DISEÑO DE UNA PLATAFORMA BACKEND QUE PERMITA LA DIFUSIÓN DE EVENTOS EN EL CAMPUS UNIVERSITARIO

CARLOS ANDRES LEON VALDERRAMA

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2019

DISEÑO DE UNA PLATAFORMA BACKEND QUE PERMITA LA DIFUSIÓN DE EVENTOS EN EL CAMPUS UNIVERSITARIO

CARLOS ANDRES LEON VALDERRAMA

Trabajo de Grado para optar el título de Ingeniero de Sistemas

DIRECTOR
GABRIEL ROGRIGO PEDRAZA FERREIRA
PhD en Ciencias de la Computación

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2019

CONTENIDO

Pág.

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

RESUMEN

TÍTULO: DISEÑO DE UNA PLATAFORMA BACKEND QUE PERMITA LA DIFUSIÓN DE EVENTOS EN EL CAMPUS UNIVERSITARIO*

AUTOR: CARLOS ANDRES LEON VALDERRAMA**

PALABRAS CLAVE: Eventos, Smart Campus, REST, HTTP, JSON, MongoDB

DESCRIPCIÓN:

Se conoce como “Smart Cities” a un tipo de desarrollo urbano basado en la sostenibilidad que es capaz de responder adecuadamente a las necesidades básicas de quienes la componen. Ahora, el “Smart Campus” es en esencia similar a la “Smart City”, pero dedicado a un grupo de individuos más específicos, con similitudes e intereses comunes, ya que todos pertenecen a una comunidad universitaria. El “Smart Campus” busca mejorar la calidad de vida de estos individuos supliendo sus necesidades básicas o mejorando las soluciones ya existentes.

Con base en lo anterior se identificó un problema existente en el campus universitario, específicamente la difusión de eventos, la cual se realiza por diversos canales de comunicación, los cuales son de una sola vía, por esto no permiten un retorno de información de valor para los organizadores. Además, ninguno de estos canales contiene todos los eventos realizados por el campus y los individuos no acceden a todos los canales, debido a lo anterior la información no llega a todos los posibles interesados. Por esto es necesario poseer una plataforma capaz de integrar todos los eventos, además de ofrecer servicios para organizadores y usuarios, de esta forma generando un canal de comunicación de dos vías, adicionalmente, dicha plataforma debe ser lo suficientemente robusta para mantener la calidad de los servicios ante un volumen de peticiones grande.

* Trabajo de grado

** Facultad de ingenierías Físico-Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director Gabriel Rodrigo Pedraza Ferreira, Ph.D.

ABSTRACT

TITLE: DESIGN OF A BACKEND PLATFORM THAT ALLOWS THE DISSEMINATION OF EVENTS IN THE UNIVERSITY CAMPUS*

AUTHOR: CARLOS ANDRES LEON VALDERRAMA**

KEYWORDS: Events, Smart Campus, REST, HTTP, JSON, MongoDB

DESCRIPTION:

"Smart Cities" is a type of urban development based on sustainability that is able to respond adequately to the basic needs of its members. Now, the “Smart Campus” is essentially the similary as the “Smart City”, but dedicated to a group of more specific individuals, with similarities and common interests, since they all belong to a university community. The "Smart Campus" seeks to improve the quality of life of these individuals by supplying their basic needs or improving existing solutions.

Based on the above, an existing problem was identified on the university campus, specifically the dissemination of events, which is carried out through various communication channels, which are one-way, so they do not allow a return of information of value for organizers. In addition, none of these channels contains all the events carried out by the campus and individuals do not access all the channels, due to the above the information does not reach all potential stakeholders. Therefore, it is necessary to have a platform capable of integrating all events, in addition to offering services for organizers and users, thus generating a two-way communication channel, additionally, said platform must be robust enough to maintain the quality of the services before a large volume of requests.

* Bachelor Thesis

** Facultad de ingenierías Físico-Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director Gabriel Rodrigo Pedraza Ferreira, Ph.D.

INTRODUCCIÓN

Las “Smart Cities” conocidas en español como “ciudades inteligentes” a veces también llamadas “ciudades eficientes” o “ciudades súper-eficientes”, se refiere a un tipo de desarrollo urbano basado en la sostenibilidad que es capaz de responder adecuadamente a las necesidades básicas de instituciones, empresas, y de los propios habitantes, tanto en el plano económico, como en los aspectos operativos, sociales y ambientales. Ahora, el “Smart Campus” es en esencia similar a la “Smart City”, pero dedicado a un grupo de individuos más específicos, con similitudes e intereses comunes, ya que todos pertenecen a una comunidad universitaria. Análogamente el “Smart Campus” busca mejorar la calidad de vida de los individuos supliendo sus necesidades básicas o mejorando las soluciones ya existentes.

Colombia está posicionada como uno de los países con mayor cantidad de usuarios de Smartphones, con 14,4 millones en 2014 y con un índice de crecimiento del 23% con respecto al año anterior [1]. Y en 2016 aumentó 50% el uso de smartphones en Colombia, Así lo revela el primer ‘Indicador de terminales por cada 100 habitantes’, realizado por la firma Infométrika para el Ministerio de Tecnologías de la Información y las Comunicaciones (MinTIC). [2] [3]

[1] EL TIEMPO. En Colombia hay 14,4 millones de usuarios de 'smartphones'. [En línea]. (Recuperado el 2 de agosto 2019) Disponible en: http://www.eltiempo.com/archivo/documento/CMS-15066597

[2] EL TIEMPO. El uso de teléfonos inteligentes aumentó 50 % en Colombia en 2016. [En línea]. (Recuperado el 2 de agosto 2019) Disponible en: https://www.eltiempo.com/tecnosfera/novedades-tecnologia/en-50-por-ciento-crecio-el-uso-de-telefonos-inteligentes-en-colombia-89060

[3] MINISTERIO DE TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES. Tenencia de smartphones aumentó 50% en Colombia en el 2016. [En línea]. (Recuperado el 2 de agosto 2019) Disponible en: http://www.mintic.gov.co/portal/604/w3-article-51641.html

En cualquier contexto, la correcta difusión de información juega un papel fundamental, tanto para el emisor como para el receptor. En el caso particular del campus universitario se presenta un problema, la inefectiva divulgación de sus eventos, debido a que la información no llega a todo el posible público que pueda estar interesado y además dicho público no genera ningún tipo de retorno de información que permita una optimización de los eventos, esto se debe a la inexistencia de un canal único que permita difundir todos los diferentes eventos y además, los canales usados actualmente no permiten retornar información de valor para los organizadores e incluso ningún tipo de información.

