UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

UTILIZACIÓN DE PROCESOS ESTOCÁSTICOS EN APLICACIÓN DE BIG DATA

TESIS

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A

Daniel Vargas Sánchez

DIRECTOR DE TESIS

Dr. Daniel Trejo Medina

Ciudad Universitaria, Cd. Mx., 2017

Agradecimientos

A mis padres Margarita y Francisco, y a mi hermano Fernando, por su infinita paciencia y completo apoyo durante toda mi trayectoria, escolar.

A Ileana, por estar conmigo y apoyarme en las buenas y en las malas, y por acompañarme por este camino hacia mis metas personales y profesionales.

A mis amigos y a mis primos; David, Arturo, Casandra, Abigail, Antonio, Iván y a muchos otros, quienes siempre creyeron en mí, me apoyaron para salir adelante y me motivaron cuando lo necesitaba.

A la UNAM; mi alma máter, por todas las enseñanzas y experiencias que me hizo vivir, por todas las oportunidades que me brindó y por todas las personas a las que pude conocer gracias a ella, incluyendo a algunos profesores y a muchos estudiantes, quienes siguen siendo para mí grandes amigos.

A todos los profesores que me aportaron su conocimiento y compartieron conmigo sus experiencias profesionales, mismas que me hicieron llegar hasta donde estoy y me han impulsado a seguir adelante.

A Daniel Trejo, por creer en mí y por apoyarme durante todo el desarrollo de este trabajo, mismo que me abrió las puertas ante nuevas oportunidades de desenvolvimiento profesional.

A todos ustedes, gracias.

Índice general

CAPÍTULO 1. CONCEPTOS GENERALES 1

INTRODUCCIÓN 1
ANTECEDENTES 3
PROBLEMA A RESOLVER Y TIPO DE ESTUDIO 4
OBJETIVOS 5
Objetivo general 5
Objetivos particulares 5
JUSTIFICACIÓN 5
HIPÓTESIS 6

CAPÍTULO 2. MARCO DE REFERENCIA 7

ANTECEDENTES 7
Ingeniería de Software 7
Historia de la ingeniería de software 8
Administración de proyectos 10
Qué es la administración de proyectos 11
El proceso administrativo como estructura de la administración de proyectos 11
Definición de proyecto 15
Proyecto de software 17
Fases de un proyecto 18
Procesos aplicados a la administración de proyectos 19
Elaboración de un proyecto 25
Metodologías para el desarrollo de un proyecto 27
Metodologías ágiles 28
Manifiesto Ágil 28
eXtreme Programming (XP) 32
Scrum 33
Big data 41
Datos estructurados y no estructurados 47
Analíticos con Big Data 49
Procesos estocásticos y modelos auto-regresivos 60
Qué es un proceso estocástico 60
Tipos de procesos estocásticos 62
Modelos auto-regresivos 65
Aplicación en el método ARIMA 66
Tecnologías 67
Java 67

Servlets ... 68
La Apache Software Foundation ... 70
Apache Tomcat ... 72
Apache Maven ... 75
Spring ... 77
Modelo-Vista-Controlador ... 79
Java Server Faces ... 81
R ... 84
Programación en paralelo ... 86
METODOLOGÍA ... 91

CAPÍTULO 3. DESARROLLO ... 95

SELECCIÓN DE TECNOLOGÍAS ... 95
PROCESO GENERAL DE LA APLICACIÓN ... 99
ADMINISTRACIÓN DE DATOS ... 101
MODELADO Y ANÁLISIS ... 104
Implementación del modelo ... 105
Arquitectura ... 106
Predicción ... 112

CAPÍTULO 4. RESULTADOS Y CONCLUSIONES ... 113

RESULTADOS ... 113
CONCLUSIONES ... 115

BIBLIOGRAFÍA ... 121

ANEXO A. GLOSARIO DE TÉRMINOS ... 125

Índice de tablas

Tabla 1. Comparando métodos Ágiles contra métodos tradicionales. Fuente: Toro López (2013) 30

