Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

“Sistema Genérico en Web para el control, ingreso y estadísticas a eventos masivos (congresos)”

Tesina que presenta:

María Martha Quintana López

Para obtener el título de:

Ingeniero en Ciencias de la Computación

Asesor:

Carlos Armando Ríos Acevedo

Puebla, Pue. Noviembre 2018

ÍNDICE

ÍNDICE DE FIGURAS

DEFINICIONES, ACRÓNIMOS Y ABREVIATURAS

Del Negocio

A) **Administrador:** Persona que organiza el evento o es delegado para la administración del mismo.
B) **Asistente:** Persona interesada en acudir al evento masivo
C) **Monitor:** Persona que accede al sistema para verificar información de asistencia y tomar decisiones para el evento.
D) **Perfil:** Sección de la cuenta de cada usuario donde aparecen sus datos.
E) **Personal:** Persona que apoya durante la realización del evento.

De Tecnología

A) **API:** Conjunto de funciones y procedimientos que cumplen una o muchas funciones con el fin de ser utilizadas por otro software.
B) **Framework:** Estructura conceptual de soporte definida con artefactos o módulos de software concretos, en base al cual éste proyecto de software es organizado y desarrollado.
C) **SGBD:** Sistemas de gestión de bases de datos, software dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.
D) **GitHub:** Plataforma de desarrollo colaborativo para alojar proyectos basado en Git: Es un software de control de versiones
E) **Licencia MIT:** concede al usuario final los derechos de usar, copiar, modificar, publicar, distribuir o sublicenciar el software.

AGRADECIMIENTOS

Éste presente trabajo es obra de la inspiración e iluminación del dador de vida, fuente de sabiduría y conocimiento inagotable.

En memoria de la persona que veló por mi formación académica, humana y espiritual desde los primeros años de mi existencia, mi abuelita Francisca Sánchez.

Por el apoyo incondicional, la paciencia y perseverancia, agradezco a mis Padres Jorge Quintana y Martha López, quienes a través de su testimonio de vida, han forjado los valores que definen mi ser.

A mi hermana Dulce Quintana, con quien he compartido éstos años y he aprendido de su calidez humana.

A mi JUFRA, que me ha desafiado a elegir por una forma de vida distinta, que me ha hecho crecer en todo aspecto.

Al Ing. David Muñoz que me ha inspirado a realizar los anhelos del alma, me acompaña incansablemente en éste caminar y su testimonio es luz.

A mi familia y amigos que confiaron en mí, que recibí su ayuda sin antes pedirlo.

A mis Maestros: Carlos Armando, Meliza, Teresa por su trabajo docente ejemplar, en la labor de guiarme y conducirme hasta éste momento.

PREFACIO

El presente proyecto consiste en el análisis, diseño e implementación de un sistema web genérico de apoyo a la organización de eventos masivos, específicamente congresos en Instituciones Educativas. El propósito de la plataforma es proporcionar a los agentes involucrados en la organización de eventos institucionales, una herramienta para la óptima gestión de ingreso de los participantes, que facilite el control del personal de apoyo y realice el procesamiento de información generada, creando estadísticas que serán cruciales para la toma de decisiones, de manera que la aportación del sistema repercute en la mejora de la logística para la optimización de procesos, en la organización de eventos masivos. Para el desarrollo del sistema se incluyeron y analizaron las funciones aptas para su automatización y se tomaron en cuenta los mecanismos repetitivos, dentro del proceso de organización de eventos. Al ser éste un sistema genérico, se infiere que toma la información relevante y de uso general, dicha información es obtenida a partir un levantamiento de requerimientos del evento que llevó por nombre “Universiada”, realizado en la Benemérita Universidad de Puebla, de manera que el sistema pueda adaptarse a cualquier gestión de eventos y posteriormente ser personalizada.

Basado en la metodología de Proceso Unificado de Desarrollo de Software (RUP), éste proyecto asegura la producción de software de calidad, ajustándose a las necesidades de los usuarios finales dentro de un proceso iterativo e incremental, guiado por los casos de uso y basado en la arquitectura.

Para la gestión de datos, mediante la implementación de la teoría de Bases de Datos Relaciones, se realizó el diagrama y los modelos Entidad-Relación y Relacional que fueron auxiliares para el desarrollo de la Base de Datos, una vez realizado el proceso de normalización de tablas, garantizando así la ejecución de las funciones de integridad, persistencia y transacciones del sistema.

Dada la metodología propuesta de desarrollo, se utilizó Laravel, éste framework basado en PHP es robusto y permite rediseñar partes del sistema o

establecer nuevos requerimientos según el criterio del usuario final. Laravel es compatible con MySQL, gestor de base de datos seleccionado, ofrece un mejor rendimiento y tiempo de respuesta frente a operaciones específicas de lectura y escritura. El sistema web se ejecuta bajo un servidor Apache.

Para el aspecto visual y de interfaz, se construyeron las vistas a través del motor de plantillas Blade. Para un diseño responsivo se implementó el framework Bootstrap, incluido en el tema Admin-Lte que proporciona un tablero y estructura para el control de un panel. Con la finalidad de que el presente sistema se distribuya y se vea enriquecido con las contribuciones, será alojado en un repositorio a través de la plataforma web Github, de manera que sea de fácil acceso al público.

Finalmente, se realizaron métodos de pruebas modulares e integrales, así como pruebas unitarias, mismas que fueron aplicadas y documentadas.

INTRODUCCIÓN

En la organización de eventos masivos, se genera información que es necesaria para la toma de decisiones; éste conjunto de datos es proporcionado antes, durante y después del evento.

Antes del evento

Se entiende como asistentes del evento, a los interesados en acudir al mismo. La información de los asistentes es recopilada a partir del registro, el registro es útil para prever un estimado de asistentes y obtener su información personal; en muchas de las ocasiones el registro se realiza el mismo día del evento, lo que trae como consecuencia un desfase en los tiempos, en proporción a la demanda del número de asistentes, si aunado a esto el registro se hace mediante el sistema tradicional de papel y lápiz, a partir de la experiencia de los asistentes, a primera instancia se calificaría al evento negativamente. Cuando en el mejor de los casos, se realiza un pre registro, los asistentes comparten datos como nombre y teléfono, ésta información se torna vulnerable, y si no es protegida por los organizadores puede ser robada y mal utilizada, esto en cuestión de seguridad. Cuando se realiza un pre registro al evento, se generan datos que permiten decidir a partir del número de asistentes, el número de personal de apoyo, un presupuesto para catering, la cotización de insumos, etc.

Durante el evento

El personal de apoyo o también conocido como Staff, tiene un papel importante durante la realización del evento, al ser un recurso humano es indispensable administrarlo de la manera más óptima. Cada persona que apoya, desempeña una o varias tareas y se sitúa en un área. Dado que pueden recaer responsabilidades sobre cada persona de apoyo, es prescindible tener un registro de sus datos, esa información generada es valiosa para decidir si se dispone del personal en determinado momento y lugar. Además del personal, cada evento (congreso) lleva a cabo un programa de actividades, algunas de éstas se

desempeñan en espacios específicos y tienen un cupo limitado de personas, la incorrecta administración de estas actividades así como la ausencia de un pre registro de los asistentes a las mismas, provocaría desorden y falta de equidad en la distribución de los asistentes en las actividades, esto también repercutiría en la forma en cómo se distribuye el personal de apoyo.

Después del evento

Dentro de la información que proporcionó el asistente, es importante tener un dato de contacto, que permita mantener a los organizadores en comunicación con los asistentes, para que éstos puedan dar información de valor a través de encuestas, enviadas por correo electrónico o preguntas telefónicas de esa forma se obtienen datos importantes para la futura organización del evento.

Éste proyecto analiza los datos que son cruciales para el organizador de eventos, de manera que, se implementen estadísticas e informes, integrados en un sistema web genérico.

Objetivos

General.

Desarrollar (analizar, diseñar e implementar) un sistema web genérico para la administración de eventos masivos (congresos).

Específicos.

- Desarrollar el módulo de personalización del evento.
- Desarrollar el módulo de registro de asistentes.
- Desarrollar el módulo de estadísticas y reportes.

Metodología de desarrollo

Este proyecto empleará la metodología del Proceso Unificado de Desarrollo de Software el cual es guiado por los caso de uso, basado en la arquitectura y es iterativo e incremental. Su propósito es asegurar la producción de software de alta calidad que se ajuste a las necesidades de sus usuarios finales con unos costos y

calendario predecibles. Dicha metodología utiliza el Lenguaje Unificado de Modelado (UML) el cual permite construir los siguientes modelos: casos de uso, diseño, implementación, despliegue y pruebas [1]. De esta forma, para cada módulo se construyen los modelos anteriores.

CAPÍTULO I
ESTADO DEL ARTE

1.1 Estado del Arte

En la última década los sistemas de software basados en web han tenido un gran auge, por lo que confirmar la asistencia presencial a un evento a distancias largas ya no es una limitante a través del pre registro en línea. Esto beneficia al organizador del evento que quiere optimizar tiempo en el registro. Cabe mencionar, que la información, de ser recopilada y procesada por el sistema antes, durante y después del evento, es capaz de apoyar al organizador para llevar a cabo una toma inteligente de decisiones. Sin embargo, existen eventos en donde se sigue realizando el registro de participantes a través del más antiguo sistema de lápiz y papel, bien porque el evento es pequeño o porque mayormente el software de registro en el mercado es de paga. A continuación se muestran las plataformas web de mayor auge en el mercado.

1.1.1 Boletia [10]

Es una plataforma en línea para el registro de asistentes, cobro de boletos y control de acceso a eventos. Boletia fue creada por emprendedores venezolanos; esta empresa tiene por misión ofrecer a los asistentes de eventos del usuario registrado en su sitio web, diferentes formas de pago y al alcance de todos, como OXXO, PayPal, tarjeta de débito y crédito y depósito bancario. La promoción de los boletos se realiza con la ayuda de un micro sitio web dentro de la página de Boletia® utilizado como canal de venta y promoción de los boletos. Con esta modalidad es posible personalizar los tipos de boleto, las formas de registro y de pago, entre otros. Ésta plataforma recibe una comisión de 2.5% + $5 a partir de cada boleto vendido, además de las comisiones respectivas a cada forma de pago. Cada asistente al evento recibirá un boleto electrónico vinculado a las opciones de control de acceso, éstas pueden ser web o móvil. El servicio que Boletia presta, no implica la prestación de servicios de promoción, mercadotecnia, organización, publicidad, ni cualquier otro servicio no descrito en los Términos y Condiciones. El sistema de organización

de eventos controlado por el Organizador y ofrecido mediante la Plataforma Digital, permite a los Organizadores tener un control directo sobre su boletaje, precios, administración financiera de eventos, y facilita a los Usuarios la cobranza y adquisición de boletos, guardar boletos de forma digital, agilizar el registro de asistentes a eventos y promoción de eventos.

Cabe mencionar que en la cláusula QUINTA de Territorio, dice que: los Servicios prestados por Boletia deberán de entenderse como servicios prestados en la República Mexicana sin importar en donde fueron originados o solicitados. Dentro de las estadísticas que ofrece, se limita al número de boletos vendidos y al número de ventas totales.

1.1.2 Evenbrite [11]

Es una plataforma global para experiencias en directo que permite que cualquiera pueda crear, compartir, encontrar y asistir a eventos.

Los organizadores pueden usar Eventbrite de forma gratuita siempre y cuando no cobren por las entradas. No hay cargos mensuales, de inscripción ni de configuración. Si se cobra por las entradas vendidas, las tarifas varían según el plan. En Estados Unidos., el plan Essentials cuesta un 2 % del precio de la entrada y 0,79 $ por entrada pagada más un 2,5 % de procesamiento de pagos por transacción; el plan Professional es un 3,5 % del precio de la entrada y 1,59 $ por entrada pagada más un 2,5 % de procesamiento de pagos por transacción; el plan Premium tiene precios personalizados. Dentro de las características que ofrece se encuentran: **Servicio de Email marketing:** Ofrece una Newsletter con mensajes personalizados para mantener informados a todos los seguidores del evento.

Facilita la difusión: Permite compartir el evento en las Redes Sociales, lo que potencia el alcance de la promoción

Geolocalización: Tiene un sistema de mapas con el que le facilitarás a los asistentes el punto exacto al cual deben acudir para la celebración del evento.

Cuenta con un sistema de navegación: Cuenta con un sistema de rastreo en tiempo real. El sistema de navegación de Eventbrite permite conocer cuál es la interacción e impacto que está teniendo la promoción que se está realizando desde la página web.

Soporte técnico de calidad: Tiene un buen soporte técnico que puede solucionar todas tus dudas y ofrecerte otro tipo de herramientas o estrategias que sirvan para alcanzar más ventas y permitir que el mensaje se propague con mayor efectividad.

Panel de gestión: Dentro del sistema de navegación, existe un área donde se puede analizar los invitados. Permite extraer toda la información en una Hoja de Cálculo de Excel.

Personalización: Si se accede al servicio de pago de Eventbrite entonces se podrá ubicar el logotipo de la empresa promotora del evento u otras variantes de la interfaz de la página que se ofrecen para publicar y promocionar.

App: El servicio puede ser visto a través de su propia aplicación para Smartphone o Tablet.

1.1.3 Google Forms [12]

Es una herramienta que permite generar formularios online sin tener conocimiento de código, y acceder rápidamente a los resultados usando una cuenta personal de Google®, en forma de una planilla también de Google Docs. ®. Es una aplicación gratuita y es capaz de guardar automáticamente los resultados del formulario en una hoja de cálculo de Google para que sea posible realizar un análisis avanzado, es intuitivo y rápido editar todos los campos que sean necesarios.

Google Forms incluye todos los campos básicos (opciones múltiples, menús desplegables, textos) pero no se encuentran campos para pagos o para subir archivos.

Si se requiere agregar campos y funciones adicionales con Google Forms Add-ons® y compartir el formulario para que se colabore para la creación del formulario.

Como se mencionó anteriormente es gratuito el servicio con una cuenta de Gmail. O desde $ 5 al mes con una cuenta de G Suite®.

1.1.4 Survey Monkey [13]

Es una herramienta web para la creación de encuestas online. La aplicación te permite diseñar y enviar encuestas a través del correo electrónico, un enlace en tu web o página de facebook, entre otras. Después se podrán consultar los resultados y analizarlos para tomar decisiones. Para el diseño de la encuesta se cuenta con modelos ya creados o si el usuario lo prefiere se puede realizar una encuesta personalizada construyendo las preguntas que se desean formular.

Para hacer uso de la herramienta es necesario registrarse en la plataforma www.surveymonkey.com y escoger el “plan” más conveniente para cada necesidad. SurveyMonkey® ofrece un “plan BASIC” gratuito con diversas funcionalidades que se pueden utilizar para empezar con la herramienta. Una vez registrado, el usuario podrá crear una encuesta nueva, diseñar las preguntas o seleccionar una plantilla de encuesta diseñada por un experto. Modificar las opciones de la encuesta: numeración de preguntas y de páginas, logo de la empresa (sólo disponible en versiones de pago), idiomas, etc. Enviar la encuesta seleccionando una o varias de las diferentes formas de envío. Por ejemplo, a través de correo electrónico o Facebook. Esperar a obtener resultados; los resultados se pueden consultar desde la plataforma web y las versiones de pago permiten también descargarlas al ordenador, establecer filtros o tabular datos cruzados.

1.2 Justificación

De las plataformas mencionadas anteriormente como herramientas de registro para eventos, se puede observar que se tienen dos clasificaciones, la primera clasificación corresponde al enfoque de boleteras, es decir, que generan boletos a partir de un registro de usuarios y posteriormente es posible realizar el *check-in* al asistir al evento, pero no se realiza una administración del personal de apoyo y sólo se tiene acceso a la información a través de un solo usuario, por lo que el rol del monitor no participa en la plataforma, mientras que con respecto a la información estadística, sólo se muestra el número de

participantes registrados y los que confirmaron la asistencia. El segundo enfoque corresponde a la recolección de información de los asistentes a través de formularios, cabe mencionar que ésta es una herramienta muy recurrida por los organizadores de talleres y eventos institucionales, debido a la simplicidad de su uso, pero puede representar una amenaza a la seguridad de la información, dado que obteniendo la URL[1] de la hoja de Excel es posible acceder a ésta, además de que los formularios no validan la repetición de los datos.

Es visible que dentro de las soluciones que ofrecen las plataformas, la información no está centralizada, pues mientras una plataforma recolecta los datos de los usuarios y hace tardío el proceso de registro de asistencia. En el otro caso, la plataforma carece de mayor control sobre la logística del evento, como la ausencia de administración del personal, que permite la toma de decisiones. En la misma herramienta, si se requiere obtener estadísticas, es necesario procesar la información manualmente, descargando de la plataforma las hojas de Excel.

[1] Uniform Resource Locator (Localizador Uniforme de Recursos). Se trata de la secuencia de caracteres que sigue un estándar y que permite denominar recursos dentro del entorno de Internet para que puedan ser localizados.

CAPÍTULO II
MARCO TEÓRICO

Es necesario definir los conceptos teóricos que se aplican durante los procesos y actividades que se llevan a cabo para el correcto cumplimiento de los objetivos propuestos. A continuación se muestran los conceptos básicos para comprender la terminología mencionada en este trabajo.

2.1 Metodologías de Análisis y diseño de software

Una metodología de desarrollo de software puede ser vista como un proceso con una serie de actividades relacionadas que conduce a la elaboración de un producto de software. Todas las metodologías se refieren a un marco de trabajo que es utilizado para estructurar, planear y controlar el proceso de desarrollo en sistemas de información [3]. Los enfoques más generales, que se desarrollan en las metodologías son los siguientes:

- *Modelo en cascada*: Lineal.
- *Modelo de desarrollo Evolutivo (Espiral):* Combinación de marco de trabajo lineal e iterativo.
- *Incremental:* Combinación de marco de trabajo lineal e iterativo.
- *Prototipado:* marco de trabajo iterativo.
- *RAD:* Rapid Application Development, marco de trabajo iterativo.
- *OO:* Orientado a Objetos, combina lo mejor de las demás metodologías.

- *Modelo en cascada*

Considera las actividades fundamentales del proceso especificación, desarrollo, validación y evolución. Los representa como fases separadas del proceso, tales como la especificación de requerimientos, el diseño del software, la implementación, las pruebas, etcétera [17].

