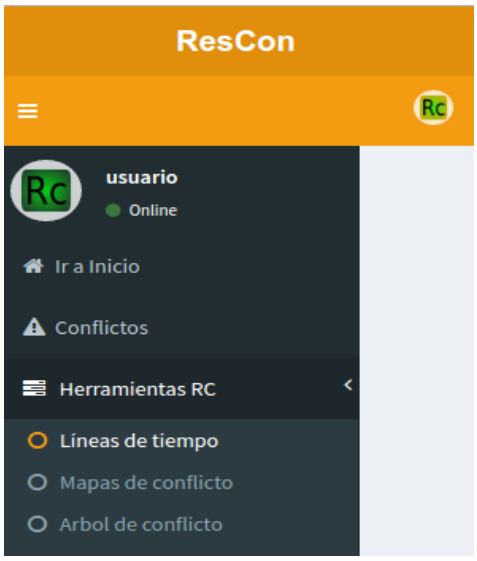

Figura 22. Menú conflictos y herramientas en prototipo. Fuente: Autor

Figura 23. Gestión de conflictos en prototipo. Fuente: Autor

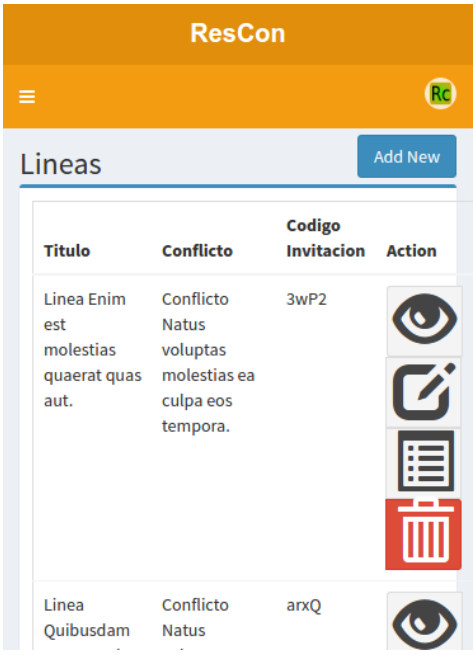

Figura 24. Gestión lineas de tiempo. Fuente: Autor

Figura 25. Gestión hechos del conflicto en prototipo. Fuente: Autor

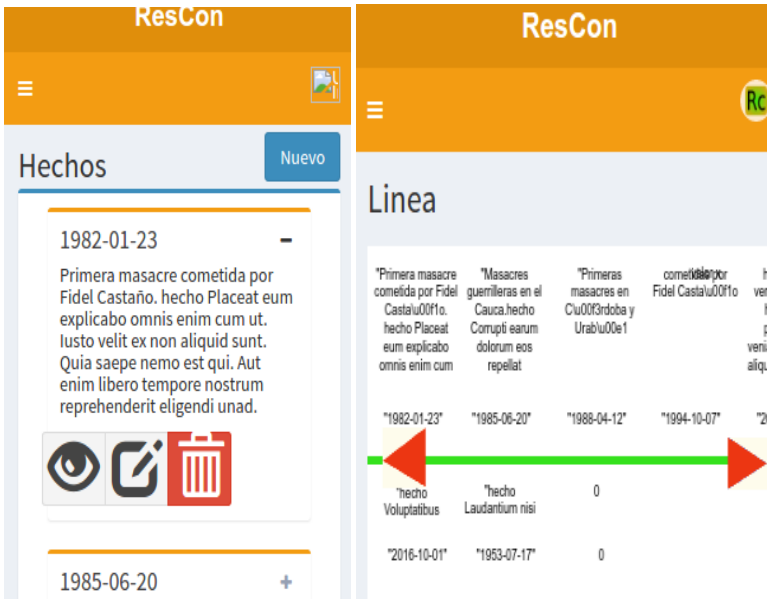

Figura 26. Hechos y visor línea de tiempo interactivo. Fuente: Autor

Ejecución de pruebas

Laravel viene configurado por defecto con el andamiaje necesario para la realización de testing con PHPUnit. De igual manera el framework trae métodos helper que facilitan testear las aplicaciones. Enseguida se describe el proceso para el llenado de la base de datos con datos de prueba mediante factories, para continuar con ejemplos de test de las entidades del prototipo. Las fábricas o factories, permiten la manipulación de datos ficticios o semillas para probar la base de datos, las factories a su vez puede usar el paquete Faker, el cual permite crear datos de manera automatizada como nombres, párrafos, direcciones, entre otros (Styde, 2015).

Para la creación de semillas, se realiza la declaración en el archivo database/factories/ModelFactory.php :

```
<?php
$factory->define(App\User::class, function(Faker\Generator $faker) {
    return[
    'name' =>$faker->name,
```

```
        'email' =>$faker->safeEmail,
        'password' =>bcrypt(str_random(10)),
        'remember_token' =>str_random(10),
        ];
});

$factory->define(App\Models\conflicto::class,function(Faker\Generator$f){
        $usuario = new App\User;
        $num_usuarios=$usuario->count();
        return[
        'usuario_id'=> rand(1,$num_usuarios),
        'titulo_conflicto'=>"Conflicto ".$f->sentence(),
        'descripcion_conflicto'=>$f->paragraph(),
        'vista_publica'=>rand(0,1),
        ];
});

$factory->define(App\Models\linea::class,function(Faker\Generator$f){
        $conflicto = new App\Models\conflicto;
        $num_conflictos=$conflicto->count()/3;
        return[
        'conflicto_id'=> rand(1,$num_conflictos),
        'titulo'=>"Linea ".$f->sentence(),
        'codigo_invitacion'=>str_random(4),
        ];
});

$factory->define(App\Models\hecho::class,function(Faker\Generator$f){
        $linea= new App\Models\Linea;
        $num_lineas=$linea->count();
```

```
        return[
        'fecha'=>$f->date(),
        'linea_id'=> rand(1,$num_lineas),
        'descripcion'=>"hecho ".$f->paragraph(4),