Debido a la necesidad de difundir correctamente la información en el campus y el alto índice de usuarios de Smartphones que pertenecen a la Universidad, es factible diseñar un software que permita la difusión efectiva de los eventos realizados en la universidad, tal que se genere un canal de comunicación de dos vías, donde los organizadores puedan enviar la información directamente a su público objetivo, quienes la recibirán en sus teléfonos inteligentes, y a su vez, estos puedan generar un retorno de información de interés para el organizador, permitiendo una optimización de todos los aspectos referentes a los eventos.

Tomando en consideración la gran cantidad de usuarios potenciales pertenecientes a la comunidad universitaria, la cual en el segundo semestre de 2016 alcanzó las siguientes cifras [4]:

- 18505 estudiantes matriculados en pregrado.
- 2147 estudiantes matriculados en posgrado.
- 835 docentes totales (tiempo completo, tiempo parcial y cátedra).
- 1174 personal administrativo.

[4] UNIVERSIDAD INDUSTRIAL DE SANTANDER. UIS en cifras 2016. Edición No 41. Julio 2017. ISSN 0120-212X

Además, teniendo en cuenta que la cantidad de actividades culturales dirigidas a la comunidad universitaria en el 2016 fue de 245, sin olvidar las actividades de tipo académico y deportivo, entre otras. Teniendo en cuenta lo anterior podemos llegar a la conclusión que es fundamental poseer una plataforma robusta capaz de responder a un volumen de peticiones grande, de forma fluida, tal que genere una experiencia de usuario agradable.

Dicho software en su primera versión estará constituido por tres partes: Dos plataformas Front-end que serán las capas de presentación del software, una web dirigida a los organizadores de los eventos, una móvil dirigida a los integrantes de la comunidad universitaria interesados en los eventos. Y una plataforma Back-end que será la capa de acceso a datos, siendo esta última el objeto de este proyecto.

1. OBJETIVOS

1.1 OBJETIVO GENERAL

Diseñar una plataforma backend que permita a las diferentes unidades de la universidad difundir sus eventos y que su público objetivo genere un retorno de información de su interés.

1.2 OBJETIVOS ESPECÍFICOS

- Diseñar la arquitectura de componentes Backend que se encargará de recibir e integrar los datos provenientes de las plataformas Front-end asociadas.
- Proveer un conjunto de APIs que permitan acceder a información de interés para los dos sectores de usuarios; interesados y administradores.
- Implementar un prototipo software basado en la arquitectura planteada que incluya un subconjunto de componentes y APIS propuestos para la arquitectura Backend.
- Validar el prototipo implementado para verificar su funcionalidad y propiedades no funcionales.

2. MARCO DE REFERENCIA

2.1 DIFUSIÓN DE EVENTOS

El éxito de un evento depende del cumplimiento de los objetivos de los organizadores. Entre esos objetivos sin lugar a dudas se encuentra el conseguir la mayor asistencia posible de nuestro público potencial. Para lograr esto, es necesario trabajar durante la etapa de planificación encontrando una estrategia de difusión que llegue a la audiencia objetivo y les interese asistir al curso, taller, congreso o conferencia que se les proponga. Para lograr esto, es necesario definir dos cuestiones:

- Cuál es el mensaje a comunicar
- Cómo es el público al que se apunta

En lo que respecta al mensaje, es necesario que sea claro y que responda una serie de interrogantes que van a definir el interés o no del posible asistente a nuestro evento: Las 5 W.

What / Qué: En este punto hablamos de la denominación del evento, el nombre y sus contenidos.

Who / Quién: quién organiza, quiénes son los invitados especiales, oradores, moderadores, coordinadores, etc.

Where / Dónde: La sede de la reunión, capacidad, servicios e infraestructura.

When / Cuándo: la fecha y duración del evento.

Why / Por qué: la temática y los objetivos que motivan la reunión.

A esto podemos sumarle una sexta pregunta:

How / Cómo será el desarrollo, el programa, los recursos, patrocinios y auspicios.

Conociendo cual es la identidad que se le quiere dar al evento, ayudará a establecer el mensaje que se quiere comunicar.

Por otro lado, es necesario analizar, el público objetivo, el TARGET, poder identificar a la audiencia, formulando cuál es el perfil de los potenciales asistentes [5]. En este aspecto tenemos un punto a favor, ya que los usuarios podrán identificar cuáles son los tipos de eventos que son de su interés, es decir, ellos mismos darán su perfil de cara a los eventos que les interesan.

Una vez que están bien claros estos puntos, determinar los canales más óptimos de difusión.

Principales canales difusión de un evento:

- Sitios Web o Blog: Va a ser el lugar al cual acudan los potenciales asistentes para ampliar la información que reciben sobre el evento.
- Correo Electrónico: Si se cuenta con una base de datos, puede ser muy eficiente, ya que podemos tener un alto alcance a muy bajo costo.
- Redes Sociales: No alcanza únicamente con una web o blog, sino que es importante posicionar el evento en las redes sociales como facebook, twitter, linkedin y hasta youtube.
- Correo Directo: Confección de Folleto informativo o “Anuncios” previos al evento y envío por correo postal. Este medio es más costoso y tomará más

[5] UNIVERSIDAD DE PALERMO. [En línea]. (Recuperado el 19 de junio 2017) Disponible en: http://fido.palermo.edu/servicios_dyc/blog/docentes/trabajos/15609_51342.pdf

tiempo, ya que se debe diseñar e imprimir y luego hacer envío por correo postal para que llegue a los domicilios de la base de datos [6]

Pero la mayoría de estos canales de difusión tienen un problema en común, son de una sola vía, a excepción de las redes sociales, ya que en estas se puede generar un canal de comunicación entre asistentes y organizadores, pero de forma ineficiente, ya que el ruido en la información usualmente suele ser grande, lo que dificulta el proceso de mejora. Debido a esto se desea diseñar un software que permita difundir eventos, dándole participación a los asistentes y posibles asistentes, generando un feedback de información valiosa para los organizadores.

Las aplicaciones de los eventos son una excelente manera de ayudar a los asistentes a consultar los planos de planta, obtener información sobre las sesiones y programar sus jornadas. A pesar de que el 83 por ciento de los asistentes utiliza las redes sociales para los eventos, el 59 por ciento de los encuestados mostraron reticencias ante las aplicaciones de los eventos, según se explica en The Event App Bible. Esto puede deberse, en parte, a que los asistentes buscan funciones que las aplicaciones de los eventos no ofrecen, como la integración del registro.