- *Espiral*

Este enfoque entrelaza las actividades especificación, desarrollo y validación. Es decir surge de un sistema inicial que se desarrolla rápidamente a partir de especificaciones abstractas. Basándose en las peticiones del cliente para producir un sistema que satisfaga sus necesidades [17].

- *Incremental*

Se lleva a través de entregas pequeñas y frecuentes del sistema y por medio de un enfoque que sirve para la descripción de requerimientos basado en las historias de clientes o escenarios que pueden ser la base para el proceso de planificación [17].

- *Prototipado*

Un prototipo es una versión no terminada del producto que se le entregará al cliente o usuario, como base para la generación de distintas versiones del producto final [16].

- *Rapid Application Development (RAD)*

El *desarrollo rápido de aplicaciones* (RAD) está basado en el uso de las iteraciones y principalmente en el manejo de prototipos [16].

- *Metodologías de desarrollo Orientado a objetos*

a) *Diseño orientado a objetos (OOD) de Grady Booch:* también conocido como Análisis y Diseño Orientado a Objetos (OOAD). El modelo incluye seis diagramas: de clase, objeto, estado de transición, la interacción, módulo, y el proceso [18].

b) *Top-downprogramming*: evolucionado en la década de 1970 por el investigador de IBMHarlan Mills (y NiklausWirth) en Desarrollo Estructurado [18].

c) *Proceso Unificado*: es una metodología de desarrollo de software, basado en UML. Organiza el desarrollo de software en cuatro fases, cada una de ellas

con la ejecución de una o más iteraciones de desarrollo de software: creación, elaboración, construcción y las directrices. Hay una serie de herramientas y productos diseñados para facilitar la aplicación. Una de las versiones más populares es la de Rational Unified Process [1].

2.1.1 Proceso Unificado Racional (RUP)

Teniendo en cuenta que en el sistema que está desarrollando se van a aplicar métodos y tecnologías de vanguardia que ofrezcan ventajas en materia de análisis, diseño, desarrollo y mantenimiento del mismo. Y considerando que las características y prestaciones que ofrece la metodología del Proceso Unificado (UP), cumplen con los requerimientos que se tienen plateados para su elaboración, se ha optado por aplicar dicha metodología para la elaboración del presente trabajo, la cual se describe a continuación:

El Proceso Unificado de Desarrollo Software o simplemente Proceso Unificado (UP) es un marco de desarrollo de software que se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y por ser iterativo e incremental. El refinamiento más conocido y documentado del Proceso Unificado es el Proceso Unificado de Rational o simplemente RUP [1].

El Proceso Unificado es un marco de trabajo extensible que puede ser adaptado a organizaciones o proyectos específicos. De la misma forma, el *Proceso Unificado de Rational (RUP)*, también es un marco de trabajo extensible, por lo que muchas veces los dos nombres suelen utilizarse para referirse a un mismo concepto [1].

Características:

- Iterativo e incremental

El Proceso Unificado es un marco de desarrollo iterativo e incremental compuesto de cuatro fases denominadas Inicio, Elaboración, Construcción y

Transición. Cada una de estas fases es a su vez dividida en una serie de iteraciones (la de inicio puede incluir varias iteraciones en proyectos grandes). Estas iteraciones ofrecen como resultado un *incremento* del producto desarrollado que añade o mejora las funcionalidades del sistema en desarrollo. Aunque todas las iteraciones suelen incluir trabajo en casi todas las disciplinas, el grado de esfuerzo dentro de cada una de ellas varía a lo largo del proyecto [18].

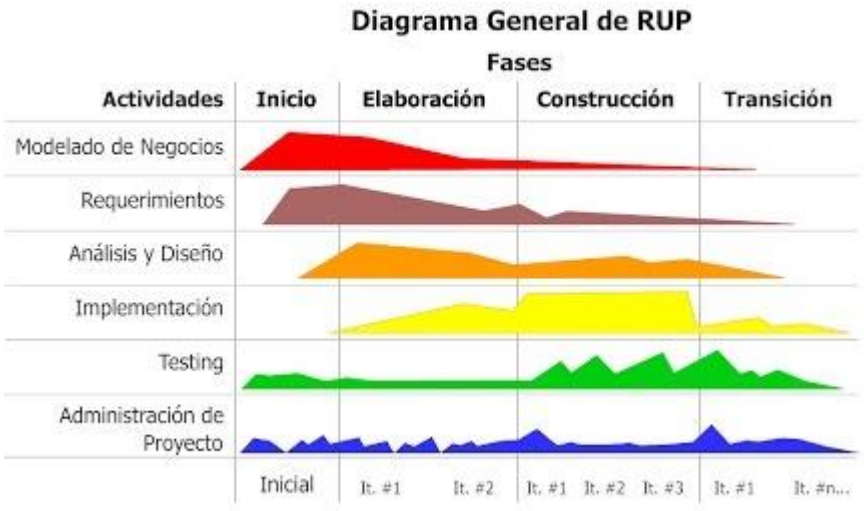

Figura 1 Diagrama de Fases de proceso unificado.

El diagrama anterior ilustra como el énfasis relativo en las distintas disciplinas cambia a lo largo del proyecto.

- Dirigido por los casos de uso

En el Proceso Unificado los casos de uso se utilizan para capturar los requisitos funcionales y para definir los contenidos de las iteraciones. La idea es que cada iteración tome un conjunto de casos de uso o escenarios y desarrolle todo el camino a través de las distintas disciplinas: diseño, implementación, prueba, etc. el proceso dirigido por casos de uso es el RUP [18].

- Centrado en la arquitectura

El Proceso Unificado asume que no existe un modelo único que cubra todos los aspectos del sistema. Por dicho motivo existen múltiples modelos y vistas que definen la arquitectura de software de un sistema. La analogía con la construcción es clara, cuando construyes un edificio existen diversos planos que incluyen los distintos servicios del mismo: electricidad, fontanería, etc. [18].

- *Enfocado en los riesgos*

El Proceso Unificado requiere que el equipo del proyecto se centre en identificar los riesgos críticos en una etapa temprana del ciclo de vida. Los resultados de cada iteración, en especial los de la fase de Elaboración deben ser seleccionados en un orden que asegure que los riesgos principales son considerados primero [18].

2.2 Lenguaje Unificado de Modelado

El soporte que utiliza la metodología de Proceso Unificado (UP) para diseñar los modelos visuales de un sistema, es el Lenguaje Unificado de Modelado o UML, El vocabulario y reglas de éste lenguaje se centran en la representación conceptual y física del sistema. El lenguaje UML tiene una notación gráfica muy expresiva que permite representar en mayor o menor medida todas las fases de un proyecto: desde el análisis con los casos de uso, el diseño con los diagramas de clases, objetos, etc., hasta la implementación y configuración con los diagramas de despliegue. UML sirve para el modelado completo de sistemas complejos, tanto en el diseño de los sistemas software como para la arquitectura hardware donde se ejecuten [18].

2.3 Arquitectura Cliente – Servidor

La arquitectura Cliente/Servidor es un modelo para el desarrollo de sistemas de información, en el que las transacciones se dividen en elementos independientes que cooperan entre sí para intercambiar información, servicios o recursos.

Bajo este modelo cada usuario tiene la libertad de obtener la información que requiera en un momento dado proveniente de una o varias fuentes locales o distantes y de procesarla según le convenga. Los distintos servidores también pueden intercambiar información dentro de esta arquitectura [19].

2.3.1 Características

- El servidor presenta a todos sus clientes una interfaz única y bien definida.
- El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.
- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- Los cambios en el servidor implican pocos o ningún cambio en el cliente [20].

La arquitectura Cliente/Servidor es una infraestructura versátil, modular y basada en mensajes que mejora la portabilidad, la interoperabilidad y la escalabilidad de los sistemas. Las aplicaciones clientes son conocidas como aplicaciones frontales. Están optimizadas para la interacción con el usuario [20].

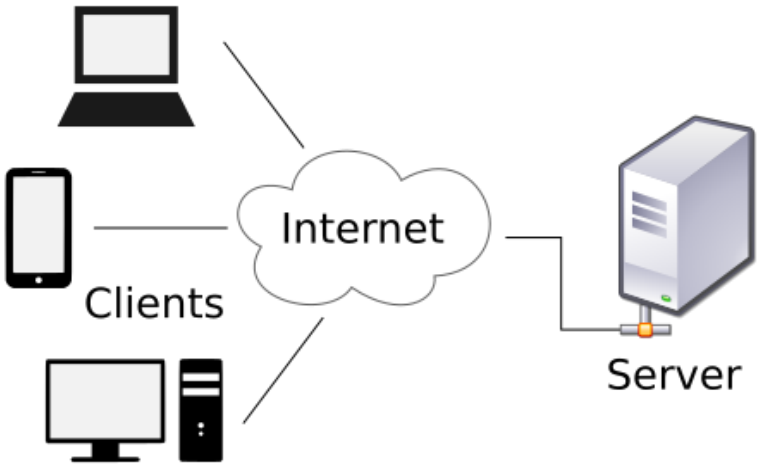

Figura 2. Diagrama Cliente –Servidor [20].

2.3 Arquitectura MVC

La arquitectura MVC (Model View Controller) fue introducida como la parte del lenguaje de programación Smalltalk2[2]. Fue diseñada para reducir el esfuerzo de programación necesario en la implementación de sistemas. Sus características principales son que: el Modelo, las Vistas y los Controladores se tratan como entidades separadas; esto hace que cualquier cambio producido en el Modelo se refleje automáticamente en cada una de las Vistas [8]. “En líneas generales MVC es una propuesta de diseño de software utilizada para implementar sistemas donde se requiere el uso de interfaces de usuario, surge de la necesidad de crear software más robusto con un ciclo de vida más adecuado donde se potencie la facilidad de mantenimiento y reutilización del código, y la separación de los conceptos”[3].

2.3.1. El Modelo

Es la capa donde trabaja con los datos y las reglas de negocio que gobiernan la manipulación de los mismos. El Modelo no tiene conocimiento específico de los Controladores o de la Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vista cuando cambia el Modelo [8].

2.3.2. La Vista

Muestra el contenido del modelo, la Vista accede a los datos a través del Modelo y especifica como los datos deben ser desplegados. También es responsable de mantener la consistencia en la presentación cuando el Modelo sufre cambios. Esto lo puede lograr gracias a una referencia al propio modelo [8].

2.3.3 El Controlador

Es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo.

Cuando se realiza algún cambio entra en acción, bien sea por cambios en la información del Modelo o por alteraciones de la Vista, es el código de lo que se llama habitualmente como “lógica del negocio”, ya que nos permite implementar las

[2] Smalltalk es un lenguaje reflexivo de programación, orientado a objetos y con tipado dinámico.
[3] Miguel Ángel Álvarez – desarrolloweb.com – 2 – Enero - 2014

operaciones típicas que pueda requerir nuestra aplicación, como editar un elemento, realizar una compra, una búsqueda de información, etc. [8].

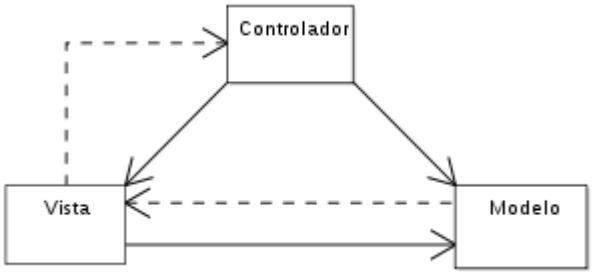

Figura 3. Interacción del modelo MVC [22].

2.3.4 Características generales de MVC

Dentro de las características generales de la arquitectura MVC es la fácil y flexible estructuración del código, es decir una "separación de poderes" entre los datos, la implementación e interfaz de la aplicación. Además de esto MVC es un patrón de diseño de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos de forma que las modificaciones al componente de la vista o a cualquier parte del sistema puedan ser hechas con un mínimo impacto en el componente del modelo de datos o en los otros componentes del sistema.

2.4 Servidor Apache

El servidor HTTP[4] Apache es un servidor web de software libre desarrollado por la Apache Software Foundation (ASF). El producto obtenido de este proyecto es un servidor de código fuente completo, descargable y gratuito. Apache es robusto y con un ciclo de desarrollo muy rápido gracias a la gran cantidad de colaboradores voluntarios de que dispone. Es también un servidor estable, eficiente, extensible y multiplataforma. Estable: es una consecuencia de su probada robustez que impide caídas o cambios en el servidor inesperados. Flexible y eficiente: es capaz de

[4] HTTP (Hypertext Transfer Protocol) es el protocolo que permite la transferencia de información a través de la web.

trabajar con el estándar HTTP/1.1[5] y con la mayor parte de las extensiones web que existen en la actualidad. Es extensible: dispone de gran cantidad de módulos que amplían su funcionalidad. Multiplataforma ya que está disponible para diferentes plataformas como GNU/Linux, Windows, MacOS [23].

2.4 PHP

PHP es un lenguaje que se ejecuta del lado del servidor, es un lenguaje de scripting[6] de propósito general y de código abierto que está pensado específicamente en el desarrollo web y puede ser incrustado en páginas HTML. El objetivo principal del lenguaje es permitir al desarrollador web escribir dinámica y rápidamente páginas web generadas. Es un lenguaje portable, multiplataforma, compatible con los sistemas operativos principales, lo que da pie a tener virtual libertad de elegir qué servidor web se usará para la ejecución del lenguaje. Y este lenguaje se puede escribir de manera procedimental o con estilo de programación orientada a objetos [24].

La generación de lenguaje que tiene inherente PHP no está limitada a HTML sino a la generación de cualquier tipo de texto, como archivos XML[7]. Y si vamos más lejos, PHP puede crear imágenes, PDF sobre la marcha. Una de las características más importantes de este lenguaje es el soporte que tiene para una amplia gama de bases de datos, como lo son MySQL y Oracle, mediante las extensiones propias de cada base de datos, usando la capa de abstracción PDO o usando ODBC [24].

[5] En la versión 1.1 se tienen los verbos GET, POST, PUT, DELETE y la web se orienta a recursos (REST).
[6] Es un lenguaje interpretado, es decir, para ejecutar las instrucciones de éste lenguaje existe un intérprete que procesa cada orden.
[7] XML proviene de eXtensible Markup Language (“Lenguaje de Marcas Extensible”). Se trata de un metalenguaje (un lenguaje que se utiliza para decir algo acerca de otro) extensible de etiquetas que fue desarrollado por el Word Wide Web Consortium (W3C), una sociedad mercantil internacional que elabora recomendaciones para la World Wide Web.

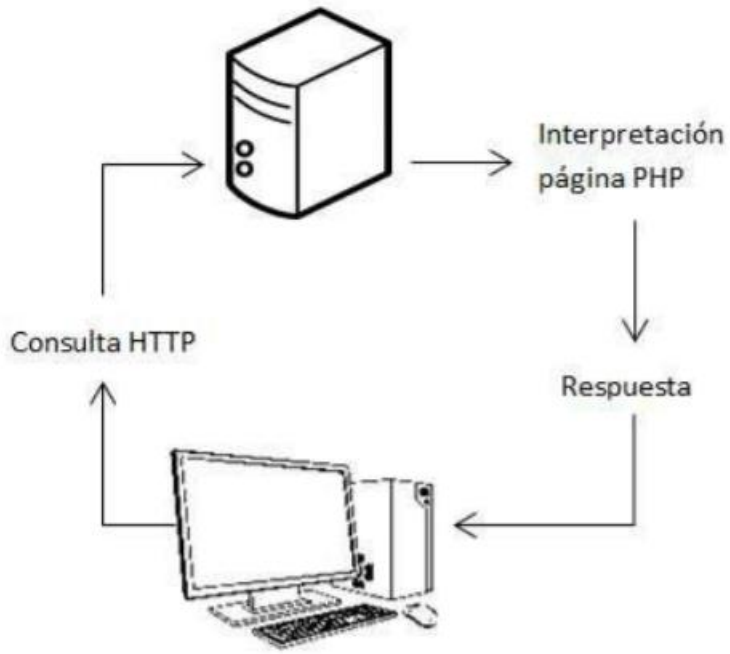

Figura 4. Funcionamiento de PHP [24].

2.5 Framework

Framework es una estructura tecnológica de soporte definido, con módulos completos de software que sirven de base para la organización y el desarrollo de aplicaciones. Los objetivos principales que persigue un framework son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones. Un framework Web, por tanto, podemos definirlo como un conjunto de componentes (por ejemplo clases en java y descriptores y archivos de configuración en XML) que componen un diseño reutilizable que facilita y agiliza el desarrollo de sistemas Web [25].

2.6 Laravel

Laravel es un framework de código abierto para desarrollar aplicaciones y servicios web con PHP, su filosofía es desarrollar código de forma elegante y simple, evitando su repetición [15].

2.6.1 Características

Dentro de las principales características que posee Laravel, para el aumento de la productividad, al desarrollar el sistema, se tienen las siguientes:

- *Modularidad*

  Laravel se ha construido utilizando más de 20 librerías diferentes fuertemente integradas con el gestor de dependencias Composer[8].

- *Testeabilidad*

  Construido para facilitar el testeo, Laravel tiene con varios asistentes que ayudan a visitar las rutas de testeo, navegando por el HTML resultante para asegurar que los métodos que se llaman desde las diferentes clases sean correctos.

- *Enrutamiento (routing)*

  Laravel proporciona una extrema flexibilidad en la definición de las rutas de la aplicación. Inspirado en la filosofía de los micro-frameworks Sinatra[9] y Silex[10]. Además es posible adjuntar funciones de filtro que se ejecuten en rutas específicas.

- *Gestor de configuración*

  Frecuentemente la aplicación se ejecutará en diferentes entornos, esto quiere decir que tanto la base de datos como credenciales o dominios serán diferentes si se ejecutan el local en el entorno de test o en los servidores de producción. Laravel nos permite definir configuraciones separadas para cada uno de los entornos.

- *Confeccionador de consultas y ORM (Object Relational Mapper)*

  Cuando se instala Laravel viene con un constructor de consultas, este nos permite lanzar consultas a la base de datos con una sintaxis PHP de métodos enlazados, en lugar de tener que escribir la SQL completa.