        ];
});
```

De esta manera se procede a la creación del semillero o seeders:

```
vagrant@homestead:~/Code/rescon$ art make:seederResconSeeder
```

Creando el archivo database/sedes/ResconSeeder.php, en el cual se declara las semillas para llenar la base de datos, declarando el modelo y el número de registros a ser creados (create ()):

```
<?php
class ResconSeeder extends Seeder
{

public function run()
    {
        User::create(['name'=>'rescon
(administrador)','email'=>'rescon@rescon.com','password'=>Hash::make('rescon')]);
User::create(['name'=>'usuario','email'=>'usuario@rescon.com','password'=>Hash::make
('usuario')]);
        factory(App\User::class,2)->create();
        factory(App\Models\conflicto::class,10)->create();
        factory(App\Models\linea::class,30)->create();
        factory(App\Models\hecho::class,300)->create();
        }
}
```

De otra parte, creamos el archivo para ejecución de test (se localizará en rescon/tests):

```
vagrant@homestead:~/Code/rescon$ art make:testresconTest
```

Editamos resconTest.php con los casos de prueba unitaria, para tener:

```php
<?php
class resconTest extends TestCase
{
use DatabaseTransactions;
        /**
    * A basicfunctional test example.
    *
    * @return void
    */
public function testWelcomePageExample()
   {
      $this->visit('/')
          ->seePageIs('/welcome');
}

public function testIngresoUsuarioRegistrado()
          {
      $this->visit('/')
          ->click('Ingresar')
          ->seePageIs('/login');
      $this->visit('/login')
          ->type('rescon@rescon.com','email')
          ->type('rescon','password')
          ->press('Aceptar')
          ->seePageIs('/home');
}
public function testIngresoUsuarioNoRegistrado()
          {
      $this->visit('/')
```

```
            ->click('Ingresar')
            ->seePageIs('/login');
        $this->visit('/login')
            ->type('rescon@rescon.com','email')
            ->type('rescon1','password')
            ->press('Aceptar')
            ->seePageIs('/login');
}

public function testRegistroUsuarioNuevo()
    {
        $this->visit('/')
            ->click('Registrarse')
            ->seePageIs('/register')
            ->type('usuariotest','name')
            ->type('usuariotest@rescon.com','email')
            ->type('usuariotest','password')
            ->type('usuariotest','password_confirmation')
            ->press('Registrarme')
            ->seePageIs('/home');
        }
}
```