Solamente un 33 por ciento de los organizadores profesionales afirma que al menos la mitad de sus invitados usan la aplicación de su evento. Entonces, ¿qué pueden hacer los organizadores profesionales de eventos para mejorar la experiencia que ofrecen estas aplicaciones a sus invitados? [7]. La respuesta es crear una aplicación participativa para los asistentes, que además de enterarse simplemente de los eventos le permita:

1. Personalizar y filtrar los eventos de acuerdo a sus intereses.
2. Demostrar su interés por el evento e incluso confirmar su asistencia.
3. Evaluar los eventos de forma ágil.

[6] Ibíd.
[7] MEETINGSIMAGINED. El futuro de las aplicaciones de los eventos. [En línea]. (Recuperado el 2 de agosto 2019) Disponible en: http://es.meetingsimagined.com/tips-trends/future-event-apps

4. Compartir los eventos de su interés en las redes sociales.
5. Adicionalmente, que lo mantenga informado de nuevos eventos o de modificaciones en eventos de su interés, a través de notificaciones push.

2.2 PLATAFORMAS BACKEND, ARQUITECTURA Y CALIDAD DE SOFTWARE

En diseño de software el front-end es la parte del software que interactúa con el o los usuarios y el back-end es la parte que procesa la entrada desde el front-end. La separación del sistema en front-ends y back-ends es un tipo de abstracción que ayuda a mantener las diferentes partes del sistema separadas. La idea general es que el front-end sea el responsable de recolectar los datos de entrada del usuario, que pueden ser de muchas y variadas formas, y los transforma ajustándolos a las especificaciones que demanda el back-end para poder procesarlos, devolviendo generalmente una respuesta que el frontend recibe y expone al usuario de una forma entendible para este. La conexión del front-end y el back-end es un tipo de interfaz.

La labor del back-end la compone el acceso a bases de datos y generación de plantillas del lado del servidor. En backend se encargan de implementar cosas como MySQL, Postgres, SQL Server o MongoDB. Luego, un lenguaje como PHP o JSP, o frameworks como RoR, Django, Node.JS o .NET se conectan a la base de datos. A través de estos lenguajes y frameworks se recibe, procesa y envía información al navegador del usuario. En código HTML (que crea el frontend) o enviando datos puros en XML, RSS o JSON, para ser procesados por Javascript [8].

La arquitectura de software, tiene que ver con el diseño y la implementación de estructuras de software de alto nivel. Es el resultado de ensamblar un cierto

[8] CRISTALAB. Qué significa backend y frontend en el diseño web. [En línea]. (Recuperado el 2 de agosto 2019) Disponible en: http://www.cristalab.com/blog/que-significa-backend-y-frontend-en-el-diseno-web-c106224l/

número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales. Para esto el sistema debe contar con los siguientes atributos, que garanticen su calidad [9]:

Disponibilidad (Availability): Es la medida de disponibilidad del sistema para el uso.

Confidencialidad (Confidentiality): Es la ausencia de acceso no autorizado a la información.

Funcionalidad (Functionality): Habilidad del sistema para realizar el trabajo para el cual fue concebido.

Confiabilidad (Reliability): Es la medida de la habilidad de un sistema a mantenerse operativo a lo largo del tiempo.

Desempeño (Performance): Grado en el cual un sistema o componente cumple con su funcionalidad, dentro de ciertas restricciones dadas, como velocidad, exactitud o uso de memoria. Se refiere a aspectos temporales del comportamiento del sistema. Capacidad de respuesta, ya sea el tiempo requerido para responder a aspectos específicos o al número de eventos procesados en un intervalo de tiempo. • Se refiere además a la cantidad de comunicación e interacción existente entre los componentes del sistema.

[9] SOPHIA.JAVERIANA Atributos de Calidad de Software. [En línea]. (Recuperado el 19 de junio 2017) Disponible en: https://sophia.javeriana.edu.co/~cbustaca/docencia/DEAS-2017-01/presentaciones/AtributosCalidadSoftware.pdf

Seguridad Externa (Safety): Ausencia de consecuencias catastróficas en el ambiente. Es la medida de ausencia de errores que generan pérdidas de información.

Seguridad Interna (Security): Es la medida de la habilidad del sistema para resistir a intentos de uso no autorizados y negación del servicio, mientras se sirve a usuarios legítimos.

2.3 REST

La transferencia de estado representacional (REST) es un estilo de arquitectura propuesta por Roy Fielding, para el diseño de servicios web, para sistemas distribuidos basados en hipermedia. En las implementaciones de REST es común usar HTTP como protocolo de aplicación. [10]

2.3.1 Características de REST

- Cliente/Servidor: Los servicios web son cliente servidor y la comunicación se hace por medio de una interfaz, gracias a esto se separan las responsabilidades entre ambos.
- Sin estado: No se mantiene estado asociado al cliente, cada petición es independiente de las demás.
- Cache: El contenido de los servicios se puede cachear, para servir de apoyo para las siguientes peticiones.
- Servicios Uniformes: Los servicios comparten la forma de invocación usando los métodos GET, POST, PUT, DELETE.
- Arquitectura en Capas: Le permite estar orientado hacia la escalabilidad.

[10] ARQUITECTURAJAVA. Introducción a Servicios REST. [En línea]. (Recuperado el 2 de agosto 2019) Disponible en: https://www.arquitecturajava.com/servicios-rest/

2.4 BASES DE DATOS NO RELACIONALES

Las bases de datos no relacionales no usan el esquema de tabular filas y columnas, aunque muchas de estas admiten consultas compatibles con SQL. Son conocidas también como bases de datos orientadas a documentos, ya que usan un modelo de almacenamiento orientado a objetos. Debido a estas características y a su escalabilidad horizontal, son muy atractivas cuando se desea trabajar con grandes cantidades de información. [11]

2.5 DISEÑO DE API

La interfaz de programación de aplicaciones (API) es un conjunto de rutas que provee acceso a los servicios prestados por determinado sistema. El diseño de una API web debe poseer las siguientes características [12]:

- Independencia de la plataforma. Cualquier cliente debe poder hacer peticiones a la API independientemente de como esta esté implementada internamente. Para esto se usan protocolos estándar.
- Evolución del servicio. La API web debe ser capaz de evolucionar y agregar más servicios, sin afectar ni modificar los servicios ya existentes.