[8] Manejador de dependencias de PHP, capaz de instalar las librerías que requiere un proyecto con las versiones que necesiten.
[9] Se caracteriza por la sencillez, flexibilidad, las soluciones ágiles, livianas y efectivas.
[10] Su funcionamiento básico consiste en definir controladores y asociarlos con rutas, todo en un solo paso.

Además proporciona un ORM[11] y una implementación de Registro Activo (ActiveRecord) llamado Eloquent[12], que permite definir modelos interconectados. Estos componentes son compatibles con distintas bases de datos.

- *Confeccionador esquema, migraciones y repoblaciones.*

  Éstas características permiten definir un esquema de base de datos dentro de PHP y mantener un registro de los cambios para así ayudar en la migración de base de datos. Las repoblaciones (Seeding) permiten poblar las tablas seleccionadas de una base de datos una vez realizada la migración para de esta forma rellenar con datos las tablas.

- *Motor de plantillas*

  Laravel viene con Blade, un lenguaje ligero de plantillas con el cual se pueden crear diseños anidados con bloques predefinidos en el que el contenido se inserta dinámicamente. Además Blade guarda en caché los archivos generados.

- *Email*

  Con la clase Mail que es un derivado de la librería SwiftMailer[13], Laravel proporciona una forma muy sencilla de enviar e-mails, con contenido HTML y adjuntos.

- *Autenticación*

  Laravel viene con las herramientas para crear en toda web un formulario de registro, autenticación e incluso envío de contraseñas a usuarios que no la recuerden.

[11] Permite convertir los datos de objetos en un formato correcto para poder guardar la información en una base de datos (mapeo).
[12] Es el ORM que incluye Laravel para manejar de forma sencilla los procesos correspondientes al manejo de bases de datos en un proyecto.
[13] Es una librería en PHP basada en componentes para enviar correo desde proyectos de desarrollo web.

- *Colas*

  El Framework se integra con diversos servicios de colas, para permitir el aplazamiento de tareas que son muy intensivas en recursos, así por ejemplo podemos enviar una gran cantidad de e-mails ejecutando esta tarea en segundo plano en lugar de hacer que el usuario espere delante de la pantalla [26].

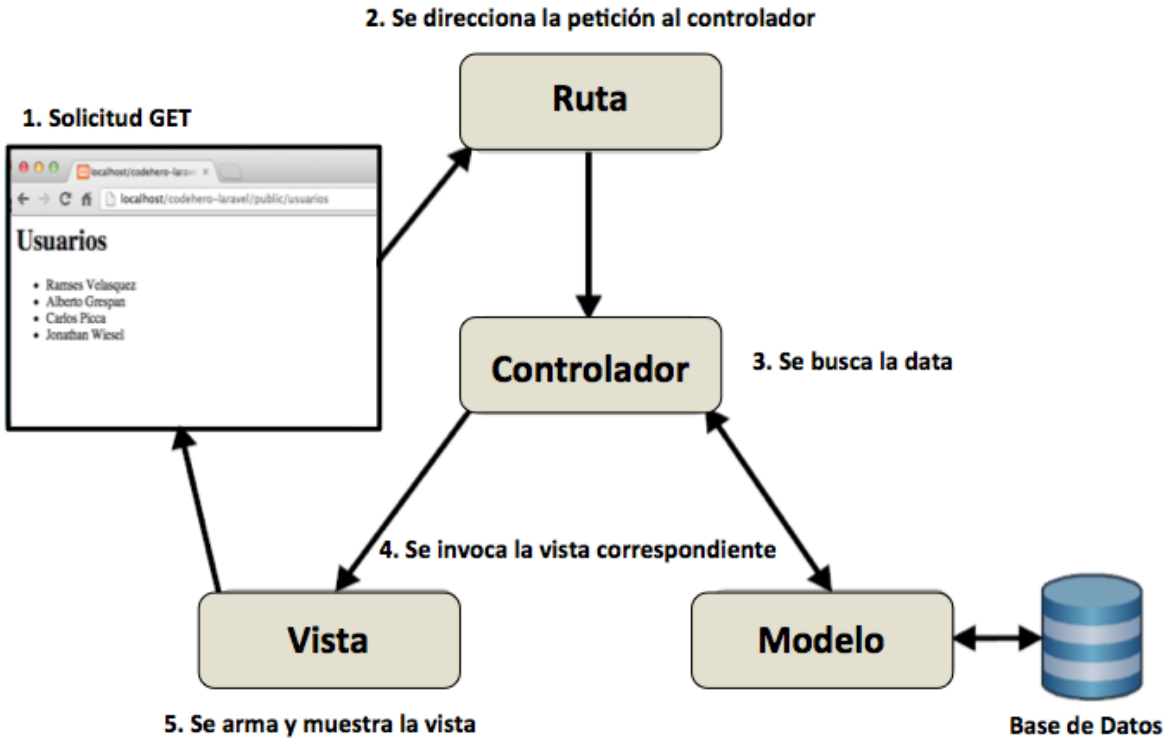

Figura 5. Enrutamiento básico de Laravel [26].

2.6.1 Ventajas

Dentro de las principales ventajas que ofrece el desarrollo a partir de éste Framework, se tienen las siguientes:

- Reducción de costos y tiempos en el desarrollo y posterior mantenimiento de la aplicación creada.
- Curva de aprendizaje relativamente baja si se compara con otros frameworks de PHP.

- Flexible y adaptable no sólo al uso del sistema MVC tradicional, sino que para reducir las líneas de código.
- Es modular y con un amplio sistema de paquetes y drivers con el que se puede extender las funcionalidades de forma sencilla, robusta y segura. Se caracteriza por la sencillez al momento de utilizar los datos mediante Eloquent.
- Facilita el manejo de las rutas de una aplicación, así como la generación de URLs amigables que ayudan a mejorar el posicionamiento web.
- Usa el sistema de plantillas Blade, que se caracterizan por ser más simples y que además incluyen un sistema de caché que las hace más rápidas.
- Una gran comunidad y mucha documentación, sobre todo en su sitio oficial.
- Cuenta con una herramienta de líneas de comando llamada *Artisan* que permite desarrollar tareas programadas como por ejemplo para ejecutar migraciones, pruebas a determinadas horas [28].

2.7 Bootstrap

Es un framework originalmente desarrollado por *Twitter*, que permite crear interfaces web con CSS[14] y JavaScript[15], cuya particularidad es la de adaptar la interfaz del sitio web al tamaño del dispositivo en que se visualice. Es decir, el sitio web se adapta automáticamente al tamaño de una PC, una Tablet u otro dispositivo. Esta técnica de diseño y desarrollo se conoce como *Responsive Design* o Diseño Adaptativo.

2.8 Diseño de la Base de Datos

Los sistemas de bases de datos se diseñan para gestionar grandes cantidades de información. La colección de datos, normalmente denominada base de datos, contiene información relevante principalmente para una empresa. La gestión de los datos implica tanto la definición de estructuras para

[14] CSS (hojas de estilo en cascada) es un lenguaje que define la apariencia de un documento escrito en un lenguaje de marcado.
[15] Es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas

almacenar la información como la provisión de mecanismos de manipulación de la información [4].

Bajo la estructura de la base de datos se encuentra el modelo de datos, el cual es una colección de herramientas conceptuales para describir los datos, las relaciones, la semántica y las restricciones de consistencia. Un modelo de base de datos muestra la estructura lógica de la base, incluidas las relaciones y limitaciones que determinan cómo se almacenan los datos y cómo se accede a ellos [4].

2.8.1 Modelo Entidad-Relación (E-R)

Está basado en una percepción del mundo real que consta de un conjunto de objetos básicos llamados entidades y de relaciones entre estos objetos. Una entidad es una “cosa u objeto” en el mundo real que es distinguible de otros [5].

El modelo es capaz de asistir al diseñador para definir y entender las cosas significativas acerca de cuál información necesita ser conocida o manejada y también las relaciones entre estas [5].

2.8.2 Normalización

La teoría de la normalización propone un procedimiento riguroso para el diseño de bases de datos. Una base de datos mal diseñada puede funcionar inicialmente pero puede mostrar anomalías en el almacenamiento debidas al agrupamiento indiscriminado de los campos cuando se efectúan en los archivos las operaciones de inserción, actualización o eliminación. La teoría de la normalización ayuda a reconocer las cualidades no deseadas en un archivo y la forma de corregirlas [6].

Para realizar el proceso de normalización, se debe determinar la forma normal de una relación a partir de su relación de atributos conocida como “dependencia funcional” [6].

2.9 Gestor de Base de Datos

Un Sistema Gestor de Base de Datos (SGBD, en inglés DBMS: DataBase Management System) es un sistema de software que permite la definición de bases de datos; así como la elección de las estructuras de datos necesarios para el almacenamiento y búsqueda de los datos, ya sea de forma interactiva o a través de un lenguaje de programación. Un SGBD relacional es un modelo de datos que facilita a los usuarios describir los datos que serán almacenados en la base de datos junto con un grupo de operaciones para manejar los datos [29].

2.10 MySQL

Es un sistema gestor de bases de datos relacionales rápido, sólido y flexible. Es idóneo para la creación de bases de datos con acceso desde páginas web dinámicas, así como para la creación de cualquier otra solución que implique el almacenamiento de datos, posibilitando realizar múltiples y rápidas consultas. Está desarrollado en C y C++[16], facilitando su integración en otras aplicaciones desarrolladas también en esos lenguajes.

Es un sistema cliente/servidor, por lo que permite trabajar como servidor multiusuario y de subprocesamiento múltiple, o sea, cada vez que se crea una conexión con el servidor, el programa servidor establece un proceso para manejar la solicitud del cliente, controlando así el acceso simultáneo de un gran número de usuarios a los datos y asegurando el acceso a usuarios autorizados solamente. Es uno de los sistemas gestores de bases de datos más utilizado en la actualidad [29].

[16] Es un lenguaje de programación orientado a objetos que toma la base del lenguaje C y le agrega la capacidad de abstraer tipos.

CAPÍTULO III

ANÁLISIS Y DISEÑO

A partir de la metodología RUP aplicada para el ciclo de vida de desarrollo de este sistema y en base a los objetivos definidos con anterioridad, se procede a realizar el análisis y diseño del sistema. A continuación se describen los requisitos funcionales y no funcionales del presente proyecto.

3.1 Requisitos funcionales

El sistema web debe incluir un control de acceso por niveles, que permita a los usuarios ingresar con un nombre de usuario y una contraseña, los cuales serán gestionados por el administrador del sistema.

El sistema permitirá al usuario del nivel uno que es el administrador, agregar, editar, eliminar y actualizar los registros de asistentes al evento, personal de apoyo, eventos y registros de usuarios, así como visualizar estadísticas. Mientras que el usuario del nivel dos denominado como monitor podrá ver estadísticas del evento para la toma de decisiones, de igual manera, será capaz de registrar la asistencia de usuarios. El usuario de nivel 3, que hace referencia al personal de apoyo podrá únicamente registrar la asistencia de usuarios al evento.

3.2 Requisitos no funcionales

3.2.1 Del Producto

Usabilidad:

RN-1 El sistema está preparado para ser operado a través de mouse y teclado.

RN-2 Las pantallas serán desarrolladas para un entorno web y adaptativo a todos los dispositivos.

RN-3 Todas las pantallas deben tener un modo de cancelar la operación en curso.

RN-4 Desplegar mensajes de error y advertencia intuitivos.

RN-5 Verificar/validar límites de campos y tipos de datos de las pantallas en relación al modelo de datos.

RN-6 La consulta Web debe ser accedida desde cualquier navegador.

Confiabilidad:

RN-7 Implementar mecanismos que aseguren la integridad de los datos.

RN-8 Se debe asegurar la disponibilidad del Sistema 24 X 7 X 365

Performance:

RN-9 Se espera que el tiempo de respuesta en el momento de presionar un botón para continuar con el flujo de la información que no supere los 20 segundos.

RN-10 Se espera mantener la escalabilidad del sistema en relación a la concurrencia de usuarios (cantidad de usuarios entre 15 y 40 concurrentes)

RN-11 El sistema deberá liberar a todos los recursos de memoria al momento de cerrar una ventana y finalizar una funcionalidad.

Soportabilidad:

RN-12 El control de integridad de datos se hará del lado de la capa de datos (a nivel de la base de datos utilizando las claves foráneas). Los mensajes de error serán capturados por la aplicación y serán visualizados al usuario final.

RN-13 Implementar Reglas de Negocio y procesos de auditoria a nivel de la capa de datos (a nivel de la base de datos utilizando desencadenadores)

Documentación:

RN-14 Correcta redacción y ortografía en las pantallas.

RN-15 Uso estandarizado de pantallas, mensajes y estilos.

3.2.2 Del Ambiente

Ético:

RN-16 El sistema debe garantizar la confidencialidad de la información de los Clientes y de los valores negociados con el Cliente.

Legales:

RN-17 Se debe cumplir lo establecido en los Contratos de Licencia MIT y uso del repositorio de GitHub.

3.3 Análisis del Sistema

A partir de la descripción del sistema especificada en los requerimientos funcionales y no funcionales, se define el comportamiento mediante el modelo de casos de uso.

3.3.1 Modelo de Casos de Uso

3.3.1.1 Diagramas de Casos de Uso

En la figura 6 se muestra el comportamiento del sistema, es decir la funcionalidad del sistema ante la intervención de los actores, mediante el diagrama de casos de uso general.

3.3.1.2 Diagramas de Casos de Uso Específico

Para cada caso de uso existen especificaciones que detallan con precisión el comportamiento de los mismos, para ello se procede a especificar en los principales casos de uso.

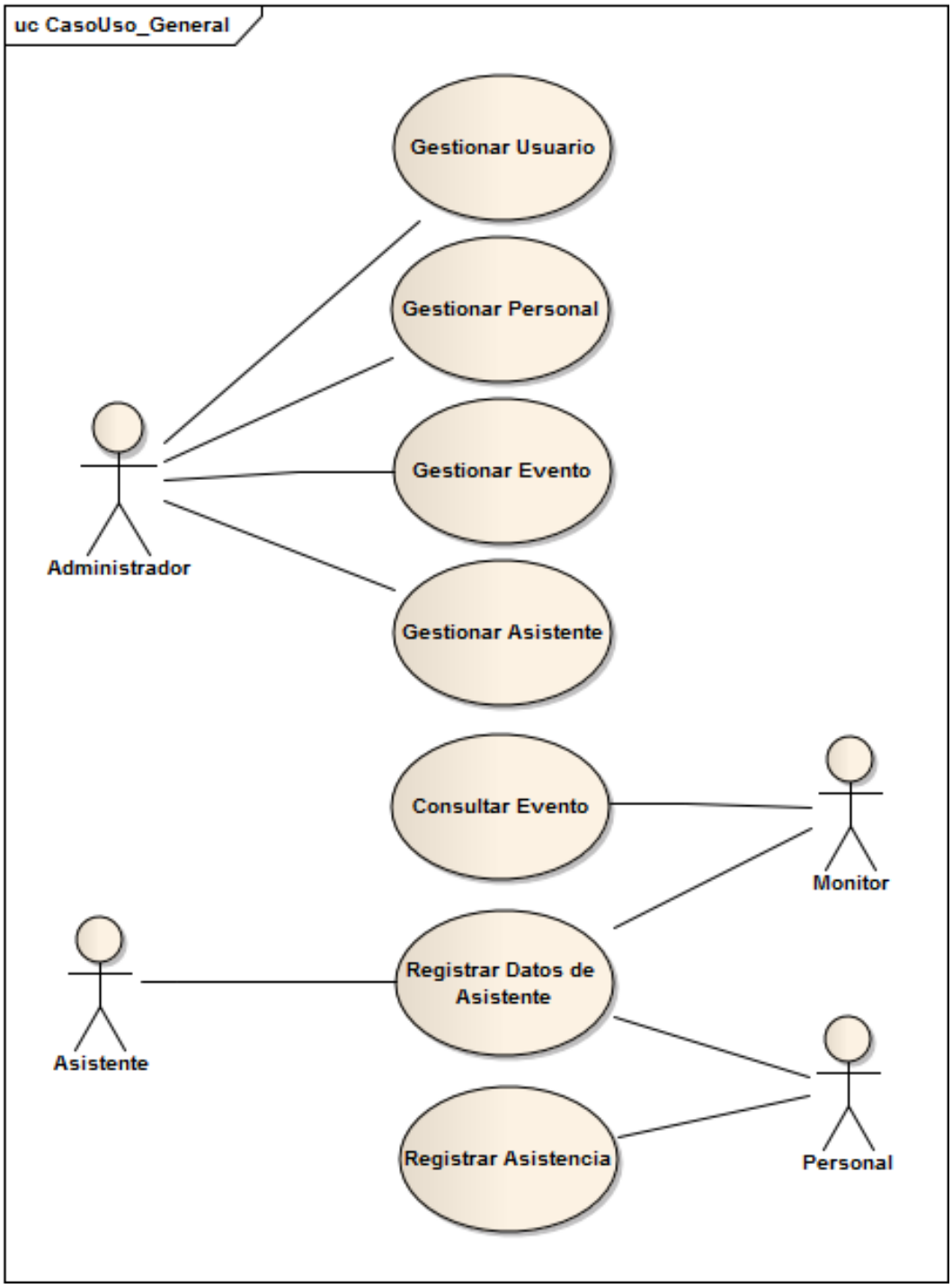

Figura 6. Diagrama general de casos de uso.