Tabla 2. Resumen de los procesos de Scrum. Fuente: adaptado de la Guía SBOK" (2013) 40

Tabla 3. Equivalencias en bytes de algunas unidades. Fuente: elaboración propia 42

Tabla 4. Versiones de Apache Tomcat. Fuente: adaptado de Apache Software Foundation (2016) 73

Índice de figuras

Figura 1. Fases, etapas y elementos del proceso administrativo. Fuente: Münich, Flores & Cacho (2014) 16

Figura 2. Impacto del riesgo, incertidumbre y el costo de los cambios en función del tiempo en la realización de proyectos. Fuente: Guía del PMBOK® (2013) 18

Figura 3. Interacción de los Grupos de Procesos. Fuente: Guía del PMBOK® (2013) 22

Figura 4. Interacción de los procesos dento de una fase o de un proyecto. Fuente: Guía del PMBOK® (2013) 22

Figura 5. Cinco Áreas de Conocimiento. Fuente: SWEBOK 2004 26

Figura 6. Seis Áreas de Conocimiento restantes (incluyendo a las disciplinas relacionadas con la ingeniería de software).Fuente: SWEBOK 2004 27

Figura 7. Flujo de trabajo en Scrum. Fuente: Guía Scrum™ (2013) 34

Figura 8. Interacción de los principales roles dentro de un proyecto Scrum. Fuente: Guía SBOK™ (2013) 37

Figura 9. Distribución de frecuencia de documentos que contienen el término "big data". Fuente: ProQuest LLC 43

Figura 10. Procesos utilizados para big data. Fuente: adaptado de Gandomi & Haider (2014) 49

Figura 11. Interacción de una aplicación que utiliza servlets/JSP. Fuente: Kurniawan, B. (2015) 70

Figura 12. Módulos de Spring Framework. Fuente: SpringSource (2016) 78

Figura 13. Interacción del modelo, la vista y el controlador. Fuente: Fernández & Díaz (2012) 81

Figura 14. Código que ejemplifica el scope de una variable en R. Fuente: adaptado de Ihaka (1998) 85

Figura 15. Salida del código mostrado en la Figura 13. Fuente: adaptado de Ihaka (1998) 85

Figura 16. Arquitectura de la aplicación. Fuente: elaboración propia 100

Figura 17. Proceso de ejecución de la aplicación. Fuente: elaboración propia ... 101

Figura 18. Tweet publicado por la Secretaría de Cultura de la Ciudad de México. Fuente: Adaptado de Twitter (2016) 103

Figura 19. Flujo de información para realizar un análisis predictivo. Fuente: elaboración propia 106

Figura 20. Arquitectura particular del módulo de predicción desarrollado. Fuente: elaboración propia 107

Figura 21. MVC dentro de la aplicación. Fuente: elaboración propia 108

Capítulo 1. Conceptos generales

Introducción

La generación de datos en diferentes sectores y organizaciones se ha incrementado de manera exponencial en estos últimos años. Esto se debe al creciente apego hacia los dispositivos computacionales que facilitan su elaboración. Los teléfonos inteligentes, computadoras portátiles, tabletas y widgets son algunas de las herramientas con las que se están generando minuto a minuto nuevos datos.

Derivado de lo anterior, la actividad de los usuarios de Internet ha ido creciendo significativamente, llevando hacia la publicación masiva de contenidos dentro de redes sociales, blogs, wikis, entre otros sitios. Además, los sectores financieros, de la salud, de registros hacendarios, de vivienda y de datos personales, se han sumado a esta revolución de producción masiva de contenido.