Ejecutamos phpunit sobre el archivo resconTest.php

```
vagrant@homestead:~/Code/rescon$  phpunittests/resconTest.php
```

Como se observa en el archivo se presentan cuatro casos de test los cuales han sido verificados de manera exitosa, tal como se muestra enseguida:

```
vagrant@homestead:~/Code/rescon$ phpunit tests/resconTest.php
PHPUnit 4.8.27 by Sebastian Bergmann and contributors.

....                                                                4 / 4 (100%)

Time: 1 minute, Memory: 14.00MB
```

Figura 27 Testing del prototipo. Fuente:Autor

De manera similar se puede realizar las pruebas unitarias de los diferentes métodos crud para las entidades conflictos, líneas de tiempo y hechos del conflicto por medio de phpUnit.

Despliegue a entorno de producción

Dentro de los ajustes previos a la fase de despliegue a hosting de una aplicación Laravel, se debe realizar el cambio de entorno debug, deshabilitándolo, esto en el archivo /config/app.php.

```
'debug' =>env('APP_DEBUG', false),
```

Opcionalmente se puede realizar el reset de los datos de la aplicación, mediante el comando:

```
vagrant@homestead:~/Code/rescon$  art mígrate:reset
```

Para el despliegue a hosting se realiza la contratación de servicios con la empresa hostgator (figura 27), las operaciones indicadas se realizan mediante Cpanel.

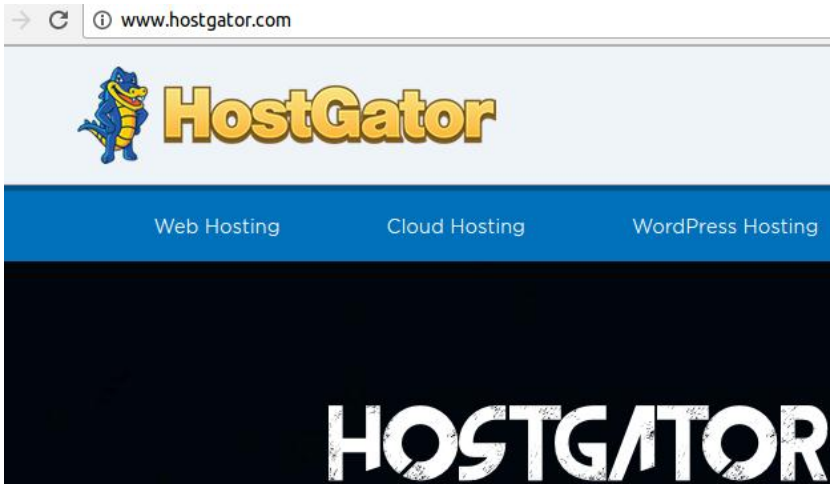

Figura 28. Homepage Hostgator. Fuente: Autor

Tal como lo ilustra la figura 28, se procede a exportar la base de datos desde el entorno local y se importa en PhpMyAdmin del servicio hosting, con los valores default:

Figura 29. Configuración dase de datos a producción. Fuente: Autor

Ajustamos el archivo .env con los valores de nombre de base de datos, usuario y contraseña de acuerdo a los asignados en el web hosting:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_DATABASE=nombreenhosting
DB_USERNAME=usuarioenhosting
DB_PASSWORD=contraseñaenhosting
```

Mediante la opción upload cargamos los archivos de la aplicación, dentro del directorio public_html (figura 29):

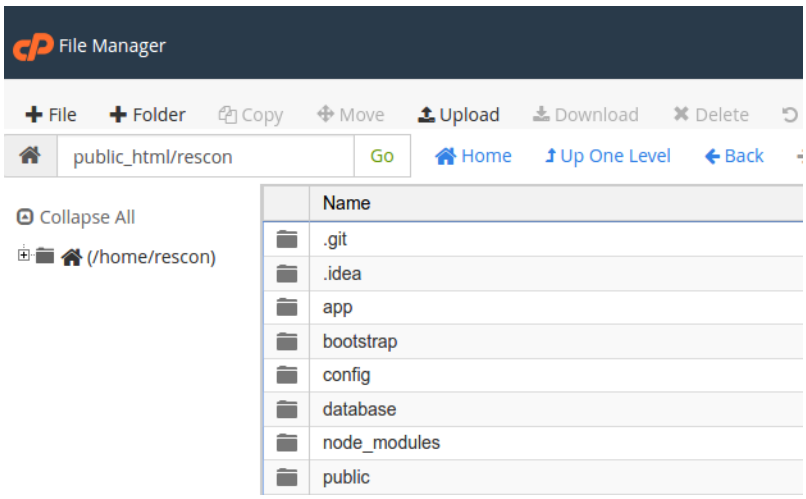

Figura 30. Despliege de prototipo a producción. Fuente: Autor

Conclusiones

La WebApp para la promoción de cultura de paz en la comunidad Unadista mediante herramientas de resolución y análisis de conflicto (líneas de tiempo) constituye una oportunidad de vincular las tecnologías Tics a una actividad realizada tradicionalmente mediante papel y lápiz, partiendo de un prototipo construido con el framework Laravel, bajo desarrollo ágil con herramienta Visual Paradigm.

La adaptación del enfoque de desarrollo ágil de software con herramienta Visual Paradigm, permitió la pertinencia y aceleración de la captura de las funcionalidades y especificaciones del sistema deseables en una aplicación web para la promoción de la paz en la universidad Nacional Abierta y a Distancia Unad, mediante la formulación de historias de usuario y wireframes del prototipo.

El registro de historias de usuario permitió la correcta delimitación la aplicación web para la promoción de la paz en la comunidad universitaria de la Unad, para el manejo de herramientas para la resolución de conflictos, implementado líneas de tiempo del conflicto, como objeto del prototipo.

El desarrollo de prototipos software con la integración de librerías y/o frameworks permite el desarrollo de software con características de calidad, al cual puede fácilmente hacerle modificación, una estructuración por directorios y el escalamiento bajo elementos construidos como cajas negras con funcionalidades como enrutado, consultas SQL, autenticación de usuarios, gestión de protocolo Http, ahorrando tiempo significativos al programador.

El desarrollo de aplicaciones software en la actualidad es realizado cada vez más sobre plataformas o máquinas virtuales, las cuales por defecto integran todos los programas y herramientas necesarias para el trabajo de implementación. De la misma manera en que facilita el despliegue a producción de las aplicaciones y permite el trabajo por equipo de desarrolladores mediante el intercambio y la simple especificación de la versión de la máquina.

Recomendaciones

Se resalta que el prototipo alcanzado puede avanzar hacia la implementación de las demás herramientas y la vinculación de áreas como inteligencia artificial para discriminar del tipo de conflicto y el perfil de los involucrados, así como estrategias de gamificación que incrementen el desempeño, productividad y vinculación de los usuarios-participantes de un conflicto

Existe la oportunidad de fortalecer e incrementar la dimensión investigativa de la Universidad Nacional Abierta y a Distancia mediante el trabajo conjunto de los programas de sicología, sociología e ingeniería de sistemas, a partir de la destinación de recursos y creación formal de instancias, organismos, procesos para la producción de conocimientos y software en áreas de las humanidades susceptibles a la sistematización o automatización.