2.5.1 Principios de diseño. Algunos de los principios más importantes en el diseño de las API REST basadas en HTTP son los siguientes:

- Se diseñan en base a recursos, que podrían ser objetos, datos o servicios al que el cliente acceder.
- Cada recurso posee un URI, el cual lo identifica de forma única

[11] MICROSOFT. Datos no relacionales y NoSQL. [En línea]. (Recuperado el 2 de agosto 2019) Disponible en: https://docs.microsoft.com/es-es/azure/architecture/data-guide/big-data/non-relational-data

[12] MICROSOFT. Diseño de API. [En línea]. (Recuperado el 2 de agosto 2019) Disponible en: https://docs.microsoft.com/es-es/azure/architecture/best-practices/api-design

- El intercambio de información se realiza en un formato acordado entre el cliente y el servicio web, en este aspecto es muy popular el uso de JSON
- Las implementaciones de los clientes y servicios están desacopladas, gracias al uso de HTTP para realizar operaciones en los recursos. Las operaciones más comunes son GET, POST, PUT, PATCH y DELETE.
- Las API REST usan solicitudes sin estado, cada solicitud es atómica y la información solo persiste en los recursos, esto permite que los servicios web sean muy escalables.
- Estas API se controlan mediante vínculos de hipermedia contenidos en la representación establecida para la información. [13]

2.5.2 Modelo de madurez. Propuesto por Leonard Richardson, este modelo determina la madurez de las API web de la siguiente manera:

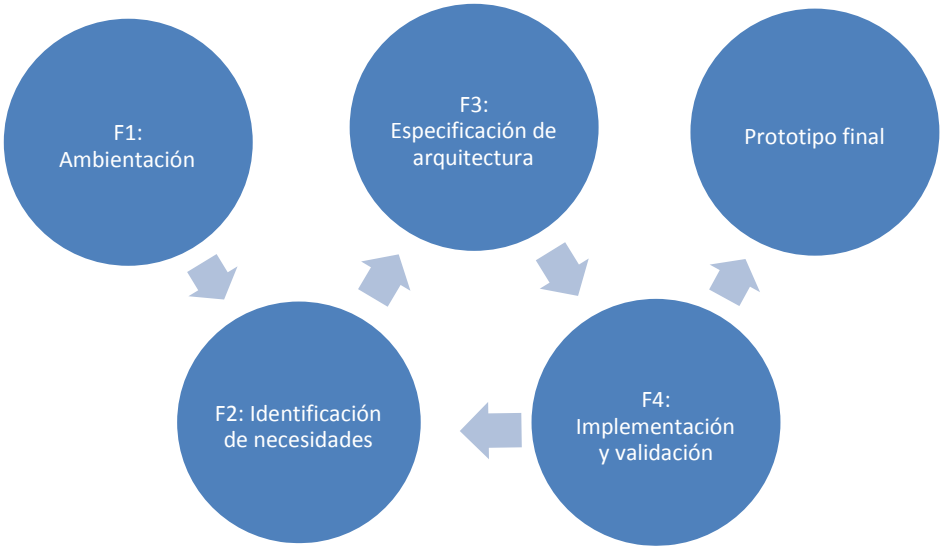
- Nivel 0: Definir un URI y todas las solicitudes son tipo POST a esta URI.
- Nivel 1: Crear URI diferentes para todos los recursos.
- Nivel 2: Usar métodos HTTP para crear operaciones en los recursos.
- Nivel 3: Usar hipermedia. [14]

[13] Ibíd.
[14] Ibíd.

3. METODOLOGÍA

La metodología a usar para el desarrollo del proyecto tiene como base la metodología de prototipos evolutivos, de forma que se puedan hacer modificaciones y/o mejoras futuras en caso de ser necesario añadir o aumentar la eficiencia de los requerimientos.

Figura 1. Fases del desarrollo del proyecto



Fase 1: Ambientación tecnológica.

Durante la primera fase se van a adquirir conocimientos relevantes para cumplir para el desarrollo del proyecto. Esta fase se extenderá a gran parte del desarrollo total del proyecto, acompañando en periodos las demás fases.

- Actividad 1.1. Revisión técnica de herramientas.
- Actividad 1.2. Construcción de ejemplos de las tecnologías seleccionadas.
- Entregable 1.1. Ejemplos implementados en las tecnologías.

Fase 2: Identificación de necesidades.

Por medio de la segunda fase se identificarán las necesidades generales y específicas que hay en el entorno a la difusión de eventos en el campus de la UIS, para esto se harán actividades como entrevistas con potenciales organizadores, por ejemplo, la dirección cultural de la UIS.

- Actividad 2.1. Entrevistas a diferentes unidades de la universidad.
- Actividad 2.2. Identificación de necesidades.
- Entregable 2.1. Documento especificación de necesidades.

Fase 3: Especificación de arquitectura

Con base en la fase anterior se propone la posible arquitectura software con sus características y posterior verificación de la misma.

- Actividad 3.1. Propuesta de arquitectura.
- Actividad 3.2. Verificación de arquitectura.
- Entregable 3.1 Documentación especificación de arquitectura.

Fase 4: Implementación y validación.

En esta fase se implementará el prototipo de acuerdo a las fases anteriores y se realizarán pruebas de simulación. Entra las actividades a realizar tenemos: codificación del prototipo, implementación de sus servicios y prueba con simulación de datos.

- Actividad 4.1. Codificación
- Actividad 4.2. Implementación
- Actividad 4.3. Pruebas
- Entregable 4.1. Prototipo

4. DESARROLLO DEL PROYECTO

4.1 DISEÑO

Después de realizar un análisis de la situación, se establecieron las necesidades de los usuarios y organizadores de los eventos, lo cual permitió determinar los requerimientos que debía cumplir el sistema para cubrir dichas necesidades, adicionalmente se incluyeron funcionalidades que permitan a los usuarios generar información de valor para los organizadores.

4.1.1 Requerimientos funcionales

- La plataforma debe ofrecer funciones básicas para las cuentas de usuarios y organizadores, específicamente crear cuenta, iniciar sesión y editar la información.
- La plataforma debe permitir a los organizadores crear y modificar eventos.
- La plataforma debe proveer información sobre los eventos a los usuarios, incluyendo servicios de búsqueda bajo diferentes parámetros.
- Es necesario que la plataforma ofrezca servicios que permita a los organizadores obtener una retroalimentación por parte de los usuarios.
- Es necesario que la plataforma pueda almacenar información de manera persistente para poder ser consultada.

4.1.2 Requerimientos no funcionales. Para proveer las funciones anteriores es necesario que el prototipo posea ciertas características no funcionales, que le permitan ofrecer un servicio de calidad a los diferentes usuarios, para esto se someterá el prototipo final a pruebas de carga y estrés con el fin de validar el

performance del mismo. Con estas pruebas deseamos evaluar el rendimiento y la escalabilidad del prototipo.

4.1.2.1 Rendimiento. Este requerimiento debe ser tenido en cuenta para examinar hasta qué punto el prototipo se ajusta a las expectativas en cuanto a tiempos de respuesta y es capaz de prestar sus servicios sin perder calidad.

4.1.2.2 Escalabilidad. Representa la capacidad de un sistema de expandirse para satisfacer sus necesidades. Lo validaremos sometiendo el prototipo a condiciones extremas.