Esta explosión y elaboración de información ha llegado a un punto tal, en donde las herramientas computacionales tradicionales para el guardado, análisis e interpretación de datos son insuficientes e inadecuadas. Por lo cual, se han ido desarrollando y aplicando alternativas que permiten una gestión eficiente de estos grandes conjuntos de datos. Estas, hacen uso de la computación en la nube y las redes de datos de alta velocidad, del aumento paulatino de capacidad de procesamiento de los equipos de cómputo, de nuevos algoritmos que permiten resolver problemas complejos, de alternativas para el guardado de información distintas a las bases de datos relacionales tradicionales, tendiendo hacia las no relacionales, entre otros factores. Además, nuevas técnicas matemáticas se han aplicado sobre estos datos con el fin de realizar un análisis de información, proveyendo indicadores de valor que han apoyado la toma de decisiones basada en evidencia.

Otro inconveniente, inherente a la generación masiva de contenido, es que el 95% del total de los datos producidos se encuentran en una forma no estructurada, es decir, no cuentan con índices, palabras clave, relaciones entre sí, o algún otro

elemento que permita una clasificación puntual de los mismos (Gandomi & Haider, 2014).

Ante este panorama, los esquemas tradicionales para el guardado de información presentan ciertas limitaciones, tales como la velocidad de respuesta para el guardado, la clasificación y la recuperación subsecuente de resultados. Por ello, se han ido desarrollado nuevas soluciones, buscando solventar dichos inconvenientes. Estas nuevas herramientas abordan la problemática desde otro ángulo, permitiendo definir esquemas innovadores que soporten el guardado de información no estructurada. Es así como las bases de datos de tipo NoSQL aparecieron.

Dentro de estos almacenes de información, se permite un guardado más flexible de los datos, comparado con un esquema tradicional de base de datos, debido a que no existen relaciones entre los diferentes documentos persistidos. Sin embargo, esto puede derivar en dificultades de desarrollo posteriores, por lo que se vuelve indispensable analizar, sobre todo al comienzo del desarrollo, el tipo de almacén de datos a utilizar, pensando en los escenarios a los que será expuesta esta nueva aplicación.

Muchas de las diferentes soluciones pensadas para almacenar tales cantidades de información también tratan de abordar el guardado de contenido multimedia para su análisis posterior. Mientras tanto, para el guardado de información no estructurada de grandes bloques de texto, se encuentran herramientas completamente basadas en texto, las cuales permiten el guardado, análisis y extracción de información a una velocidad mucho mayor, comparada con la que podría aportar una solución relacional tradicional.

Además de lo anterior y derivado del tamaño de los datos, así como de la diversidad de los mismos, es preciso realizar algún proceso de limpieza, previo al consumo y a la utilización de estos. Por ello, se ha vuelto una necesidad que, antes de la aplicación de algún tipo de herramienta de análisis, se limpie esta información, definiendo reglas y nomenclaturas suficientemente claras para que un sistema computacional sea capaz de entenderla.

Ante tales escenarios, se ha ido materializando un término que los describe, así como a las soluciones, métodos y técnicas subsecuentes, capaces de hacerles frente y solventar las complicaciones venideras. Este anglicismo, que ya es sumamente conocido y utilizado habitualmente, es el denominado "*big data*".

Antecedentes

En el quehacer cotidiano dentro del área de las tecnologías de la información, en el que es necesario estar informado y actualizado en cuanto a las herramientas que entregan valor a terceros, es común encontrar el término *big data*. Por ello, se vuelve fundamental el conocer a qué se refiere, abriendo así la posibilidad de utilizarlo para entregar y mejorar soluciones, al mismo tiempo que para promover y aportar nuevos conocimientos, realizando así una continua mejora del mismo.

El desarrollo de la presente tesis toma como base a este supuesto, en donde se busca llevarlo hacia un entrono aplicado, además de plasmar la investigación inherente, en un esfuerzo por conocer con detenimiento parte de lo que abarca el fenómeno del *big data*.

Además, utilizando los conocimientos de ingeniería de software y administración de proyectos obtenidos en la Facultad de Ingeniería, se pretende desarrollar este documento siguiendo las buenas prácticas, desglosando también parte de las mismas.