A) Caso De Uso: Gestionar Usuario

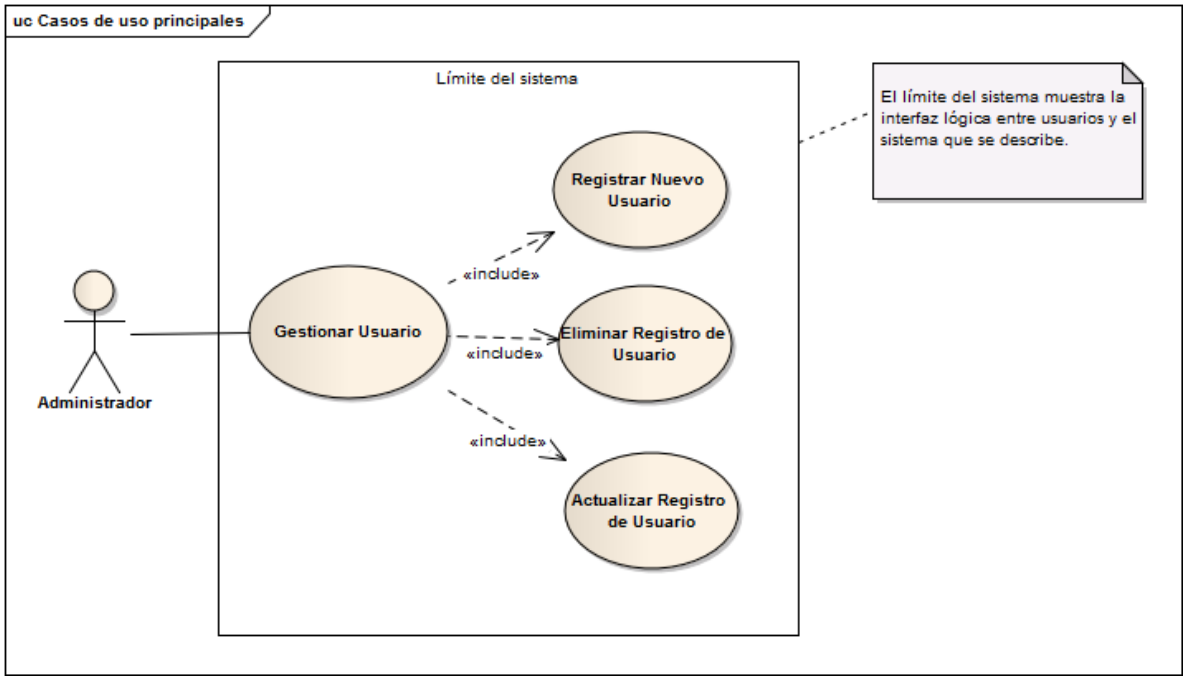

Figura 7. Diagrama de Caso de Uso Específico - Gestionar Usuario.

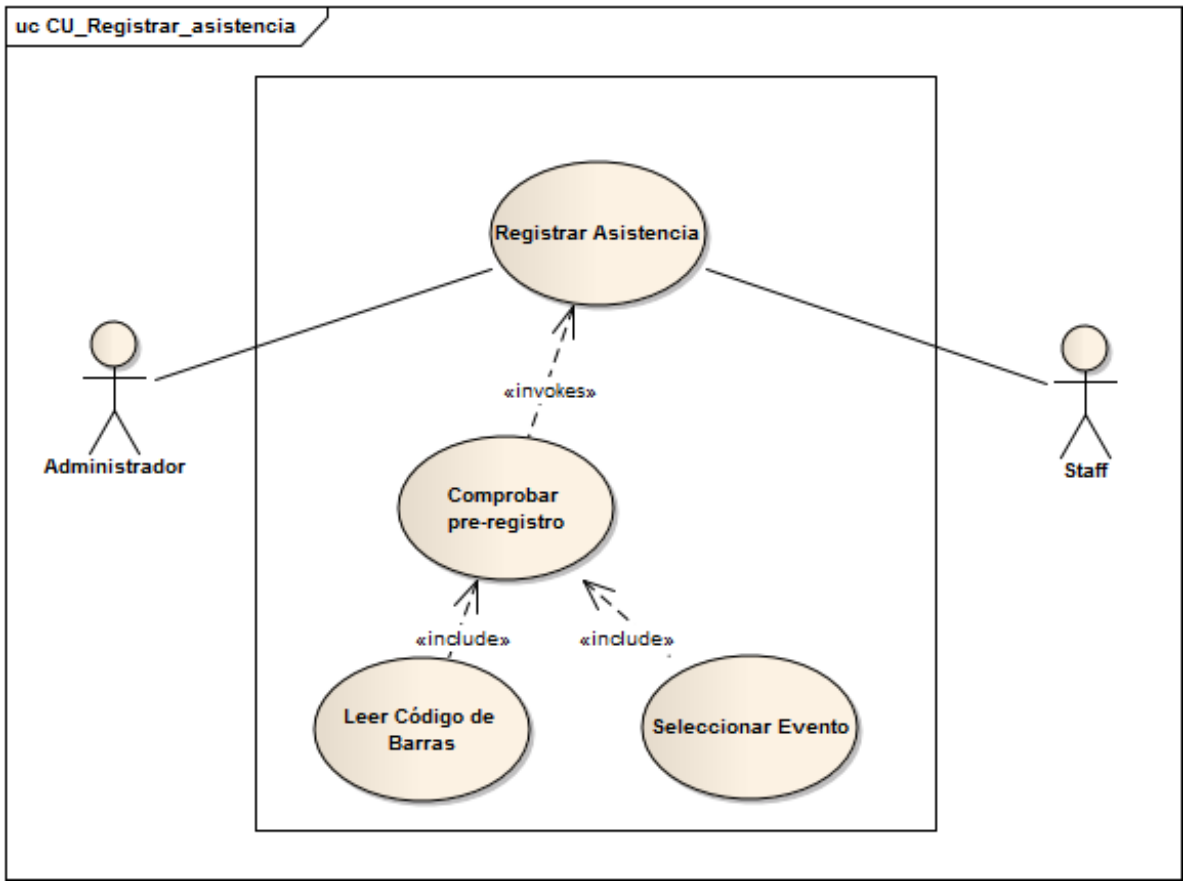

Figura 8. Diagrama de Caso de Uso Específico - Registrar Asistencia.

3.3.1.3 Especificación de Caso de Uso

Un caso de uso se debe documentar para describir el propósito que tiene y el flujo de eventos que realiza, así como para tener en cuenta las pre/post condiciones que se deben cumplir para su funcionamiento, a continuación se lista la documentación de los casos de uso del diagrama anterior: [5]

1. Nombre:	Gestionar Usuario
2. Autor:	María Martha Quintana López
3. Fecha:	07-Abril-2018
4. Descripción: En este caso de uso, el usuario es capaz de crear, modificar y eliminar a los usuarios que interactúan con el sistema.	

5. **Actores:** Administrador.
6. **Pre-Condiciones:** El usuario debe iniciar sesión como Administrador.
7. **Flujo básico:**

7.1 Para ***Registrar un nuevo usuario***, el usuario debe llenar los campos: Nivel, Correo, Contraseña.

7.1.1 Hacer clic en Guardar.

7.1.2 Se valida que no haya un registro igual.

7.1.3 Muestra un cuadro de diálogo con una leyenda de la acción realizada, el usuario hace clic en Aceptar.

7.1.3 Se da de alta al personal en la base de datos.

7.2. Para ***Eliminar un registro de usuario***, el usuario debe seleccionar el nombre del usuario a eliminar.

7.2.1 Hacer clic en Eliminar.

7.2.2 Muestra un cuadro de diálogo con una leyenda para confirmar la acción o revertirla.

7.2.3 El usuario confirma que desea eliminar haciendo clic en Aceptar.

7.2.4 Se elimina al personal de la Base de Datos.

7.3. Para ***Actualizar un registro de usuario***, el usuario selecciona el asistente a Editar.

7.3.1 Hacer clic en Editar.

7.3.2 Se habilitan los campos de texto para su edición y con ello dos botones: Guardar y cancelar.

7.3.3 El usuario da clic en Guardar.

7.3.4 Se actualiza al personal en la Base de Datos.

8. Flujos Alternativos

8.1 Para **Registrar un nuevo usuario**, si al terminar el flujo 7.1.2 el usuario no realiza la acción del flujo 7.1.3. Se muestra un cuadro de diálogo con la leyenda acción no realizada y se muestra de acuerdo a la validación la razón y regresa.

8.2 Para **Eliminar un registro de usuario**, si el usuario da clic en Revertir la acción del cuadro de diálogo del flujo 7.2.3 se interrumpe el flujo y el registro del personal no es eliminado.

8.3 Para **Actualizar un registro de usuario**, si el usuario da clic en el botón Cancelar del flujo 7.3.2, se deshabilitan los campos y el registro del personal no es actualizado.

9. Flujo de excepción

9.1.E.1. El sistema no puede guardar la información porque no hay conexión a la BD.

9.2.E.2. El sistema no puede guardar la información porque ya existe.

9.3.E.3. El sistema no puede guardar porque no se ingresó ninguna información.

9.4.E.4. No existe información con los criterios de búsqueda indicados.

1. Nombre:	Registrar Asistencia
2. Autor:	María Martha Quintana López
3. Fecha:	07-Abril-2018
4. Descripción: Confirma la asistencia, registra la hora y el tipo de evento (si aplica) de los convocados, mediante un código de barras asignado al momento de su pre registro en línea.	

5. **Actores:** Administrador, Personal de apoyo.
6. **Pre-Condiciones:**
   6.1 El usuario debe ser un Administrador o Personal.
   6.2 El Usuario debe registrar sus datos previamente.
7. **Flujo básico:**

   7.1 Pasar el lector del código de barras por el registro impreso.
   7.2 Seleccionar el Tipo de Evento.
   7.3 Dar clic en el botón Registrar Asistencia.
   7.4 Desplegar un mensaje de confirmación y las opciones Aceptar y Cancelar.
   7.5 El usuario da clic en Aceptar.
   7.6 Se registra al Asistente correspondiente al evento en la

base de datos.

8. **Flujos alternativos:**

8.1. Si al terminar el flujo 3 no cumple con los requerimientos de validación, muestra una leyenda correspondiente en cada campo y se reanuda en el flujo 7.3, hasta que cumpla las condiciones.
8.2. Si en el flujo 4 el usuario da clic en Cancelar, se reanuda en el flujo 3.

9. **Flujo de excepción:**

9.1.E.1. El sistema no puede guardar la información porque no hay conexión a la BD.

9.2.E.2. El sistema no puede guardar la información porque ya existe.

9.3.E.3. El sistema no puede guardar porque no se ingresó ninguna información.

3.3.2 Modelo Conceptual

3.3.2.1 Diagrama Conceptual

Como primera propuesta se tiene el siguiente Modelo Conceptual (Figura 9) para cuáles son las asociaciones pertinentes entre los conceptos más relevantes para el sistema web. En este punto del análisis se aplican los conceptos del paradigma orientado a objetos, que nos ayuda a identificar los grupos de objetos y abstraerlos en clases.

A partir de la descripción de los requerimientos del sistema, es posible definir la estructura, se comienza examinando dicha descripción y se construye un primer modelo para identificar las clases que integran el sistema y sus relaciones.

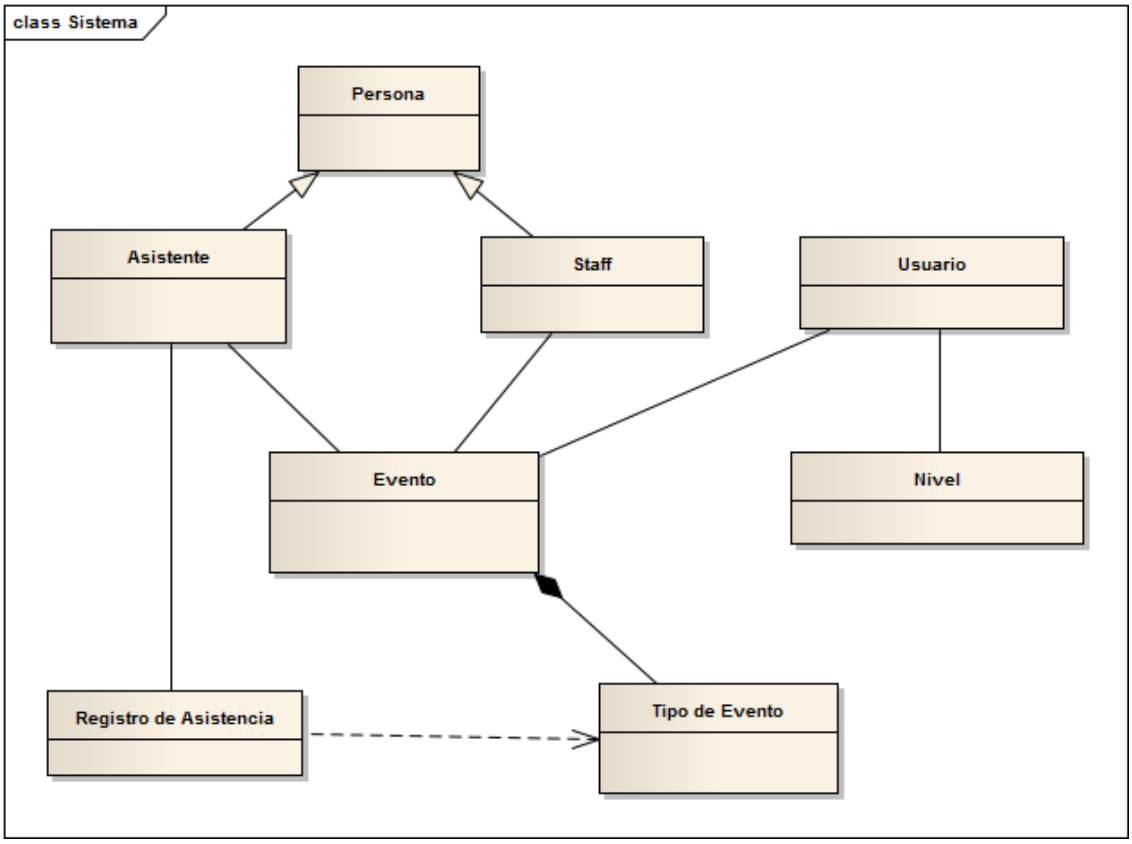

Figura 9. Diagrama Conceptual del Sistema Web.

3.3.2.2 Diccionario del Modelo

A continuación se describen las clases del modelo anterior a través de éste diccionario.

Persona: Datos de contacto y particulares de que identifican generalmente a una persona.

Usuario: Información de la persona con conocimiento en el manejo del sistema con una previa capacitación por parte de los manuales en base a los criterios de administración del mismo, dependiendo del nivel que le corresponda.

Nivel: Define el acceso a las tareas que un usuario desempeña en el sistema, así como restringir o autorizar el acceso a vistas.

Staff: Persona encargada en el área de Registro que deberá tener conocimiento de la interfaz web de Registro de Asistencia y noción del uso de lector de código de barra, también desempeña una actividad en algún *tipo de evento*.

Evento: Es el acontecimiento de interés particular al que son convocados los asistentes, información acerca de su realización.

Tipo de Evento: Es la actividad que se desempeña dentro del evento, pueden ser una o varias durante la realización del mismo.

Asistente: Persona que interactuará continuamente con el sistema, su educación no puede ser definida y no debe tener una capacitación con respecto al sistema únicamente debe tener un conocimiento básico sobre navegación en páginas Web.

Registro de Asistencia: Mantiene un control de los asistentes que acuden a un evento, guardando un registro de su entrada al mismo.

3.4 Diseño del Sistema

3.4.1 Diagrama de Componentes

Como puede observarse en la figura 10, encontramos componentes de software principales a implementar en el sistema. En éste diagrama simplificado se muestran todos los modelos de la aplicación, así como una simplificación de sus controladores y de las vistas que los muestran en pantalla.

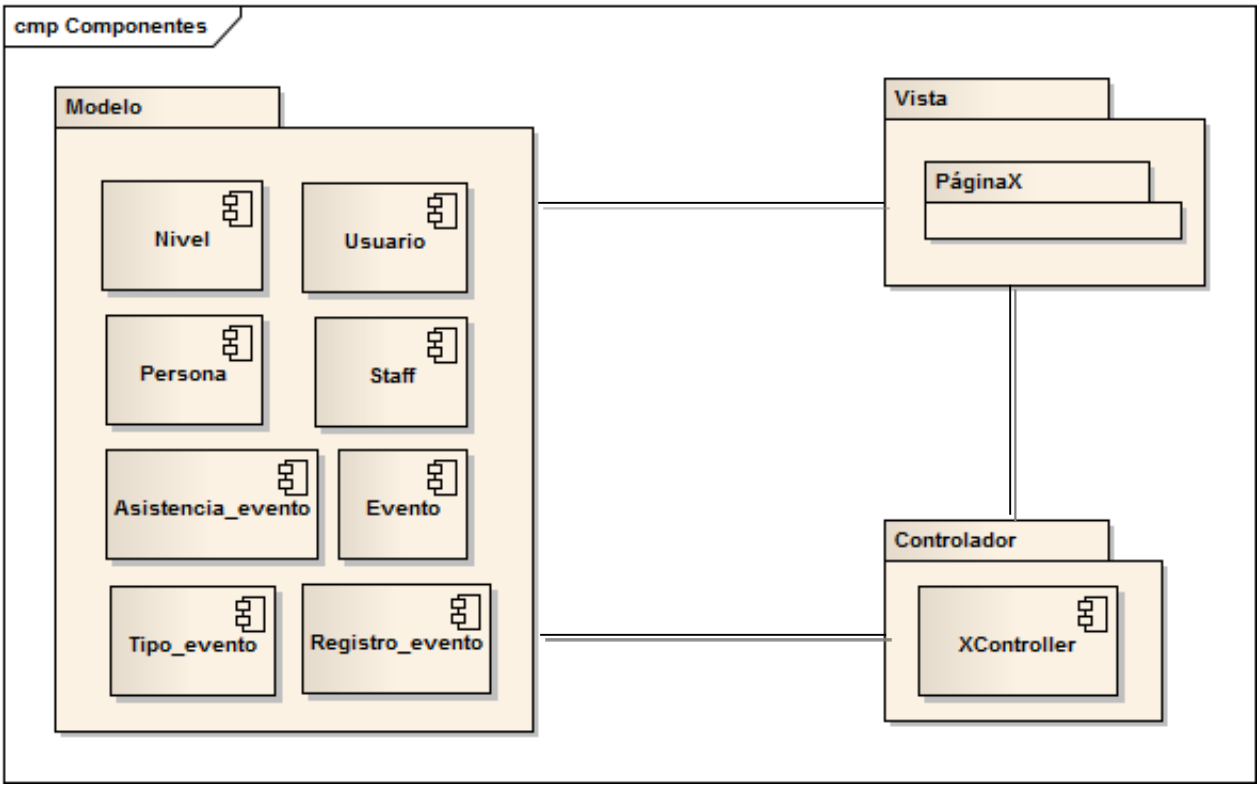

Figura 10. Diagrama de Componentes simplificado del Sistema Web.