4.1.3 Tecnología seleccionada

Tabla 1. Tecnología seleccionada

Tipo	Opción escogida	Descripción
Lenguaje	Java	Es un lenguaje de programación y una plataforma informática. Se eligió por ser rápido, seguro y fiable. [15]
Servidor de aplicaciones	Apache Tomcat	Se escogió como servidor web, debido su popularidad, además de estar desarrollado en Java, lo que permite que funcione en cualquier sistema operativo, con su máquina virtual java. [16] [17]
Implementación JAX-RS	Jersey	Es un cliente Resful, implementación de referencia de JAX-RS. Este último es un API de Java que proporciona soporte a los

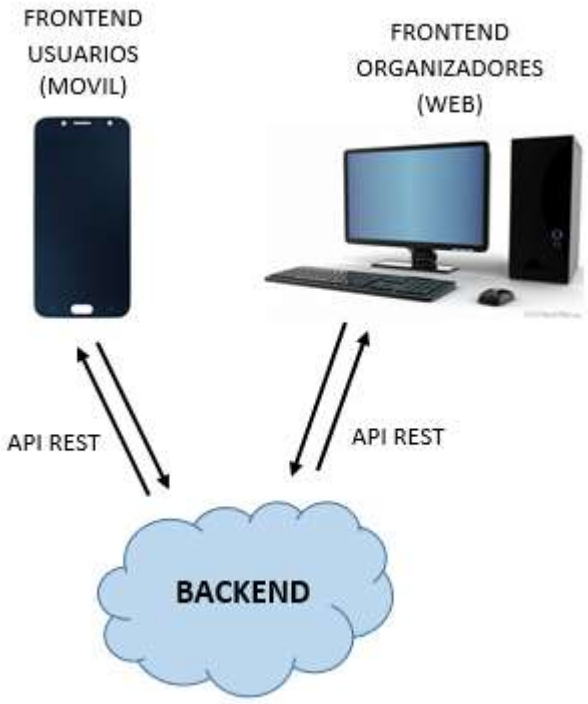
[15] ORACLE. ¿Qué es la tecnología Java y para qué la necesito? [En línea]. (Recuperado el 2 de agosto 2019) Disponible en: https://www.java.com/es/download/faq/whatis_java.xml
[16] AJPDSOFT. Tomcat, Apache Tomcat, Jakarta Tomcat. [En línea]. (Recuperado el 2 de agosto 2019) Disponible en: http://www.ajpdsoft.com/modules.php?name=Encyclopedia&op=content&tid=769
[17] APACHE TOMCAT®. Apache Tomcat. [En línea]. (Recuperado el 2 de agosto 2019) Disponible en: http://tomcat.apache.org/

		servicios web creados con una arquitectura REST. [18] [19]
Persistencia	MongoDB	La persistencia de la información se apoyó en una base de datos NoSQL, debido a los beneficios que esta ofrece, particularmente se escogió MongoDB por ser la más representativa dentro de este tipo de bases de datos.
Formato de la Información	JSON	Es un formato ligero de intercambio de datos. Es fácil de leer y escribir para las personas, y simple de interpretar y generar por parte de las máquinas. [20]

4.1.4 Arquitectura planteada. La arquitectura planteada para este proyecto se expone desde dos perspectivas, lógica que describe el comportamiento y física que describe la ejecución. Cabe recordar que el presente proyecto pertenece a la arquitectura general de un macro proyecto compuesto por tres proyectos, del cual forman parte dos proyectos de grado adicionales encargados del frontend, de esta forma la estructura del macro proyecto es la siguiente:

[18] IMAGINANET. Creación de aplicaciones Restful con Jersey. [En línea]. (Recuperado el 2 de agosto 2019) Disponible en: https://www.imaginanet.com/blog/creacion-de-aplicaciones-restful-con-jersey.html
[19] ORACLE. Java EE 7 y JAX-RS 2.0. [En línea]. (Recuperado el 2 de agosto 2019) Disponible en: https://www.oracle.com/technetwork/es/articles/java/java-ee7-y-jax-rs-2-2108434-esa.html
[20] JSON ORG. Introducción a JSON. [En línea]. (Recuperado el 2 de agosto 2019) Disponible en: https://www.json.org/json-es.html

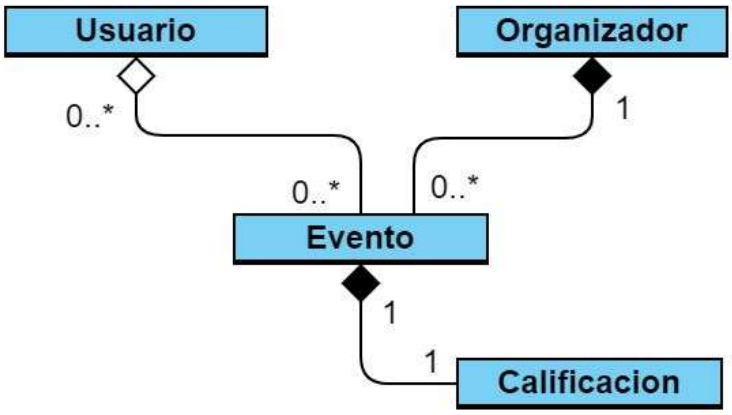
Figura 2. Arquitectura General del Macro Proyecto



La arquitectura presentada en la figura permite apreciar que la plataforma backend interactúa por medio de interfaces con las plataformas frontend asociadas, de las cuales recibe información que persiste en la base de datos, con la cual también posee interfaces para la comunicación. De igual forma la plataforma transmite información desde la base de datos a los diferentes tipos de usuarios de acuerdo a las peticiones de estos.

4.1.5 Diagrama de clases. Para conocer el comportamiento del sistema, se empleó un modelo de clases, que nos permite obtener un nivel de abstracción idóneo para este fin. El modelo de clases planteado es el siguiente:

Figura 3. Diagrama de clases conceptual

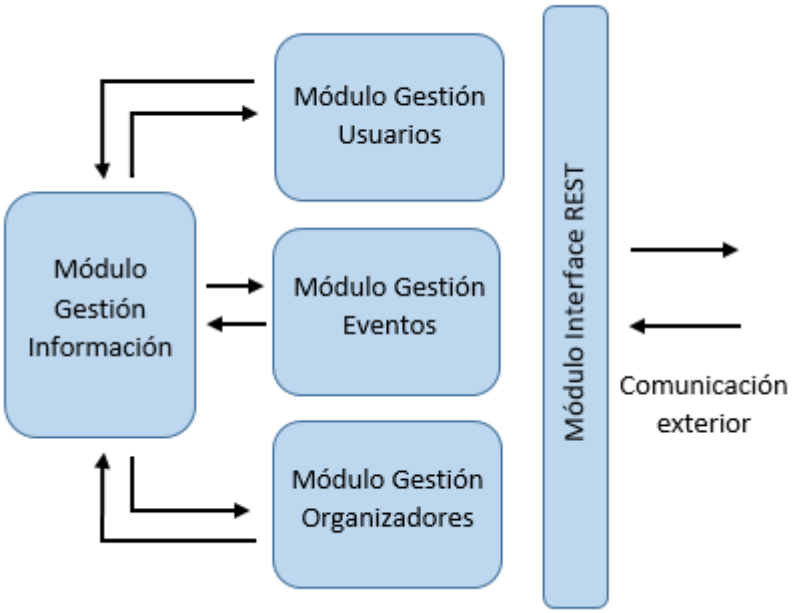


En el diagrama podemos observar los tres diferentes recursos que posee el sistema: usuario, organizador y evento. Adicionalmente, existe una clase calificación la cual posee una relación de composición con la clase evento, de igual forma la clase evento posee una relación de composición con la clase organizador, de tal forma que una calificación y un evento no existirían sin estar relacionados a un evento y a un organizador respectivamente.

A continuación, se presenta la arquitectura específica de la plataforma backend:

4.1.6 Arquitectura lógica

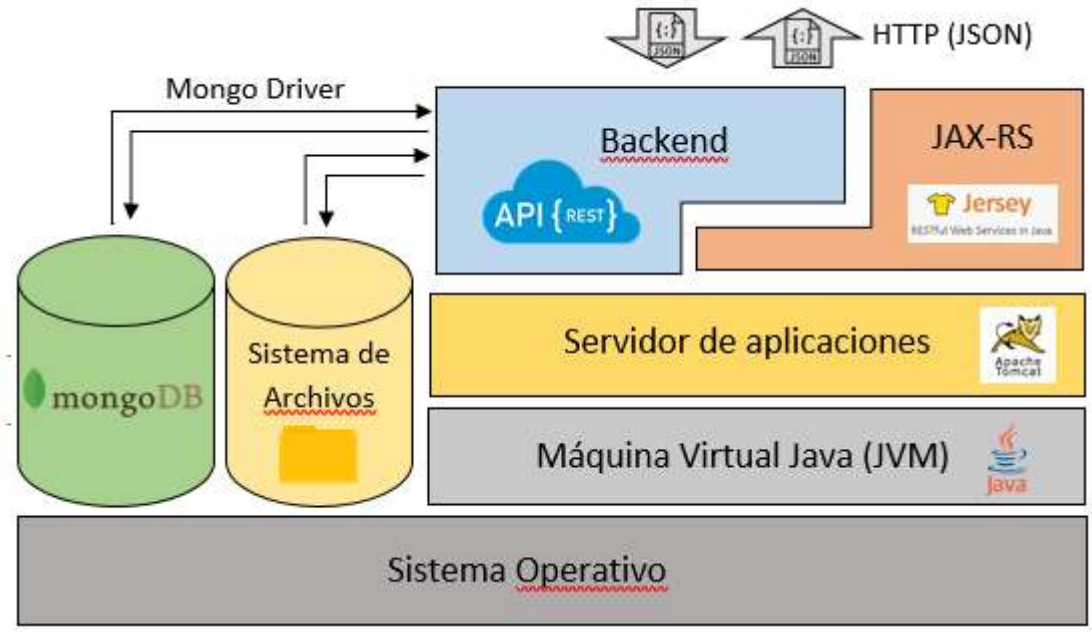
Figura 4. Arquitectura Lógica del prototipo



En la figura anterior se puede observar la arquitectura lógica del prototipo.

4.1.7 arquitectura física del prototipo

Figura 5. Arquitectura Física del prototipo

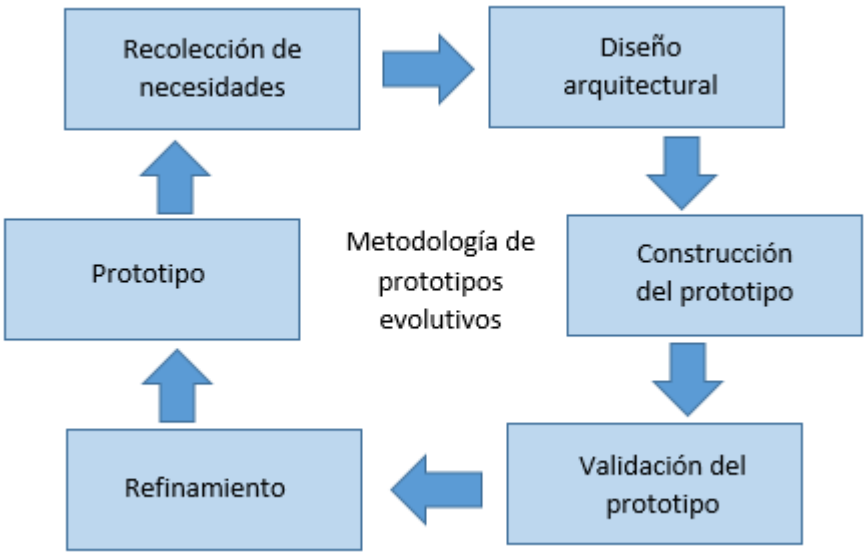


En la arquitectura presentada en la figura anterior podemos observar que el sistema operativo contiene la Máquina Virtual de Java (JVM), lo que permite ejecutar nuestro prototipo en cualquier maquina sin preocuparnos por el sistema operativo que posea. Sobre el JVM se encuentra Apache Tomcat como servidor de aplicaciones web para desplegar el prototipo, este último se apoya en las librerías de Jersey para ofrecer los servicios REST para establecer la comunicación con las plataformas asociadas mediante el protocolo HTTP, donde el intercambio de información se realiza en formato JSON. Finalmente, la persistencia de los datos se realiza en la base de datos no relacional MongoDB, y la persistencia de las imágenes se hace mediante un sistema de archivos.

4.2 IMPLEMENTACIÓN

El desarrollo del proyecto se basó en el modelo de prototipos evolutivos, con el cual es posible elaborar un producto software mediante ciclos, obteniendo un prototipo en cada iteración hasta llegar a una versión madura que satisfaga los requerimientos.