Se parte del hecho de que un estudiante de ingeniería tiene el conocimiento matemático base sobre procesos probabilísticos, llevándolo hacia su utilización, empleando además herramientas big data para el manejo de grandes cantidades de información. Tomando este supuesto, se pretende implementar un módulo que pueda realizar predicciones de datos usando información histórica. Dicho histórico será utilizado para alimentar a un modelo matemático, quién realizará el procesamiento y análisis de información para entregar resultados de valor que ayudarán al usuario final con tomas de decisiones futuras.

La idea inició como parte de la continua mejora del software empresarial *Engine*, desarrollado previamente por una empresa privada. Dicha herramienta fue pensada como una solución para la administración del conocimiento que permite a las organizaciones aprovechar su conocimiento tácito, recuperado de las redes sociales y de diversas fuentes de noticias, para generar estrategias de atracción de clientes, ubicar sus áreas del mercado más provechosas e, inclusive, detectar focos rojos derivados de alguna mala estrategia de *marketing* o de difamaciones mediáticas, para así atacarlos y minimizarlos gradualmente.

La aplicación brinda al usuario analista la posibilidad de detectar índices de particular interés, mismos que serían complicados de encontrar mediante las técnicas de exploración convencionales.

Es por ello que, en un esfuerzo por optimizar las herramientas pertenecientes al software *Engine*, la organización exhortó a que algún alumno de la Facultad de Ingeniería ayudara en su construcción, al mismo tiempo que este realizaba el trabajo documental respectivo. Dicha propuesta, que puede verse como un común acuerdo en donde todas las partes ganan, derivó en la elaboración de la presente tesis.

Así, mientras la empresa antes mencionada continúa desarrollando herramientas de valor para sus clientes, genera en sus colaboradores, internos y externos, nuevos conocimientos, orientados hacia el crecimiento profesional conjunto, tanto dentro de la Faculta de Ingeniería de la Universidad Nacional Autónoma de México, como para todo el país.

Problema a resolver y tipo de estudio

Dentro de la aplicación de la ingeniería, modelos de ciencia de datos o actuariales, se tiene un gran campo de uso para el apoyo en la mejora de toma de decisiones.

El tener diferentes escenarios y poder predecir con mayor exactitud comportamientos futuros se ha convertido en una de las necesidades del denominado big data en forma aplicada.

Se pretende aplicar los procesos estocásticos para la predicción de eventos de una aplicación de big data.

Objetivos

Objetivo general

Mejorar la toma de decisiones en escenarios donde los resultados son parcialmente al azar y parcialmente basado en decisiones o evidencias previas, utilizando datos actuales, pasados y posibles condiciones futuras.

Objetivos particulares

Desarrollar un sistema que facilite la toma de decisiones con fuentes de datos abiertas y no estructuradas utilizando procesos estocásticos.

Justificación

Actualmente (2016), la empresa en caso cuenta con una aplicación que analiza grandes cantidades de datos, haciendo uso de algoritmos y herramientas big data. Dentro de ella se identifican “actores”, quienes fungen como los elementos centrales sobre los que es necesario obtener información.

En este software corporativo se incluyen diferentes módulos, útiles para un análisis integral de los datos, entre estos se encuentran: el de información geográfica, empleado para identificar la popularidad (qué tanto se habla de algún actor determinado) por zona; el de redes de vínculos, con el que se visualizan las relaciones que tienen los diferentes actores y cómo es que están relacionados; o el de análisis sentimental, que muestra un análisis de opiniones positivas, neutras y negativas, sobre alguna entidad de particular interés.

Sin embargo, se identificó que algunas otras herramientas de exploración de datos cuentan también con métodos de predicción, útiles para valorar el comportamiento futuro que tendrán, en este caso las opiniones generales y la popularidad censada.

Es por ello que, en un esfuerzo cooperativo entre la empresa, la Facultad de Ingeniería y el sustentante de la presente tesis, se busca cubrir esa necesidad, al mismo tiempo de brindar los conocimientos inherentes a la Facultad y apoyar en el crecimiento profesional de sus estudiantes.