Es deseable que las asignaturas del programa ingeniería de sistemas relacionadas con la programación fortalezcan las competencias de ingeniería de software, mediante la integración de experiencias actualizadas y modernas desde la especificación de requerimientos, la codificación y testing hasta el despliegue a producción, permitiendo a los estudiantes alcanzar el desarrollo de prototipos software de calidad.

Bibliografía

Agile team (2016) The Agile Movement. Consultado el 20 de octubre de 2016 de la url: http://agilemethodology.org/

El Tiempo.Redacción (2014).'Apps' para la paz y la reconciliación. Se escogieron cinco de una maratón de desarrollo y se lanzaron cuatro ya disponibles. Diario EL Teimpo.Consultado el día 11 de octubre de 2016 de la World Wide Web:http://www.eltiempo.com/tecnosfera/novedades-tecnologia/apps-para-la-paz-y-la-reconciliacion/14952078

Coronel Carlos.Morris Steven (2011). Base de datos, diseño, implementación y administración. Consultado el 4 de marzo de 2016 de la url: https://issuu.com/cengagelatam/docs/bases_de_datos_coronel

Fisher Simon (2000). WorkingWithConflict: Skills and Strategiesfor Action.Consultado el día 1 de octubre de 2016 de la World Wide Web:https://books.google.com.co/books?id=YCPEoKBlS54C&lpg=PA1&dq=simon%20fisher%20conflict%20action&hl=es&pg=PA1#v=onepage&q=simon%20fisher%20conflict%20action&f=false

Fundación Unir. (2008). Transformación constructiva del conflicto. Guía de capacitación. [pdf]Consultado el día 5 de octubre de 2016 de la World Wide Web:www.bivica.org/upload/transformacion-conflicto.pdf

Giraldo J. (2015). La paz también se construye en las universidades de Cali. Diario el País. Consultado el día 9 de octubre de 2016 de la World Wide Web:http://www.elpais.com.co/elpais/cali/noticias/paz-tambien-construye-universidades-cali

Git, 2010. GettingStarted - AboutVersion Control. Consultado el 5 de octubre de 2016 de la url: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

Gobierno de Colombia. (2012). Mesa de conversaciones para la terminación del conflicto y la construcción de una paz estable y duradera en Colombia. Consultado de el 8 de abril de 2016 de la url: https://www.mesadeconversaciones.com.co

Guzman H. (2014). Manual para el análisis y la intervención en conflictos sociales. [pdf]Consultado el día 5 de octubre de 2016 de la World Wide Web:http://formacionsocial.iteso.mx/documents/10901/0/Manual%2Bde%2BConflicto%2B2015%2BNov%2B2014/28410164-87bf-49ea-8bc4-0fc1e270e2ad?version=1.1

HashiCorp (2016). Developmentenvironmentsmadeeasy. Consultado el 4 de mayo de 2016 de la url: https://www.vagrantup.com/

InfyomLabs. (2015). Installation. Recuperado el 16 de septiembre de 2016, de http://labs.infyom.com/laravelgenerator/docs/develop/installation

Jetbrains (2016). Lightning-smart PHP IDE. Consultado el 1 de noviembre de 2016 dela url: https://www.jetbrains.com/phpstorm/

Kendall, kenneth E. y kendall, Julie E. Análisis y diseño de sistemas. Sexta edición. Consultado 18 de octubre de 2016 de la url https://luiscastellanos.files.wordpress.com/2014/02/analisis-y-disenio-de-sistemas-kendall-kendall.pdf