Figura 6. Metodología de prototipos evolutivos



Debido a que el desarrollo del proyecto se basó en la metodología de prototipos evolutivos, se realizaron tres ciclos de la siguiente manera:

4.2.1 Ciclo 1: Conocimiento de las tecnologías. El desarrollo se realizó desde el IDE Eclipse, el código fue escrito en Java, inicialmente se desarrolló un prototipo básico que aplicaba los conceptos fundamentales de los servicios REST, con la capacidad de recibir y responder peticiones exitosamente, estaba orientado únicamente al recurso evento. En este punto la información no persistía en ninguna base de datos. Las únicas funciones prestadas en este punto era la consulta de todos los eventos a manera de lista por medio de una petición GET y

la consulta individual de algún evento específico con una petición del mismo tipo, pero especificando el identificador de este en la URL.

4.2.2 Ciclo 2: Implementación de la base de datos e inclusión de servicios para los organizadores. En este segundo ciclo se introdujo MongoDB como base de datos encargada de la persistencia de la información, adicionalmente se desarrollaron los servicios para los organizadores, también fue necesario mejorar los servicios relacionados a los eventos.

4.2.3 Ciclo 3: Implementación de los servicios de usuario, optimización de los demás servicios, e inclusión de imágenes. Gracias al grado de apropiación de los conocimientos y las tecnologías, fue posible en este último ciclo desarrollar los servicios para los usuarios, además, optimizar los demás servicios. También se agregó una funcionalidad importante, que permite a la plataforma recibir, devolver y persistir imágenes en un sistema de archivos independiente a la base de datos, las cuales son cargadas al sistema únicamente por los organizadores, cabe mencionar que los servicios de carga verifican el peso de estas, para aceptar o rechazar la petición, si se da el primer caso el sistema determina a partir del peso si es necesario realizar una compresión de las imágenes. El conjunto de API´s que posee el sistema está documentadas.

5. VALIDACIÓN

Una vez finalizado el ciclo 3, se logró obtener el prototipo que satisface todas las necesidades funcionales, el cual fue sometido a pruebas de carga y estrés para evaluar su performance y determinar la calidad de los servicios. Cada una de estas pruebas las realizaremos para dos casos diferentes, una que involucre solo lectura de la base de datos y otra que requiera escritura en la base de datos.

Cabe mencionar que de acuerdo a las funciones que ofrece la plataforma, en un escenario real la mayor cantidad de peticiones serán hechas por los usuarios (interesados en los eventos), ya que la cantidad de estos será mucho mayor a la cantidad de organizadores, y así mismo las peticiones más recurrentes serán de tipo lectura de la base de datos, debido a que la función principal es consultar los eventos.

Para realizar estas pruebas usaremos la herramienta JMeter. Debido a que deseamos validar el rendimiento y la escalabilidad del sistema, los valores obtenidos de las pruebas que nos interesan son los tiempos de respuesta y los porcentajes de error, para a partir de estos realizar un análisis del performance del prototipo.

También es necesario tener en cuenta el ambiente en que se realizaron las pruebas.

5.1 CARACTERISTICAS DEL SISTEMA

- 6 GB RAM, DDR4 1200.5 MHz

- Disco Duro SATA 5400 rpm
- QuadCore Intel Core i5 8250U 1.60GHz 1.8GHz

5.2 PRUEBAS DE CARGA

Con este tipo de prueba intentamos determinar la cantidad de usuario concurrentes que soporta el prototipo, en un lapso de tiempo determinado, según los usuarios que se esperan atender, sin forzarlo a una capacidad mayor de la esperada, para efectos de esta prueba se establecen 3000 usuarios como los esperados en un lapso de 10 segundos.

5.2.1 Prueba de lectura. Para esta prueba simularemos un grupo de usuarios concurrentes en un determinado lapso de tiempo, en este caso específico establecimos 250 usuarios concurrentes en un lapso de 10 segundos, aumentando la cantidad de usuarios sumando 250 cada vez, hasta llegar a un máximo de 3000 usuarios**.**

En esta prueba enviaremos solicitudes GET para que el prototipo retorne el listado de eventos que deseamos, como se mencionó anteriormente, esta función será la más usada, por lo tanto, es primordial que se conserve la calidad de esta.

Tabla 2. Resultados prueba de carga - lectura

Número de usuarios	Tiempo promedio	Tiempo mínimo	Tiempo máximo	Error %
250	4	2	9	0.00%
500	3	2	113	0.00%
750	3	2	70	0.00%
1000	49	2	961	0.00%
1250	3	2	114	0.00%
1500	4	1	147	0.00%
1750	10	1	306	0.00%

Número de usuarios	Tiempo promedio	Tiempo mínimo	Tiempo máximo	Error %
2000	241	1	1600	0.00%
2250	156	2	1676	0.00%
2500	3	2	90	0.00%
2750	8	1	234	0.00%
3000	438	2	1869	0.00%

5.2.2 Prueba de escritura. En esta prueba enviaremos solicitudes POST, en el cuerpo del request enviamos los datos tipo JSON que deben ser insertados en la base de datos MongoDB. Debido al proceso de escritura podemos deducir que los tiempos de respuesta serán mayores que en la prueba anterior, lo que conlleva a un deterioro de los servicios, pero teniendo en cuenta que los servicios de escritura no serán los más usados, se realiza la prueba hasta un máximo de 1000 usuarios en el lapso de 10 segundos.

Tabla 3. Resultados prueba de carga - escritura

Número de usuarios	Tiempo promedio	Tiempo mínimo	Tiempo máximo	Error %
250	380	14	2494	0.00%
500	3946	9	10603	0.00%
750	11747	8	30538	0.00%
1000	22196	6	62990	0.00%

Los resultados de la tabla anterior demuestran que, aunque no se generan errores, si existe un deterioro en la calidad de los servicios, que se ve reflejado en los tiempos de respuesta.

5.3 PRUEBAS DE ESTRÉS

En esta prueba simularemos un pico de usuarios haciendo solicitudes, sometiendo el prototipo a condiciones de uso extremas, el objetivo es saturar la plataforma

hasta un punto de quiebre donde aparezcan errores. En este caso vamos a realizar solicitudes partiendo de 500 usuarios y aumentándolos cada vez esa misma cantidad, hasta que la plataforma no pueda responder las solicitudes

5.3.1 Prueba de lectura

Tabla 4. Resultados prueba de estrés - lectura

Número de usuarios	Tiempo promedio	Tiempo mínimo	Tiempo máximo	Error %
500	802	431	1396	0.00%
1000	1037	10	1519	0.00%
1500	944	8	1590	0.00%
2000	975	17	1638	0.00%
2500	1761	169	3017	0.00%
3000	4370	1741	7006	0.00%
3500	3310	55	5489	0.00%
4000	5760	2187	9103	0.00%
4500	4997	65	8291	0.00%
5000	29191	0	53002	82.98%

Con base en estos resultados podemos determinar que con un pico de 5000 usuarios la plataforma no es capaz de responder ni mantener la calidad de los servicios, aunque con 4500 si lo hace, lo cual es un buen resultado ya que los usuarios concurrentes que se esperan son 3000.