3.4.2 Diagrama de Clases

El diagrama de clases (figura 11) muestra las clases incorporadas. Dado que Laravel propone el uso de la arquitectura MVC, en el diagrama se han incluido únicamente las clases referentes al tratamiento de datos en la aplicación, es decir los modelos y controladores.

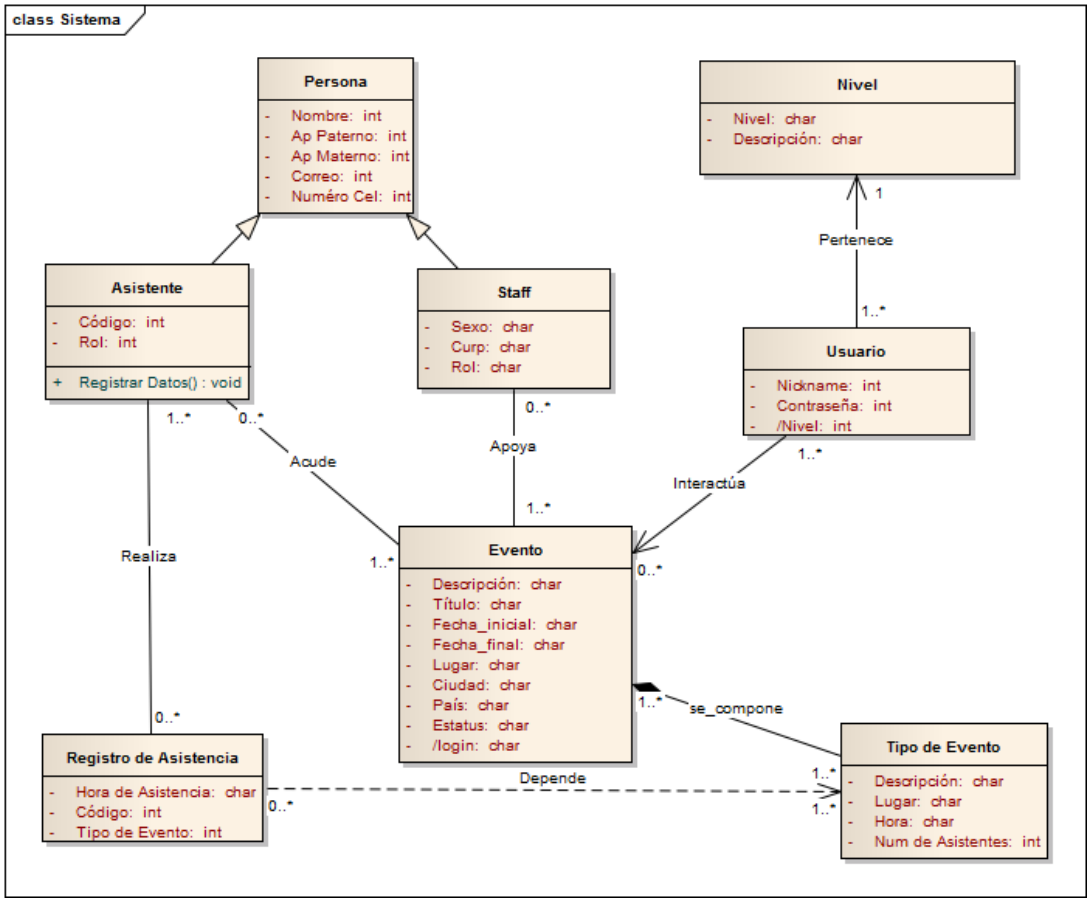

Figura 11. Diagrama de Clases del Sistema Web.

3.4.3 Diagrama de Secuencia

Los diagramas de secuencia facilitan la descripción de la interacción entre los componentes principales del sistema para cumplir el objetivo de cada caso de uso. Gracias a éstos diagramas podemos ver fácilmente cómo se distribuyen las tareas entre componentes e identificar patrones de iteracción.

El siguiente diagrama (figura 12) corresponde a una petición realizada en Laravel.

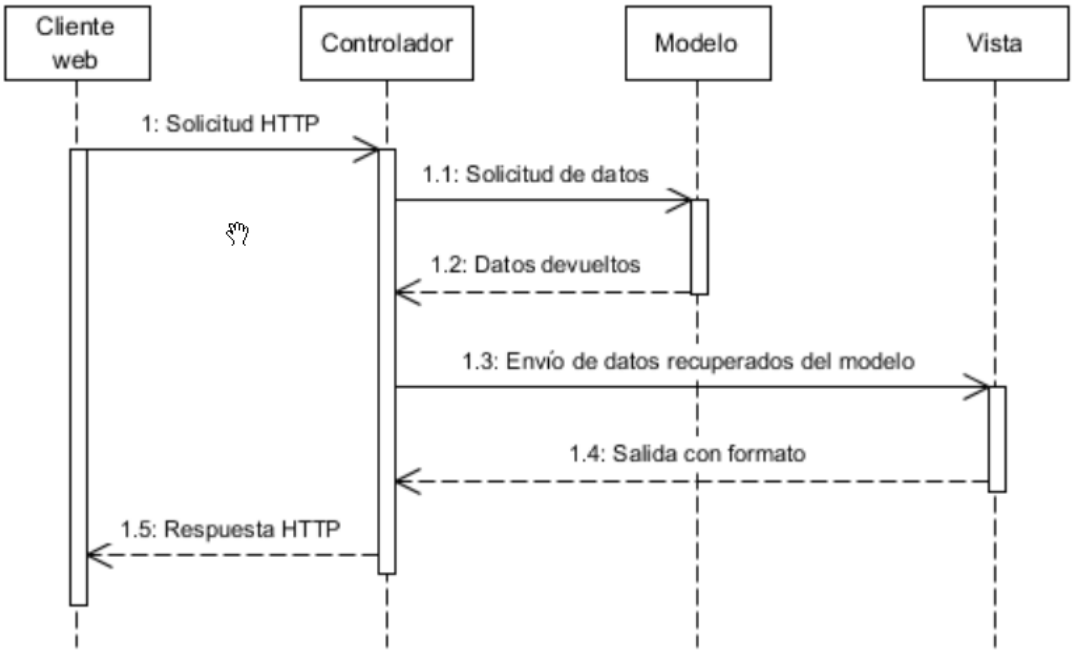

Figura 12. Diagrama de Secuencia de una petición en Laravel.

Para los casos de Uso, se realizaron los siguientes diagramas de secuencia.

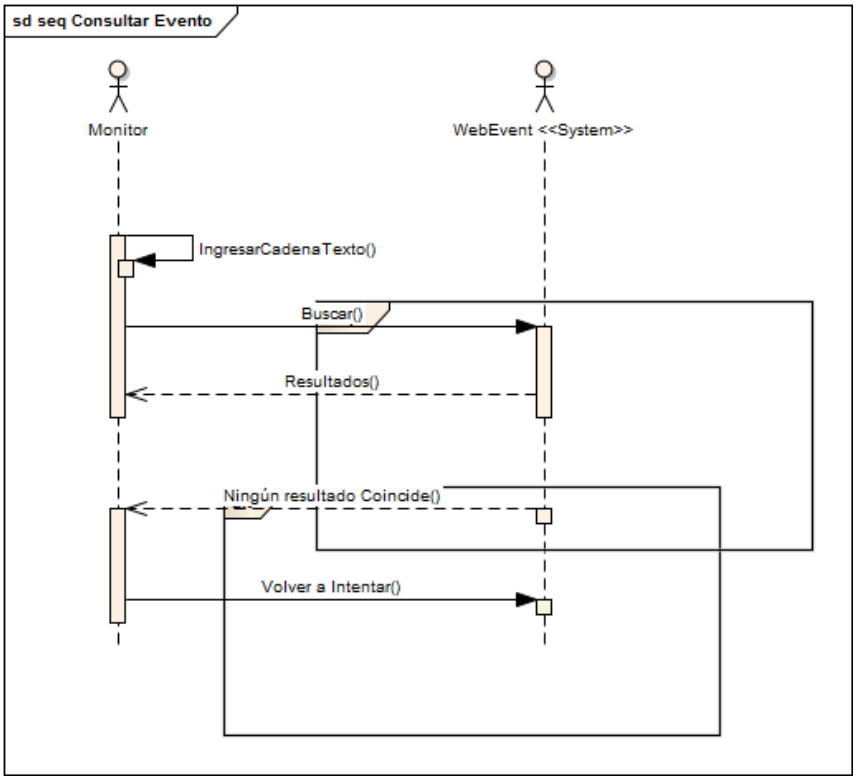

Figura 13. Diagrama de Secuencia – Caso de Uso Consultar Evento.

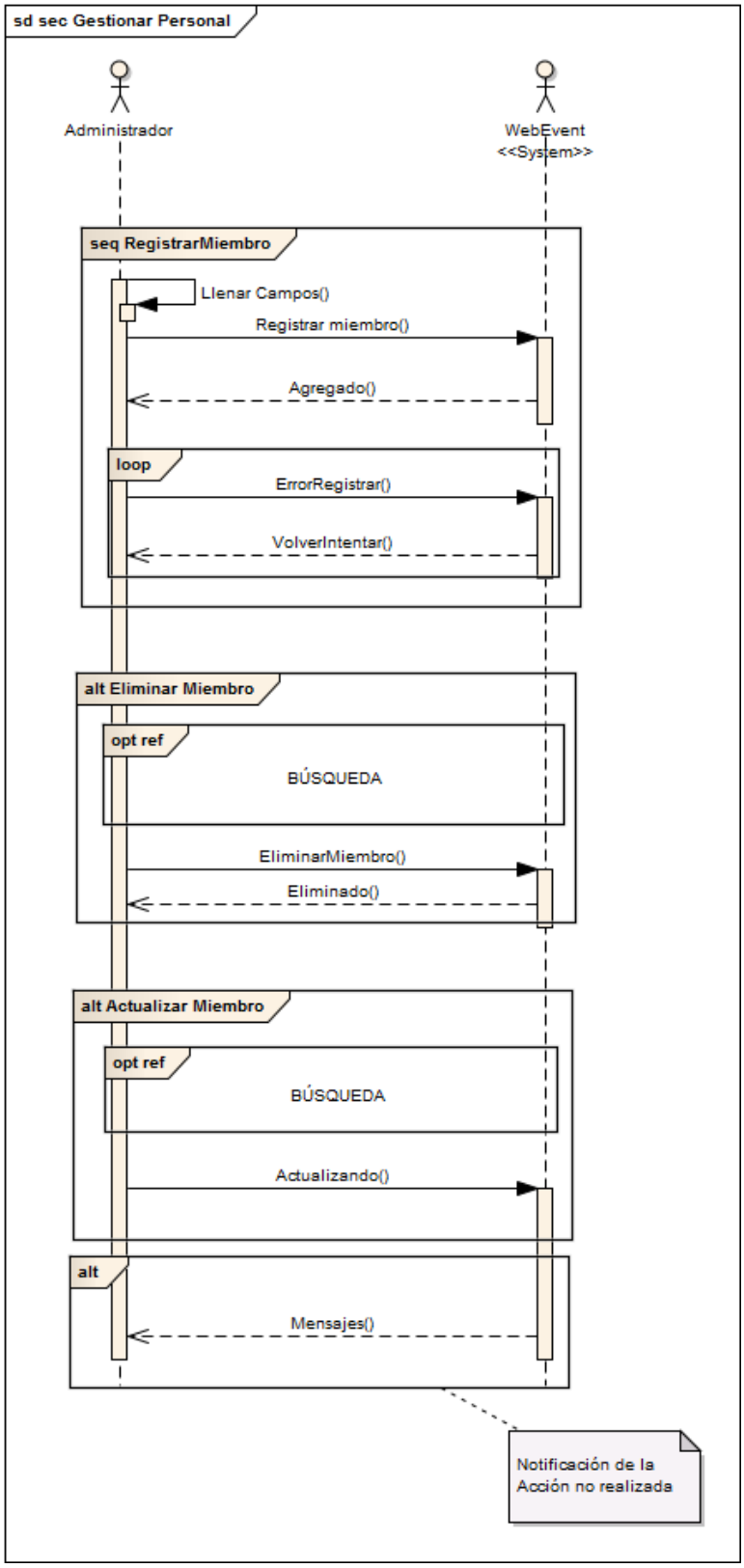

Figura 14. Diagrama de Secuencia – Caso de Uso Gestionar Personal.

3.4.4 Diagrama de Actividades

La vista de proceso define los aspectos del diseño del software que están relacionados con la concurrencia y sincronización de la ejecución. Representaremos ésta vista mediante un diagrama de actividades.

La arquitectura de procesos tiene en cuenta requisitos no funcionales como el rendimiento y la disponibilidad. Puede ser descrita en varios niveles de abstracción, dependiendo de lo que nos interese definir. Al más alto nivel, esta vista puede verse como un conjunto de programas que se comunican mediante redes lógicas.

Como puede observarse en el diagrama (figura 15), distinguimos la estructura principal de una arquitectura MVC clásica, a la cual se le han añadido funcionalidades extra propias de Laravel.

Si describimos el camino que toma una solicitud a través del sistema, observamos que la petición es recogida por el módulo de **Routing,** el cual la dirigirá al controlador adecuado dependiendo de su tipo. Antes de llegar al controlador, existe un módulo intermedio, denominado **Middleware**, que realizará ciertas acciones sobre los datos de la petición. Este middleware comprueba aspectos como la autenticación de las peticiones (comprobando la validez del token de usuario), la limpieza de los datos introducidos, redirecciones y otras variables en la ejecución.

Una vez que la petición pasa a ser gestionada por su **Controlador** correspondiente, es este último el que debe recabar la información necesaria y procesarla para producir la salida esperada por el usuario. En Laravel, el controlador realiza una llamada a un proveedor de servicios (Indicado como **Servicio** en el diagrama) que se accederá al **Repositorio** de entidades que se han introducido en el sistema. De esta manera, éste repositorio accederá al modelo concreto que ha sido

solicitado cargando todos los datos del mismo en memoria (normalmente desde BBDD o una API) y generando una carga que será devuelta de forma recursiva al controlador.

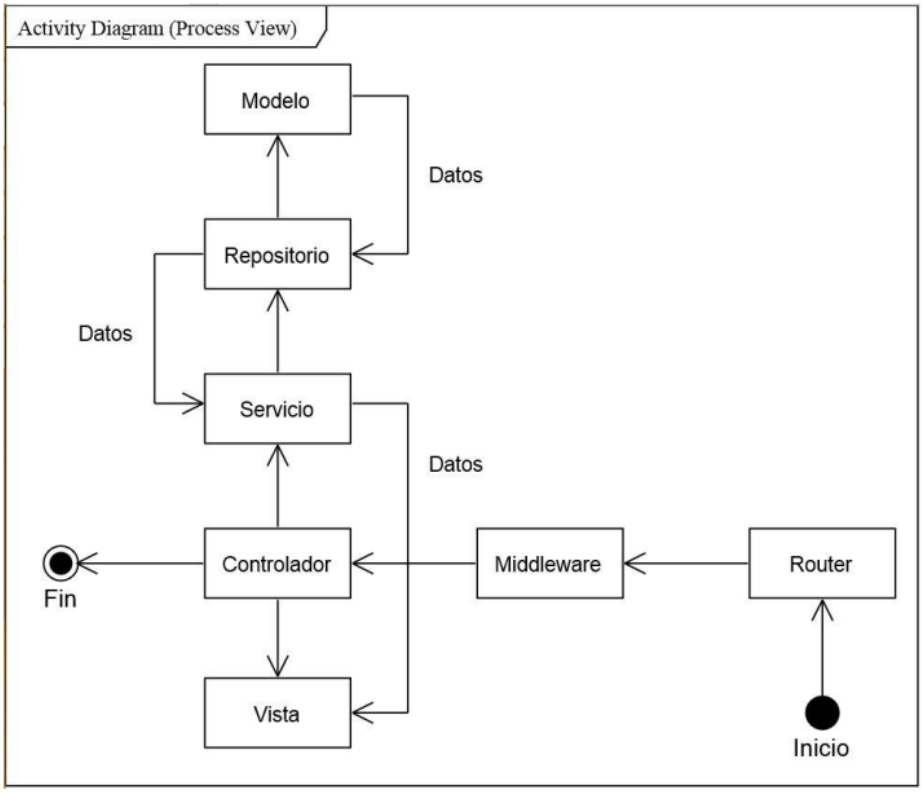

Figura 15. Diagrama de actividades (Vista de proceso).

Obtenidos todos los datos necesarios del modelo, el controlador operará con los datos de la manera que queda descrita en sus métodos, cumpliendo así la petición. El resultado será mostrado en la **Vista**, compuesta por una página web enriquecida que integra esos datos de una forma coherente para ser comprendidos por el usuario, finalizando así el ciclo de la petición.

Es necesario mencionar que el ciclo anteriormente descrito es el correspondiente a una solicitud de página de un cliente web. Sin embargo, el envío de datos por parte del cliente sugeriría un esquema muy similar al descrito, añadiendo una carga de datos que el controlador deberá gestionar

de acuerdo a las especificaciones del modelo. Las peticiones generadas por el propio sistema (como puede ser a través de notificaciones push) no están completadas por éste esquema.

3.4.5 Diseño de Prototipo

A continuación se muestran los prototipos realizados con el software *Enterprice Architect®.* Corresponden a la interfaz del Administrador.