Laravel. (2015). LaravelHomestead. Recuperado el 12 de septiembre de 2016, de https://laravel.com/docs/5.1/homestead

Linowski (2016). Consultado el 6 de mayo de 2016 de la url: http://www.goodui.org/

Lorna Mitchell (2011). PHP Master: Write Cutting-edge Code. Consultado el 4 de abril de 2016, de la url: http://dl.finebook.ir/book/1f/11110.pdf

Luna F.(2014).Desarrollo web para Dispositivosmóviles. Herramientas para diseñar y programarwebapps. Users editors.

Mason Simon (2005). ConflictAnalysis Tools [pdf]. Consultado el día 5 de octubre de 2016 de la World Wide Web:http://www.css.ethz.ch/content/dam/ethz/special-interest/gess/cis/center-for-securities-studies/pdfs/Conflict-Analysis-Tools.pdf

MinTic. (2012). Ministerio TIC presentó iniciativas y programas del Plan Vive Digital en Andicom 2012. Consultado el 15 de abril de 2016 de la url: http://www.mintic.gov.co/portal/604/w3-article-1139.html

Naciones Unidas. (1999). Declaración sobre una cultura de paz. Consultada el 4 de abril de 2016 de la url: http://www3.unesco.org/iycp/kits/sp_res243.pdf

Pecoraro, C. J. (2015). MasteringLaravel. Developrobustmodern web-basedsoftwareapplications and RESTfulAPIswithLaravel. Birmingham, UK: Packt Publishing.

Presidencia República de Colombia. (2014). Decreto ley 1732. Recuperado el 14 de octubre de 2016, de http://wsp.presidencia.gov.co/Normativa/Leyes/Documents/LEY%201732%20DEL%2001%20DE%20SEPTIEMBRE%20DE%202014.pdf

Restrepo Gómez, B. (2003). Conceptos y Aplicaciones de la Investigación Formativa, yCriterios para Evaluar la Investigación científica en sentido estricto [pdf]. Consultado el día 27 de septiembre de 2016 de la World Wide Web: http://www.cna.gov.co/1741/articles-186502_doc_academico5.pdf

Salamanca M. (2016). Guía para la implementación de la cátedra de la paz. [pdf]. Santillana.Consultado el día 8 de octubre de 2016 de la World Wide Web:santillanaplus.com.co/pdf/cartilla-catedra-de-paz.pdf

Styde (2015). Modelfactories en Laravel 5.1. Consultado el 5 de abril de 2016 de la url: https://styde.net/model-factories-en-laravel-5-1/

Unad (2011). La investigación ciencias en la escuela de básicas, tecnología e ingeniería, Consultado el 27 de octubre de 2016 de la url: http://datateca.unad.edu.co/contenidos/243009/LINEAS_INV_ECBTI.pdf

Unad (2013). Reglamento Académico, General Estudiantil y de Egresados de laUniversidad Nacional Abierta y a Distancia (UNAD) [pdf]. Consultado el día 27 de septiembre de 2016 de la World Wide Web:http://datateca.unad.edu.co/contenidos/503018/PROYECTO_REGLAMENTO_ACADEMICO_GENERAL_ESTUDIANTIL_Y_DE_EGRESADOS.pdf

Unad (2016). Acerca de la UNAD. Consultado el 27 de octubre de 2016 de la url: https://informacion.unad.edu.co/transparencia-y-acceso-a-la-informacion/acerca-de-la-unad/

Unad. (2016). Lanzamiento CampoUNAD. Consultado el 27 de octubre de 2016 de la url: https://noticias.unad.edu.co/index.php/2-unad-noticias/1624-lanzamiento-campounad

Visual Paradigm International ltd (2016). Agile Handbook [pdf]. Consultado el día 28 de septiembre de 2016 de la World Wide Web:https://d1dlalugb0z2hd.cloudfront.net/resources/Agile-Handbook.pdf

Yank, K. (9 de octubre de 2001). Which Server-SideLanguageIsRightForYou? Recuperado el 28 de septiembre de 2016, de https://www.sitepoint.com/server-side-language-right/