5.3.2 Prueba de escritura

Tabla 5. Resultados prueba de estrés - escritura

Número de usuarios	Tiempo promedio	Tiempo mínimo	Tiempo máximo	Error %
500	5454	39	11855	0.00%
1000	23986	34	66595	0.00%
1500	76346	22	242915	0.00%
2000	157257	98495	163896	90.00%

Como se esperaba en la prueba de escritura, los tiempos de respuesta son demasiado altos, aunque no presenten errores, pero ante una nueva oleada de peticiones si se podrían presentar. En este caso con 2000 usuarios se logró quebrar el sistema.

5.4 ANALISIS DE LAS PRUEBAS

Como se pudo observar en las distintas pruebas, se obtuvo un buen rendimiento en los servicios de lectura, siendo estos los más importantes, por ser lo más usados por el sector de usuarios masivos.

En los servicios de lectura la plataforma sufre deterioro en los tiempos de respuesta ante picos de usuarios muy altos, sin embargo, logra mantenerse libre de errores hasta un pico muy superior a la cantidad de usuarios concurrentes esperados, lo que garantiza la escalabilidad del sistema.

Los servicios de escritura se deterioran más rápido, tanto en la prueba de carga como de estrés, no obstante, estos servicios no serán usados de forma masiva.

6. CONCLUSIONES

- La plataforma backend aporta una solución ante el problema de unificar la información sobre los eventos y mejorar su difusión.
- Se implementó un prototipo software basado en la arquitectura planteada, que posee un subconjunto de API´s necesarias para enviar y acceder a información de valor para los diferentes usuarios.
- Se validó la funcionalidad del prototipo por medio de sus API´s, así como el performance del mismo a través de pruebas de carga y estrés.
- Las pruebas realizadas al prototipo arrojaron resultados satisfactorios, manteniendo la calidad de sus servicios frente a la cantidad de usuarios concurrentes esperados y su solidez ante picos de carga extrema.

7. TRABAJO FUTURO

- El prototipo fue desplegado en un ambiente no muy potente, mejorando este factor el performance ofrecido sería mucho mejor.
- Si existen demandas mayores a las esperadas o se desean implementar funciones que involucren escritura en la base de datos y sean de uso masivo, se debe implementar una infraestructura TI que responda y permita mantener la calidad de los servicios.
- En versiones futuras sería ideal añadir la seguridad a los requerimientos no funcionales.
- El objetivo fue desarrollar un prototipo funcional, por lo tanto no se profundizó en la seguridad, pero en versiones futuras sería ideal añadir esta a los requerimientos no funcionales.
- Sería interesante prestar el servicio de apartar la entrada, controlando esto por ejemplo con la generación de códigos QR

BIBLIOGRAFÍA

AJPDSOFT. Tomcat, Apache Tomcat, Jakarta Tomcat. [En línea]. (Recuperado el 2 de agosto 2019) Disponible en: http://www.ajpdsoft.com/modules.php?name=Encyclopedia&op=content&tid=769

APACHE TOMCAT®. Apache Tomcat. [En línea]. (Recuperado el 2 de agosto 2019) Disponible en: http://tomcat.apache.org/

ARQUITECTURAJAVA. Introducción a Servicios REST. [En línea]. (Recuperado el 2 de agosto 2019) Disponible en: https://www.arquitecturajava.com/servicios-rest/

CRISTALAB. Qué significa backend y frontend en el diseño web. [En línea]. (Recuperado el 2 de agosto 2019) Disponible en: http://www.cristalab.com/blog/que-significa-backend-y-frontend-en-el-diseno-web-c106224l/

EL TIEMPO. En Colombia hay 14,4 millones de usuarios de 'smartphones'. [En línea]. (Recuperado el 2 de agosto 2019) Disponible en: http://www.eltiempo.com/archivo/documento/CMS-15066597

EL TIEMPO. El uso de teléfonos inteligentes aumentó 50 % en Colombia en 2016. [En línea]. (Recuperado el 2 de agosto 2019) Disponible en: https://www.eltiempo.com/tecnosfera/novedades-tecnologia/en-50-por-ciento-crecio-el-uso-de-telefonos-inteligentes-en-colombia-89060

IMAGINANET. Creación de aplicaciones Restful con Jersey. [En línea]. (Recuperado el 2 de agosto 2019) Disponible en: https://www.imaginanet.com/blog/creacion-de-aplicaciones-restful-con-jersey.html

JSON ORG. Introducción a JSON. [En línea]. (Recuperado el 2 de agosto 2019) Disponible en: https://www.json.org/json-es.html

MEETINGSIMAGINED. El futuro de las aplicaciones de los eventos. [En línea]. (Recuperado el 2 de agosto 2019) Disponible en: http://es.meetingsimagined.com/tips-trends/future-event-apps

MICROSOFT. Datos no relacionales y NoSQL. [En línea]. (Recuperado el 2 de agosto 2019) Disponible en: https://docs.microsoft.com/es-es/azure/architecture/data-guide/big-data/non-relational-data

MICROSOFT. Diseño de API. [En línea]. (Recuperado el 2 de agosto 2019) Disponible en: https://docs.microsoft.com/es-es/azure/architecture/best-practices/api-design

MINISTERIO DE TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES. Tenencia de smartphones aumentó 50% en Colombia en el 2016. [En línea]. (Recuperado el 2 de agosto 2019) Disponible en: http://www.mintic.gov.co/portal/604/w3-article-51641.html

ORACLE. Java EE 7 y JAX-RS 2.0. [En línea]. (Recuperado el 2 de agosto 2019) Disponible en: https://www.oracle.com/technetwork/es/articles/java/java-ee7-y-jax-rs-2-2108434-esa.html

ORACLE. ¿Qué es la tecnología Java y para qué la necesito? [En línea]. (Recuperado el 2 de agosto 2019) Disponible en: https://www.java.com/es/download/faq/whatis_java.xml

SOPHIA.JAVERIANA Atributos de Calidad de Software. [En línea]. (Recuperado el 19 de junio 2017) Disponible en: https://sophia.javeriana.edu.co/~cbustaca/docencia/DEAS-2017-01/presentaciones/AtributosCalidadSoftware.pdf

UNIVERSIDAD DE PALERMO. [En línea]. (Recuperado el 19 de junio 2017) Disponible en: http://fido.palermo.edu/servicios_dyc/blog/docentes/trabajos/15609_51342.pdf

UNIVERSIDAD INDUSTRIAL DE SANTANDER. UIS en cifras 2016. Edición No 41. Julio 2017. ISSN 0120-212X