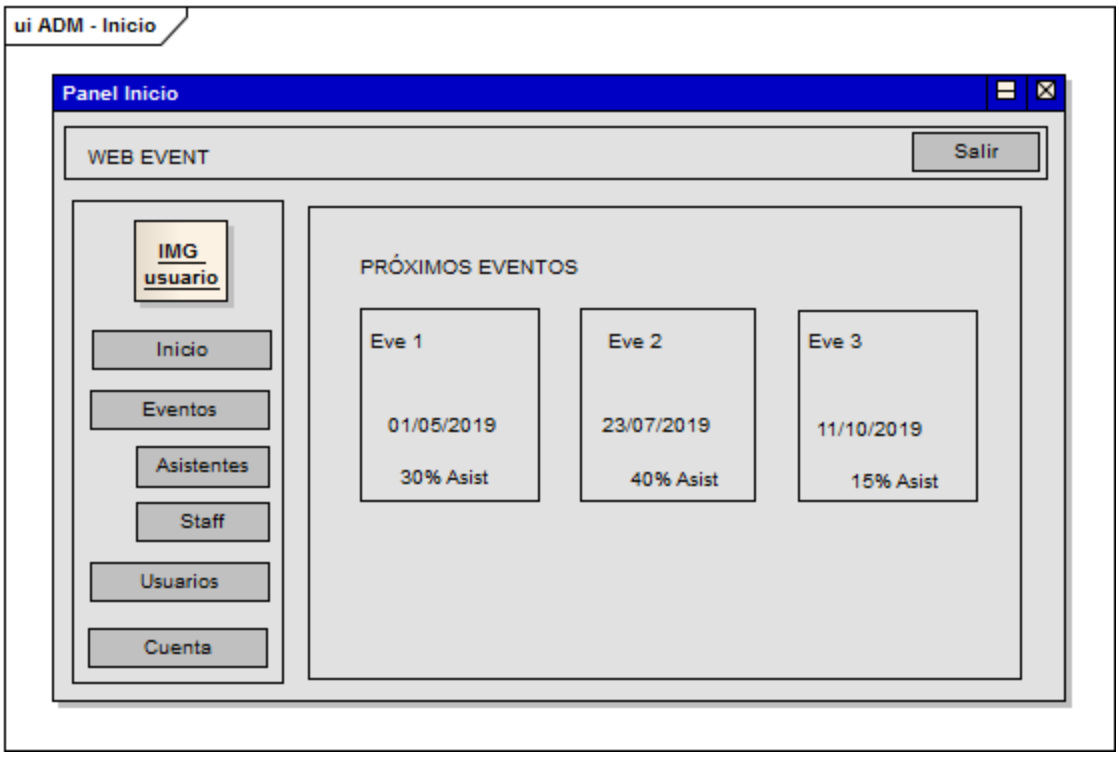

Figura 16. Prototipo- Inicio – Interfaz administrador.

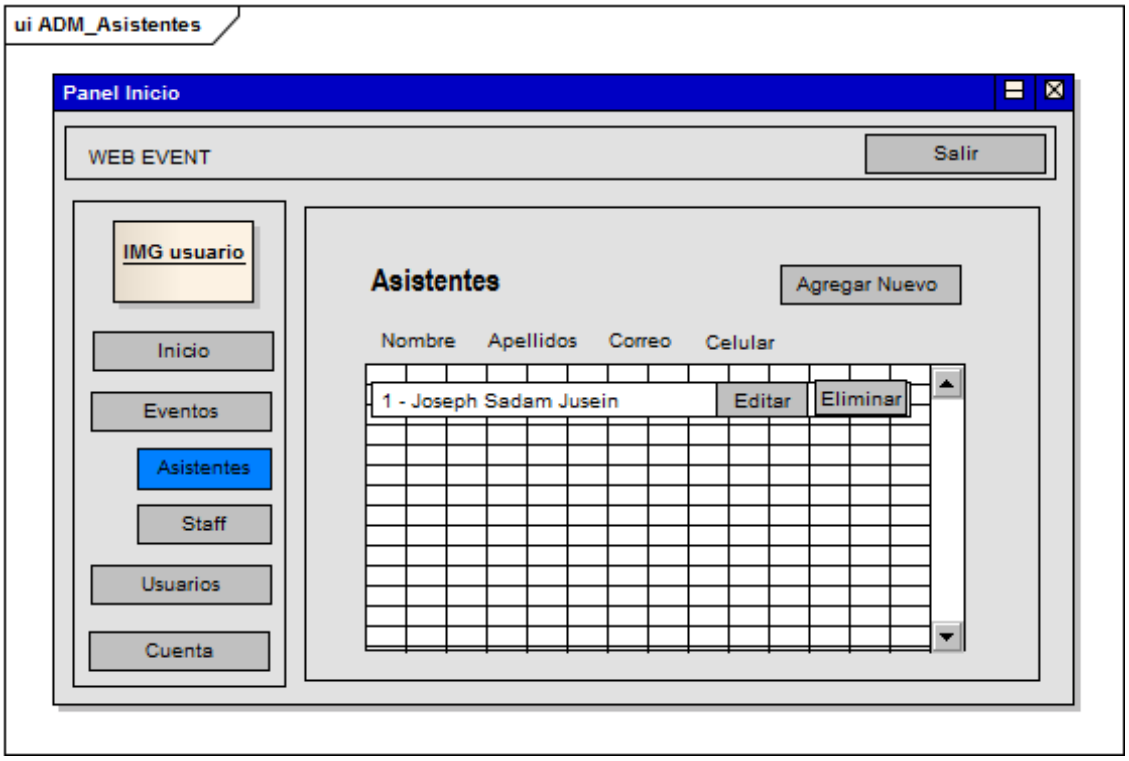

Figura 17. Prototipo- Inicio – Interfaz administrador.

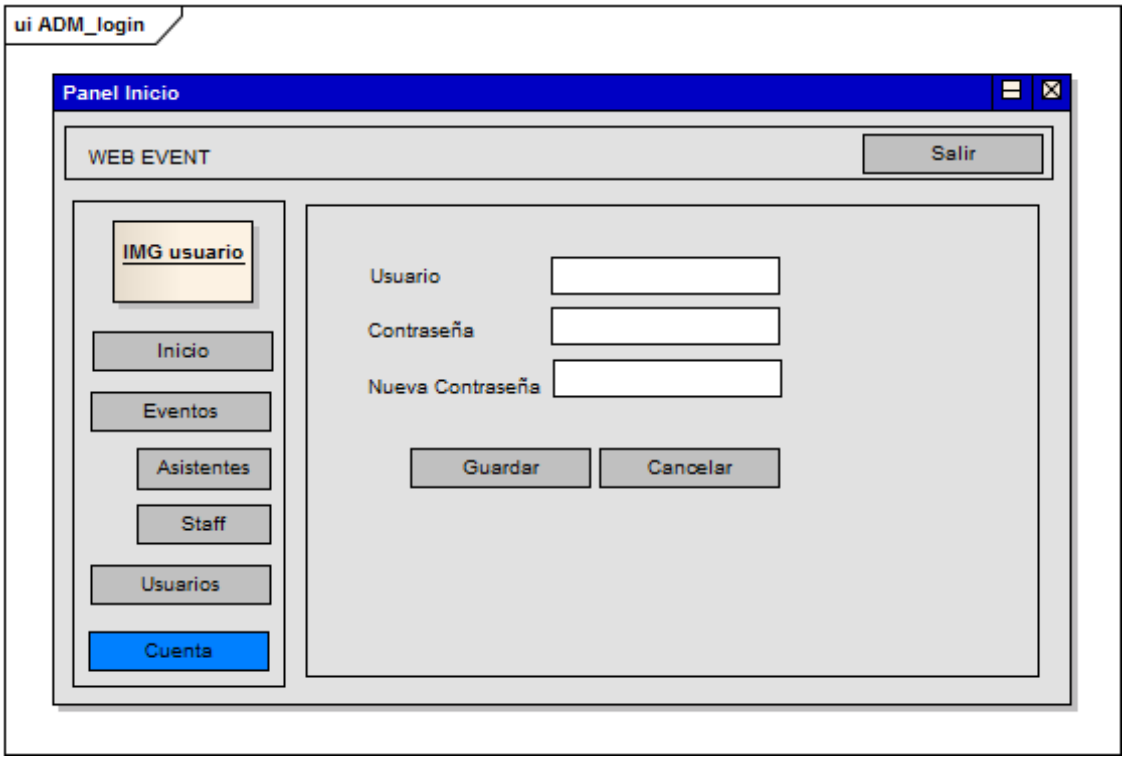

Figura 18. Prototipo- Login – Interfaz general.

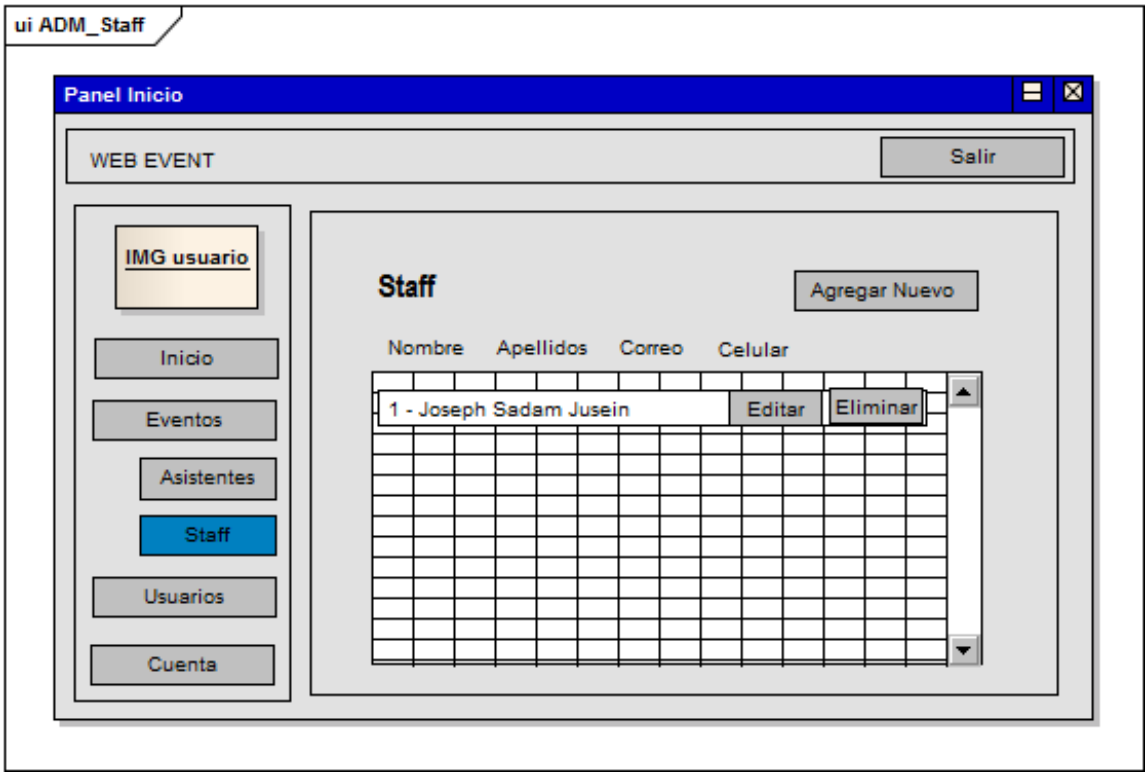

Figura 19. Prototipo - Staff – Interfaz administrador.

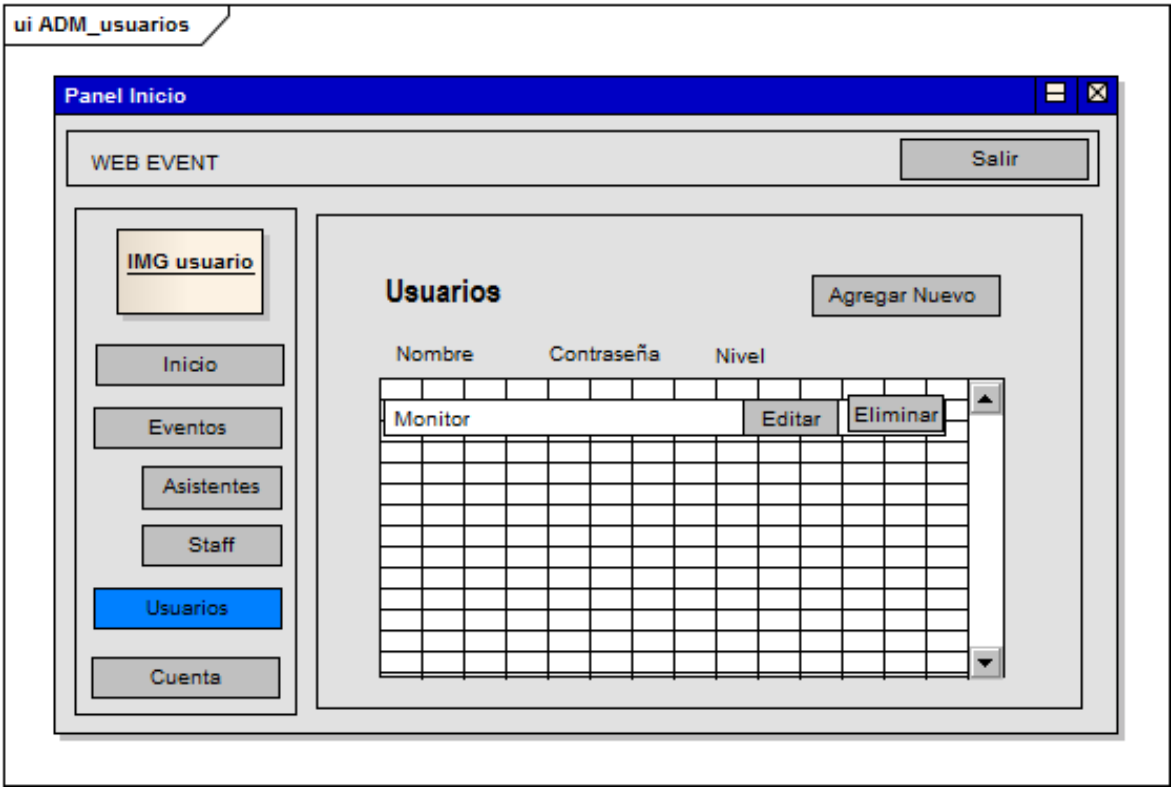

Figura 20. Prototipo- Usuarios – Interfaz administrador.

3.4.5 Diagrama de Despliegue

A continuación se muestra un diagrama de despliegue que modela, a alto nivel, la distribución de las piezas de software del sistema sobre los elementos de hardware que se usarán para ejecutarla, indicando las asociaciones entre nodos con caminos de comunicación, que indican la tecnología requerida para que ésta se lleve a cabo exitosamente.

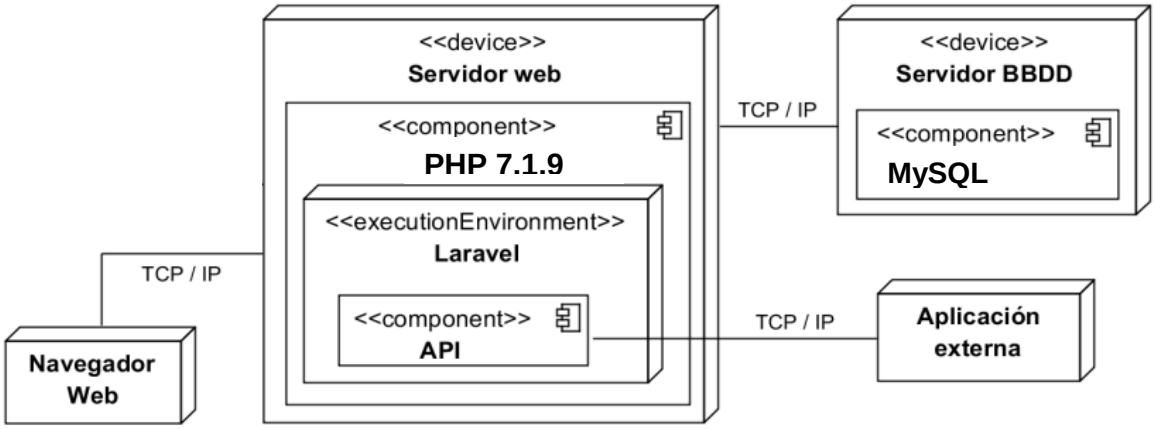

Figura 21. Diagrama de despliegue (Vista Física).

- **Servidor Web**

  Durante la fase de desarrollo, el software utilizado ha sido el servidor interno de PHP. Su uso está desaconsejado para sistemas en producción por no tener las medidas de seguridad y rendimiento necesarias, pero facilita la creación rápida de un entorno de desarrollo completamente funcional. Debido a esto para la puesta en producción se propone el uso de Apache HTTP Server 2.4.27, que será el que reciba las peticiones del cliente y devuelva la respuesta.

- **PHP 7.1.9**

  El procesador PHP, sobre el que se ejecuta el Sistema Web, es el subsistema base que permite ejecutar todas las instrucciones del programa. En éste proyecto, el servidor web es un intermediario que gestionas las conexiones entre cliente y servidor, mientras que el

ejecutable y bibliotecas de PHP (incorporado como un módulo de Apache) son los responsables de ejecutar la aplicación.

- **Laravel**
  Es el framework que conforma la base sobre la que se construye el sistema. Facilita y acelera el desarrollo mediante funciones ya implementadas, permitiendo al desarrollador centrarse en el *core* de la aplicación y enfocarla al negocio. Provee además una plataforma estandarizada a través de la cual pueden colaborar distintos individuos guiados por una documentación común.

- **API**
  La interfaz de programación de aplicaciones permite la comunicación entre distintas aplicaciones, incluso si se están ejecutando en distintos sistemas. La API REST definida para la aplicación es usada internamente para proveer los servicios accedidos desde un navegador web, pero también puede ser usada para compartir información hacia otros dispositivos, como aplicaciones móviles o servidores.

- **Aplicación externa**
  Es el software que consume o provee datos a través de la API del sistema. Lo más común es que esto se realice a través del protocolo TCP/IP. La aplicación externa deberá disponer de los mecanismos necesarios para interpretar la información obtenida.

- **Navegador web**
  Sirve de ventana entre el usuario y la aplicación construida. El sistema web ha sido diseñado para ser ejecutado en un servidor web, por lo que los clientes que se conecten al mismo utilizarán un navegador como primera herramienta para utilizarlo. Cabe mencionar que el navegador web, aunque es una aplicación externa, no consumiría datos de la misma manera que un cliente a través de su API (éste último puede carecer de una interfaz gráfica de usuario).

- **Servidor BBDD**

Conforma otro entorno físico sobre el que ejecutar un Sistema de Gestión de Base de Datos (SGBD). Se comunicará con el servidor web a través del protocolo TCP/IP. Este servicio podría ser provisto desde el mismo hardware que el servidor web, algo que es habitual en aplicaciones a pequeña escala.

- **MySQL 5.0.12**

  Es el SGBD que se ha elegido para el desarrollo y puesta en producción. Sirve como el almacén de información principal de la aplicación, mediante el cual se pueden obtener los datos necesarios para gestionar las peticiones que llegan al sistema.