Hipótesis

Si se desarrolla un sistema que haga uso de algún proceso estocástico para analizar grandes volúmenes de datos de fuentes abiertas, incluyendo redes sociales y medios de comunicación, se podrá obtener información de valor que ayude a una mejor toma de decisiones sobre eventos futuros.

Capítulo 2. Marco de referencia

Antecedentes

Dentro de la carrera de Ingeniería en computación, se obtuvieron diversos conocimientos base que permitieron desarrollar la presente tesis, las asignaturas que apoyaron a la construcción de la misma se listan a continuación, resaltando que estas las materias son las más representativas y que tienen una relación estrecha con el proyecto:

- Computación para ingenieros.
- Algoritmos y estructuras de datos.
- Programación avanzada y métodos numéricos.
- Introducción a la economía
- Probabilidad y estadística.
- Sistemas operativos.
- Ingeniería de software.
- Lenguajes de programación.
- Ética profesional.
- Administración de proyectos de software.

Ingeniería de Software

Cuando se desarrolla software en el entorno profesional, se da por hecho que este tiene como objetivo beneficiar a propósitos de negocio específicos, en donde el fin último es que dicho software se utilice por terceros y no únicamente por el programador. Dentro de este contexto, la elaboración de los productos de software se lleva a cabo por equipos de desarrollo en lugar de ser codificados individualmente.

En este ámbito, la ingeniería de software aparece como una herramienta fundamental para el desarrollo cooperativo de software empresarial, la cual brinda técnicas que apoyan a la especificación, diseño, desarrollo, validación, evolución de

la construcción y mantenimiento de un sistema. Además, esta lleva consigo no solo la metodología para construir programas, sino también toda la documentación asociada, incluyendo su descripción, la guía para el usuario final, entre otros factores determinantes, que permiten organizar y llevar un control preciso del desarrollo del software en cuestión (Sommerville, 2011).

Historia de la ingeniería de software

La historia de la ingeniería de software se remonta a 1968, año en el que la Organización del Tratado del Atlántico Norte (OTAN) realizó una conferencia para delimitar los retos y dificultades del desarrollo de grandes sistemas de software, buscando soluciones ante estos. Como resultado, surgieron herramientas para facilitar y automatizar la documentación de código fuente y pruebas relacionadas con la construcción de sistemas complejos.

Una década antes, habían aparecido grandes computadoras dedicadas a la investigación por parte de universidades e institutos. En aquel entonces, las computadoras eran utilizadas principalmente en el área de la ingeniería y de las ciencias naturales, aunque también comenzaban a ser indispensables en los negocios. Debido a su alto costo, estos sistemas fueron adquiridos por un pequeño número de instituciones y resguardados dentro de laboratorios cerrados, permitiendo ser operados principalmente por ingenieros eléctricos, siendo estos pertenecientes a la misma organización. Fue así como la profesión de programador apareció.

En los años siguientes, los programadores fueron surgiendo y siendo empleados por diferentes compañías. Estos eran los responsables del desarrollo y codificación de los primeros sistemas y algoritmos implementados. Para facilitar dicha codificación, ellos mismos establecieron notaciones formales que más adelante adoptaron el nombre de lenguajes de programación. El primer lenguaje de propósito general que apareció, desarrollado por IBM en 1957, fue Fortran, seguido de Algol en 1958 y Cobol en 1962, este último dado a conocer por el Departamento de

Defensa de los Estados Unidos y utilizado principalmente para aplicaciones de negocio (Wirth, 2008).

Conforme avanzaba la capacidad computacional, la demanda de programas cada vez más complicados se hizo también presente, haciendo más difícil y tediosa la labor del programador, por lo que los tiempos de desarrollo también se incrementaban. Ante esto, los lenguajes de programación subsecuentes buscaron ser más eficientes y fáciles de utilizar.

En 1963, el primer sistema de tiempo compartido[1] apareció, diseñado por John McCarthy en el Instituto Tecnológico de Massachusetts (MIT, por sus siglas en inglés), proporcionando una interacción que el *bash processing*, o procesamiento en línea de comandos, no permitía.