CAPÍTULO IV

DISEÑO DE LA BASE DE DATOS

El modelo de datos son abstracciones que permiten la implementación de un sistema eficiente de base de datos. Así en el diseño de esta base de esta base de datos se necesita definir la estructura de los datos, así como las relaciones que existen entre estos.

Para lograr esto lo primero a analizar es el diseño conceptual y se hace a continuación.

4.1 Diseño conceptual

Este modelo es utilizado para representar la realidad a un alto nivel de abstracción. Así que, lo que se va a hacer será construir una descripción de la realidad fácil de entender.

Para expresar los resultados del diseño conceptual se utilizará el modelo entidad-relación y en particular para este proyecto de tesis se usa la notación estándar, misma que está disponible en el editor de diagramas DIA.

4.1.1 Modelo entidad-relación

El modelo de datos entidad-relación está basado en una percepción del mundo real que consta de un conjunto de objetos básicos llamados entidades y de relaciones entre estos objetos.

4.1.1.1 Entidades

Una entidad es una “cosa” u “objeto” en el mundo real que podemos distinguir de otros objetos y del que nos interesan sus propiedades.

Teniendo esto en cuenta y considerando que previamente hemos realizado el análisis de requerimientos

donde obtuvimos las clases de análisis de tipo entidad, las cuales representan la información persistente en el sistema, tomaremos dichas clases como las entidades del presente modelo, ya que representan los objetos de información del mismo.

Dado lo anterior, las entidades del presente modelo son las siguientes:

- NIVEL
- USUARIO
- PERSONA
- ASISTENTE
- STAFF
- EVENTO
- TIPO DE EVENTO
- REGISTRO DE ASISTENCIA

4.1.1.2 Atributos

Un atributo es una característica o propiedad de una entidad, conocido como elemento de datos (valor específico) para cada uno de sus atributos que se encuentran en los campos de un registro que describen a una entidad y así será posible su identificación única.

Aplicando la teoría y los conceptos del modelo para encontrar los atributos de las entidades anteriores, los atributos quedarían de la siguiente manera:

- NIVEL (Nivel, Descripción)
- USUARIO (Nickname, Contraseña)
- PERSONA (Nombre, ApPaterno, ApMaterno, CORREO, NumCel)
- ASISTENTE (Código, Rol)

- STAFF (Sexo, Curp, Rol)
- EVENTO (Título, Descripción, FechaInicial, FechaFinal, Lugar, Ciudad, País, Estatus)
- TIPO DE EVENTO (Descripción, Lugar, Hora, NumDeAsistentes)
- REGISTRO DE ASISTENCIA (HoradeAsistencia, Código, TipoDeEvento)

4.1.1.3 Identificadores únicos

Un identificador de una entidad es un atributo o conjunto de atributos que determina de modo único cada ocurrencia de esa entidad.

En base a la definición anterior, se procede a analizar los atributos de las entidades para distinguir un identificador único o clave primaria, dado que no se han especificado en los requerimientos identificadores únicos, es como se asignará un identificador único para cada entidad, tal como se muestra a continuación:

- NIVEL (#nivel)
- USUARIO (#login)
- PERSONA (#id_persona)
- ASISTENTE (#id_registro)
- STAFF (#id_staff)
- EVENTO (#id_evento)
- TIPO DE EVENTO (#id_tipo)
- REGISTRO DE ASISTENCIA (#id_asistencia)

4.1.1.4 Relaciones

Una relación es una asociación entre varias entidades. Y en este proyecto representan las reglas y la información que el negocio necesita.

Así, en base a las entidades que ya se han establecido es importante establecer las relaciones que puedan existir entre las mismas y el tipo de relación. Las relaciones entre dichas entidades con el uso de la herramienta *GLIFFY* en su modalidad online se ilustran en la figura 22.

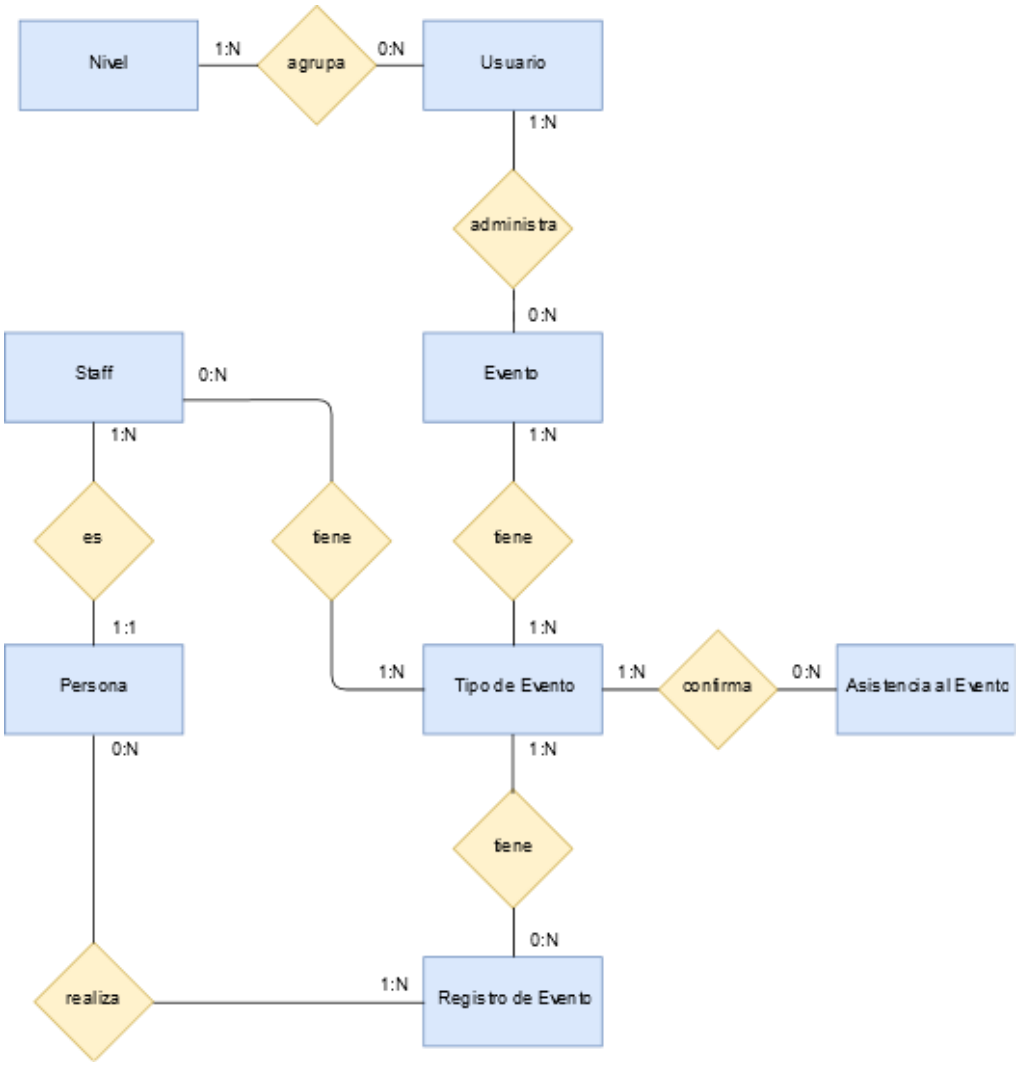

Figura 22. Diagrama del modelo Entidad –Relación.

4.1.1.5 Normalización

Ya que se han obtenido las entidades con sus respectivos atributos e identificadores únicos, es momento de normalizar dicha base de datos, para después poder llevarla a su implementación. Cabe destacar que para la normalización de esta base de datos solo se hará uso de las 3 primeras formas normales, ya que son consideradas como las que garantizan que la base de datos cumple con los requisitos naturales para funcionar óptimamente.

CAPÍTULO V

IMPLEMENTACIÓN Y PRUEBAS DEL SISTEMA

En base a los modelos y diagramas de las fases de análisis y diseño, y mediante las herramientas proporcionadas por la plataforma de implementación, se aplican los procesos indicados para cada una de estas, concibiendo así, el producto final, objetivo del proyecto.

Pertenecen a ésta etapa las pruebas a módulos y sistema en general, con el objetivo de verificar su correcto funcionamiento.

5.1 Implementación

En el marco de desarrollo de código abierto para el diseño de aplicaciones web adaptados a los dispositivos móviles, siendo Bootstrap el framework elegido para el presente proyecto, de decidió utilizar *AdminLTE* que es un panel de administración creado por *Almsaeed*, está basado en un diseño modular que permite una construcción y personalización sencillas. El objetivo es que cada uno de estos elementos sea un *plugin* o un *widget* a través del cual, se va creando la interfaz de usuario tanto por parte de *front-end* como de *back-end* del sistema.

5.1.1 Base de Datos

La transformación del diseño lógico de la BD a nuestro modelo físico, es mediante la elaboración de un script de instrucciones MySQL, que nos proporcionan la base de datos con todas las características diseñadas en el capítulo anterior.

Para ello se utilizó MySQL Workbench, en esta herramienta es posible llevar el diseño de la base de datos a la implementación. En primera instancia se diseñan las tablas, se conceptualiza el problema y se generan las relaciones pertenecientes al modelo Entidad – Relación, mismas que ya se establecieron en el capítulo anterior. La implementación de ésta etapa se muestra en la siguiente figura 23.

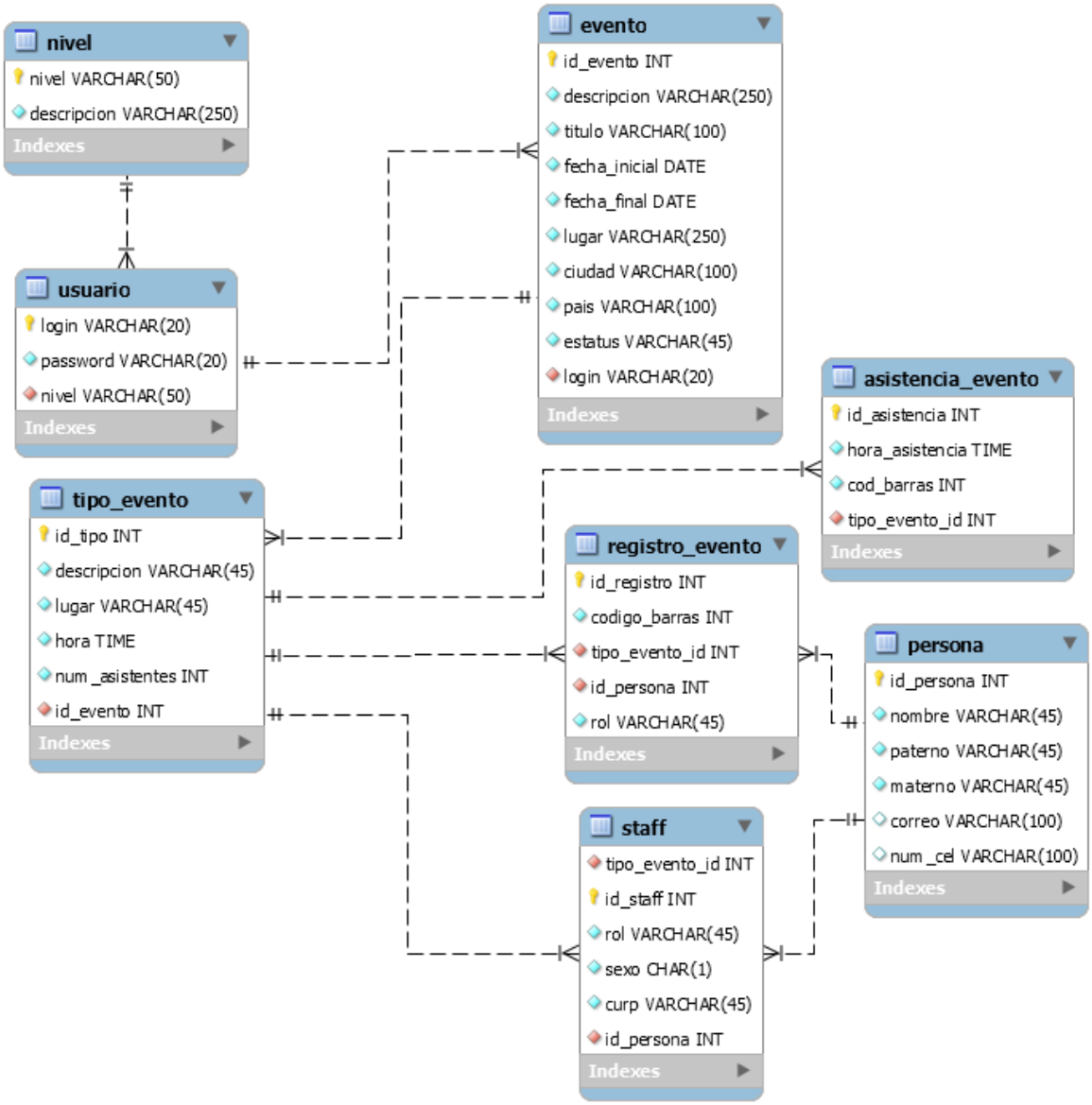

Figura 23. Relaciones entre las tablas del sistema.

Cuando se tiene el modelo entidad-relación de la base de datos, es momento de exportarlo como un script de instrucciones SQL, este script se muestra en el Anexo 1 de éste documento.

5.1.2 Interfaz del Sistema

5.1.2.1 Autenticación de Usuario

Como requisito para el uso del sistema es necesario identificarse como usuario, esto último define el rol y las funciones con las que podrá interactuar dentro del sistema.

Lo antes dicho corresponde a una pantalla de acceso que contempla un id de usuario y su contraseña para hacerlo, tal como se muestra en la figura 24.

Figura 24. Pantalla inicial de Login del Sistema.

Los datos son sometidos a validación. Si una persona está registrada e ingresa su usuario y contraseña de manera correcta, accede a las funcionalidades que ofrece el sistema, correspondiente a cada nivel.

5.1.2.2 Usuario Administrador

Dentro de las funciones imprescindibles del usuario administrador, está la de manipular la gestión relacionada a los eventos. En la siguiente figura se muestra la información de los eventos almacenados en la base de Datos. Así mismo, las filas de las tablas pueden ser vistas a detalle, modificadas y eliminadas, a través de los botones ubicados a la derecha de cada fila.

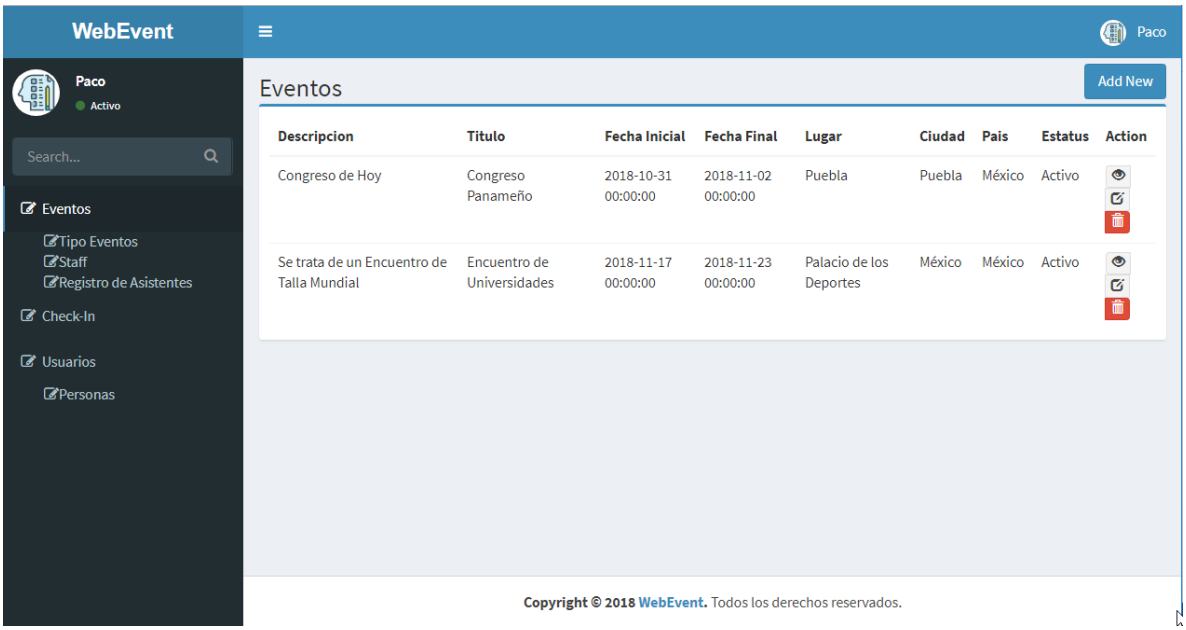

Figura 25. Panel de Administración de eventos.

5.2 Pruebas

Finalmente para la evaluación del funcionamiento de la plataforma se han aplicado una serie de pruebas que han permitido comprobar tanto la funcionalidad de los módulos como las interfaces de usuario, y gracias a estas se lograron detectar y corregir las fallas que había en el sistema.

De igual manera, dichas pruebas de evaluación demuestran que la plataforma realiza las funciones que se han especificado en los requerimientos iniciales. Esto se ha logrado debido a que se han aplicado dos tipos de pruebas: funcionales y estructurales.

Estructurales: son pruebas que se concentran en lo que hay codificado o diseñado a bajo nivel por lo que no es necesario reconocer la especificación de requisitos. Su cometido es comprobar los flujos de ejecución dentro de cada función o módulo.

Funcionales: son pruebas que parten de los requisitos funcionales aplicándolas sobre el sistema empleando un determinado conjunto de datos de entrada y observando las salidas que produce, de manera que se contraste con las especificaciones y determinar si está desempeñando correctamente su función.

A continuación se describen algunas de las pruebas realizadas para la plataforma, tanto funcionales como estructurales.

5.2.1 Pruebas estructurales

Para la realización de estas pruebas se eligieron módulos específicos de la plataforma de cada aplicación, se elaboraron casos de prueba y se realizó la verificación de las mismas.

Módulo: Validación de Usuario.

Descripción: No se ingresa un usuario y se ingresa el password: **1234,** por tanto la casilla de nombre de usuario se encuentra vacía. El modulo verifica que se haya ingresado información en los cuadros de texto usuario y password.

Resultado esperado: no hay datos en el cuadro de texto usuario, el sistema envía un mensaje de Error: El campo Correo es requerido.

Vista

<!--Se ingresa un usuario y password en los cuadros de texto del formulario y se envía para su validación->

```
<form method="post" action="{{ url('/login') }}">
        {!! csrf_field() !!}

        <div class="form-group has-feedback {{ $errors->has('email') ? ' 
has-error' : '' }}">
            <input type="email" class="form-control" name="email" 
value="{{ old('email') }}" placeholder="Correo">
            <span class="glyphicon glyphicon-envelope form-control-
feedback"></span>
            @if ($errors->has('email'))
               <span class="help-block">
               <strong>{{ $errors->first('email') }}</strong>
            </span>
            @endif
        </div>

        <div class="form-group has-feedback{{ $errors->has('password') 
? ' has-error' : '' }}">
```

```
            <input type="password" class="form-control"
placeholder="Contraseña" name="password">
            <span class="glyphicon glyphicon-lock form-control-
feedback"></span>
            @if ($errors->has('password'))
                <span class="help-block">
                <strong>{{ $errors->first('password') }}</strong>
            </span>
            @endif

        </div>
        <div class="row">
            <div class="col-xs-8">
                <div class="checkbox icheck">
                    <label>
                        <input type="checkbox" name="remember">
Recuérdame
                    </label>
                </div>
            </div>
            <!-- /.col -->
            <div class="col-xs-4">
                <button type="submit" class="btn btn-primary btn-block btn-
flat">Entrar</button>
            </div>
            <!-- /.col -->
        </div>
    </form>
```

Controlador

```
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Foundation\Auth\AuthenticatesUsers;

class LoginController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    | Login Controller
    |--------------------------------------------------------------------------
    |
    | This controller handles authenticating users for the application and
    | redirecting them to your home screen. The controller uses a trait
```

```
    | to conveniently provide its functionality to your applications.
    |
    */

    use AuthenticatesUsers;

    /**
     * Where to redirect users after login.
     *
     * @var string
     */
    protected $redirectTo = '/home';

    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest')->except('logout');
    }
}
```

5.2.3 Pruebas Funcionales

Función: Realizar la eliminación de un Evento.

Descripción: Se posiciona sobre la fila del evento a eliminar, se presiona el botón con el ícono de bote de basura (figura 26).

Resultado Esperado: Un mensaje con una Leyenda para confirmar la acción, y al dar en aceptar (figura 27), la fila es eliminada de la vista, aparece un mensaje con la leyenda de acción exitosa y es guardado el registro de su eliminación en la base de datos (figura 28).

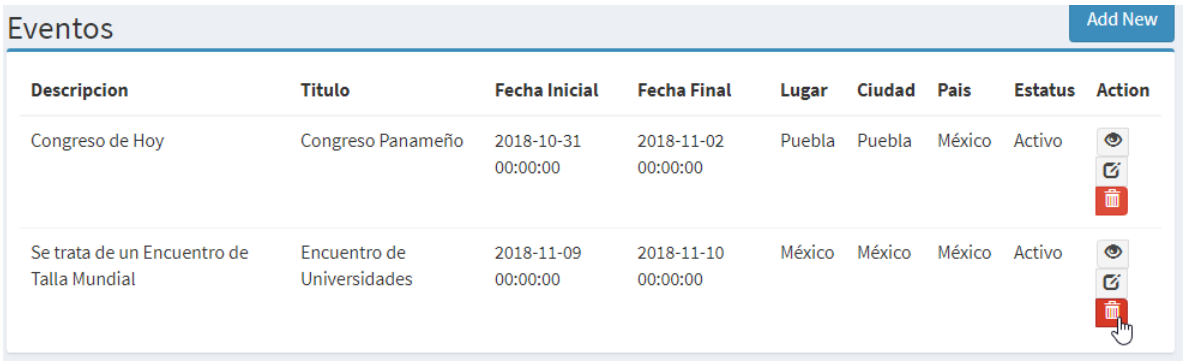

Figura 26. Acción de presionar sobre el ícono para eliminar elemento.

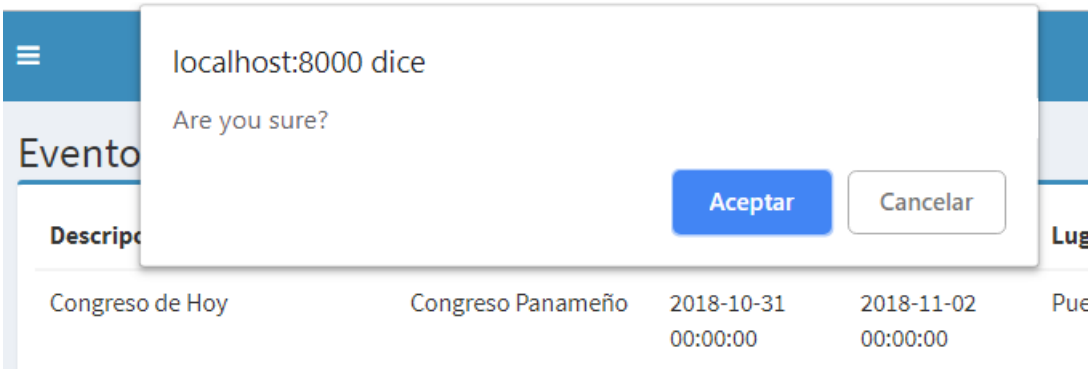

Figura 27. Cuadro de diálogo de confirmación

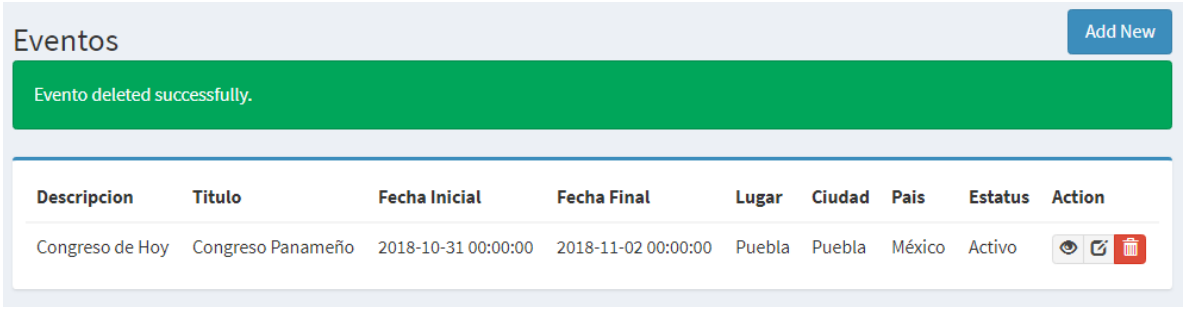

Figura 28. Mensaje de acción realizada con éxito.

CONCLUSIONES Y TRABAJOS A FUTURO

El auge de las aplicaciones web y su uso como herramienta diaria, facilita en demasía muchas de las tareas que ya son cotidianas en los contextos donde la tecnología está al alcance. Es por ello que cada vez más se buscan maneras de hacer fácilmente actividades, que ahora ya son familiares a través de la usabilidad de sistemas. En la organización de Eventos, acortar tiempos en realización de tareas comunes es esencial para que la logística sea un proceso que rinda frutos.

A pesar de que las aplicaciones actuales permiten realizar la venta de tickets mediante plataformas en línea, es posible encontrar desventajas con respecto a la realización de eventos que se llevan a cabo de manera continua, ya que la base de datos es obtenida mediante hojas de cálculo y el procesamiento de ésta información, puede resultar un proceso tedioso. El segmento al que va dirigido éste proyecto es a las instituciones quienes realizan eventos constantemente y poseen una base de datos que puede ser utilizada para el óptimo registro y check-in de usuarios.

Trabajo a Futuro

Algunas propuestas de trabajo futuro que contribuirían al trabajo realizado en esta tesina son:

Centralización de la Información: Dado que la realización de un evento involucra datos como material, tarjetas de contacto, papers, es necesario centralizar ésta información para que el usuario pueda tener acceso a ella, y la plataforma pretende ser un mediador para que posterior al evento el contacto que asistió al taller sea portador de la misma.

Colaboración con otros desarrollos: Existen plataformas de pago que cada institución ha desarrollado particularmente, tal es el caso de “Pagos Referenciados BUAP”, en colaboración con el sistema, sería posible vincular un pago que genere un Boucher y éste al ser ingresado al sistema genere el acceso al Evento.

Veracidad de la Información: Se ha comprobado que exalumnos han hecho uso de material de la institución, al que sólo tienen acceso los alumnos activos. En

cotejamiento con la base de Datos institucional, se podrá corroborar la autenticidad de los datos de la persona que se registre y así evitar procesos engorrosos como ir presencialmente a dejar boletas de calificaciones.

REFERENCIAS

Bibliografía

1. Philippe Kruchten, The Rational Unified Process An Introduction, Addison Wesley, 2001.
2. “Open Source Definition, v1.9”. Open Source Initiative. 2.005.
    A. Gómez Labrador: “Software Libre en el Escritorio de Inpro”, versión previa. 2.005.
3. Sommerville, I. (2011). Ingeniería de Software. México: Pearson.
4. Silberschatz, A; Korth, H. & Sudarshan, S. (2002). Fundamentos de Bases de Datos. España: Mc Graw Hill.
5. Pérez de Celis, M. & Somodevilla, M. (2012). El Modelo Entidad Ralación. En M. Bonne. (Eds.). Objetos de Aprendizaje para la Enseñanza de Bases de Datos. (pp. 19-31). Puebla.
6. Ambrosio, A. & Ochoa, J. (2012). El Proceso de Normalización. En M. Bonne. (Eds.). Objetos de Aprendizaje para la Enseñanza de Bases de Datos. (pp. 34-55). Puebla.
7. Kendall & Kendall, Análisis y diseño de sistemas, Tercera Edición, Ed. Prentice Hall.
8. Klever Ayala, A. H. (01 de 02 de 2007). Herramienta generadora de aplicaciones web con arquitectura MVC en Java. Herramienta generadora de aplicaciones web con arquitectura MVC en Java. Quito, Pichincha, Ecuador: ESPOL.

Mesografía.

9. Open Source Initiative.
   http://opensource.org/
10. Ayuda de Github
    https://help.github.com/
11. Comisiones Boletia.

https://boletia.com/precios-y-comisiones

12. Evenvrite.
    https://www.eventbrite.es.
13. Formularios de Google.
    https://gsuite.google.com.mx/intl/es-419_mx/products/forms/
14. Survey Monkey
    https://es.surveymonkey.com/
15. Laravel Documentación.
    https://laravel.com/
16. AdminLTE.
    https://adminlte.io/
17. Metodologías de Desarrollo
    https://okhosting.com/blog/metodologias-del-desarrollo-de-software/
18. Modelo y Metodologías para el desarrollo de Software
    http://www.eumed.net/tesis-doctorales/2014/jlcv/software.htm
19. Proceso Unificado (UP)
    http://es.wikipedia.org/wiki/Proceso_Unificado
20. Cliente –Servidor
    https://www.ecured.cu/Cliente-Servidor
21. Arquitectura Cliente –Servidor
    http://deprogramacion.cubava.cu/2016/01/11/arquitectura-cliente-servidor/
22. Wikipedia. (15 de 07 de 2014). wikipedia.com. Obtenido de eikipedia.com: http://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador
23. Curso Mentor Apache
    https://sede.educacion.gob.es/publiventa/PdfServlet?pdf=VP15824.pdf&area=E
24. PHP, «Prefacio - Manual de PHP,» [En línea]. Available: https://secure.php.net/manual/es/preface.php. [Último acceso: 10 Octubre 2018].

25. M Cristina. (16 de 12 de 2014). Nubelo. Obtenido de: http://blog.nubelo.com/que-son-losframeworks/
26. Laravel para principiantes https://www.bcnbit.com/laravel-4-resumen-para-principiantes-parte-i/
27. Enrutamiento básico https://richos.gitbooks.io/laravel-5/content/capitulos/chapter9.html
28. Ventajas Laravel https://github.com/bertus193/devCode/wiki/Ventajas-Laravel
29. Sistema Gestor de Base de Datos https://www.ecured.cu/Sistema_Gestor_de_Base_de_Datos

ANEXOS

ANEXO 1. SCRIPT PARA GENERAR LA BASE DE DATOS DEL SISTEMA.

```
DROP SCHEMA IF EXISTS `webevent` ;

-- -----------------------------------------------------
-- Schema webevent
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `webevent` DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci ;
USE `webevent` ;

-- -----------------------------------------------------
-- Table `webevent`.`nivel`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `webevent`.`nivel` ;

CREATE TABLE IF NOT EXISTS `webevent`.`nivel` (
  `nivel` VARCHAR(50) NOT NULL,
  `descripcion` VARCHAR(250) NOT NULL,
  PRIMARY KEY (`nivel`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `webevent`.`usuario`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `webevent`.`usuario` ;

CREATE TABLE IF NOT EXISTS `webevent`.`usuario` (
  `login` VARCHAR(20) NOT NULL,
  `password` VARCHAR(20) NOT NULL,
  `nivel` VARCHAR(50) NOT NULL,
  PRIMARY KEY (`login`),
  INDEX `fk_usuario_nivel_idx` (`nivel` ASC),
  CONSTRAINT `fk_usuario_nivel`
    FOREIGN KEY (`nivel`)
    REFERENCES `webevent`.`nivel` (`nivel`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `webevent`.`evento`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `webevent`.`evento` ;

CREATE TABLE IF NOT EXISTS `webevent`.`evento` (
  `id_evento` INT NOT NULL AUTO_INCREMENT,
  `descripcion` VARCHAR(250) NOT NULL,
  `titulo` VARCHAR(100) NOT NULL,
  `fecha_inicial` DATE NOT NULL,
  `fecha_final` DATE NOT NULL,
  `lugar` VARCHAR(250) NOT NULL,
  `ciudad` VARCHAR(100) NOT NULL,
  `pais` VARCHAR(100) NOT NULL,
  `estatus` VARCHAR(45) NOT NULL,
  `login` VARCHAR(20) NOT NULL,
```

```
  PRIMARY KEY (`id_evento`),
  INDEX `fk_evento_usuario1_idx` (`login` ASC),
  CONSTRAINT `fk_evento_usuario1`
    FOREIGN KEY (`login`)
    REFERENCES `webevent`.`usuario` (`login`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `webevent`.`tipo_evento`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `webevent`.`tipo_evento` ;

CREATE TABLE IF NOT EXISTS `webevent`.`tipo_evento` (
  `id_tipo` INT NOT NULL AUTO_INCREMENT,
  `descripcion` VARCHAR(45) NOT NULL,
  `lugar` VARCHAR(45) NOT NULL,
  `hora` TIME NOT NULL,
  `num_asistentes` INT NOT NULL,
  `id_evento` INT NOT NULL,
  PRIMARY KEY (`id_tipo`),
  INDEX `fk_tipo_evento_evento1_idx` (`id_evento` ASC),
  CONSTRAINT `fk_tipo_evento_evento1`
    FOREIGN KEY (`id_evento`)
    REFERENCES `webevent`.`evento` (`id_evento`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `webevent`.`persona`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `webevent`.`persona` ;

CREATE TABLE IF NOT EXISTS `webevent`.`persona` (
  `id_persona` INT NOT NULL AUTO_INCREMENT,
  `nombre` VARCHAR(45) NOT NULL,
  `paterno` VARCHAR(45) NOT NULL,
  `materno` VARCHAR(45) NOT NULL,
  `correo` VARCHAR(100) NULL,
  `num_cel` VARCHAR(100) NULL,
  PRIMARY KEY (`id_persona`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `webevent`.`registro_evento`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `webevent`.`registro_evento` ;

CREATE TABLE IF NOT EXISTS `webevent`.`registro_evento` (
  `id_registro` INT NOT NULL AUTO_INCREMENT,
  `codigo_barras` INT NOT NULL,
  `tipo_evento_id` INT NOT NULL,
  `id_persona` INT NOT NULL,
  `rol` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id_registro`),
  INDEX `fk_registro_evento_tipo_evento1_idx` (`tipo_evento_id` ASC),
  INDEX `fk_registro_evento_persona1_idx` (`id_persona` ASC),
```

```
  CONSTRAINT `fk_registro_evento_tipo_evento1`
    FOREIGN KEY (`tipo_evento_id`)
    REFERENCES `webevent`.`tipo_evento` (`id_tipo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_registro_evento_persona1`
    FOREIGN KEY (`id_persona`)
    REFERENCES `webevent`.`persona` (`id_persona`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `webevent`.`asistencia_evento`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `webevent`.`asistencia_evento` ;

CREATE TABLE IF NOT EXISTS `webevent`.`asistencia_evento` (
  `id_asistencia` INT NOT NULL AUTO_INCREMENT,
  `hora_asistencia` TIME NOT NULL,
  `cod_barras` INT NOT NULL,
  `tipo_evento_id` INT NOT NULL,
  PRIMARY KEY (`id_asistencia`),
  INDEX `fk_asistencia_evento_tipo_evento1_idx` (`tipo_evento_id` ASC),
  CONSTRAINT `fk_asistencia_evento_tipo_evento1`
    FOREIGN KEY (`tipo_evento_id`)
    REFERENCES `webevent`.`tipo_evento` (`id_tipo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `webevent`.`staff`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `webevent`.`staff` ;

CREATE TABLE IF NOT EXISTS `webevent`.`staff` (
  `tipo_evento_id` INT NOT NULL,
  `id_staff` INT NOT NULL AUTO_INCREMENT,
  `rol` VARCHAR(45) NOT NULL,
  `sexo` CHAR(1) NOT NULL,
  `curp` VARCHAR(45) NOT NULL,
  `id_persona` INT NOT NULL,
  PRIMARY KEY (`id_staff`),
  INDEX `fk_tipo_evento_has_personal_tipo_evento1_idx` (`tipo_evento_id` ASC),
  INDEX `fk_apoyo_evento_persona1_idx` (`id_persona` ASC),
  CONSTRAINT `fk_tipo_evento_has_personal_tipo_evento1`
    FOREIGN KEY (`tipo_evento_id`)
    REFERENCES `webevent`.`tipo_evento` (`id_tipo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_apoyo_evento_persona1`
    FOREIGN KEY (`id_persona`)
    REFERENCES `webevent`.`persona` (`id_persona`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```