A partir de entonces, comenzó la transición hacia los sistemas de uso concurrente. Sin embargo, las complicaciones que esto conllevaba fueron mayores a las contempladas, llevando consigo retrasos en el desarrollo y problemas operacionales subsecuentes. A consecuencia, muchas grandes empresas se fueron a la quiebra.

La conferencia realizada por la OTAN en 1968 fue llevada a cabo en medio de esta situación, en donde se expusieron y discutieron los problemas a los que se enfrentaban los programadores y desarrolladores de estos grandes sistemas. Este momento en la historia de la computación fue denominado la “Crisis del Software”, y dio inicio a la Ingeniería de Software.

En la conferencia, las técnicas de desarrollo fueron modificadas y actualizadas, culminando en nuevas, más eficientes y metódicas, y que en poco tiempo pudieron ser adoptadas por toda la comunidad de desarrolladores (Wirth, 2008).

[1] Posibilidad de utilizar un recurso computacional de forma concurrente entre diferentes usuarios.

Administración de proyectos

Se debe tener en cuenta que para que un proyecto se desarrolle, deberán existir personas interesadas para la realización del mismo, así como procesos y metodologías, involucrando en todo momento el desarrollo de documentación.

Con relación a los procesos necesarios para la realización de un proyecto, se consideran elementos tales como las entradas, salidas (productos, resultados o servicios), técnicas, herramientas y equipo, activos organizacionales, personas e indicadores de desempeño. Además, es importante señalar que en todo momento se deberá tener la percepción de un valor agregado, ya que sin este se pierde todo el sentido de existencia del proyecto (Martínez & Chávez, 2010).

En general, durante el desarrollo de un proyecto es posible agrupar los procesos en cinco grupos principales: iniciación, planificación, ejecución, seguimiento y control, y cierre del proyecto. La administración de proyectos se basa en estos grupos. Cabe señalar que los procesos intermedios (planificación, ejecución y seguimiento y control) interactúan entre sí, lo que lleva a que, durante la ejecución del proyecto, será necesario volver a planificar y modificar la línea base las veces que sea necesario (Martínez & Chávez, 2010).

Otro punto importante para la realización y administración de proyectos, son las personas interesadas en que este se desarrolle. Una lista general de estos individuos, son en primer lugar, los clientes y usuarios, que pueden ser personas u organizaciones quienes utilizarán el producto una vez se termine de desarrollar.

Aunados a ellos se encuentran los patrocinadores, el equipo del proyecto, los gerentes funcionales y operacionales, los vendedores y los socios de negocios.[2]

[2] Trejo, Daniel. (ca. 2015). Planeación de proyectos informáticos. Curso presentado para el SUAYED de la F. C. A. Universidad Nacional Autónoma de México. México D. F.

Qué es la administración de proyectos

El término *administración de proyectos*, se puede definir como “La aplicación del conocimiento, habilidades, herramientas y técnicas […] para conocer los requerimientos del proyecto” (Darnall & Preston, 2010).

Para administrar correctamente un proyecto es necesario que exista un *Gerente de proyecto*, este es un individuo que cuenta con una serie de características, necesarias para llevar a cabo el proyecto. Esta persona debe tener la capacidad de: identificar y asegurar expectativas y necesidades de los *stakeholders* y de los patrocinadores del proyecto; comunicar de manera efectiva los mecanismos de iniciación de las actividades; Explicar a los miembros de un equipo las actividades que se deben realizar, acorde a un plan; establecer procesos de control del alcance de un proyecto; conocer, evaluar y reconocer las características de un trabajo que hay que realizar; comunicar el estado del proyecto, conforme se lleve a cabo, a los grupos de interesados (Toro López, 2013).

El proceso administrativo como estructura de la administración de proyectos

La administración lleva consigo una serie de etapas, cuyo conocimiento es esencial para aplicar el método, los principios, las técnicas y los enfoques de la gestión. Asimismo, en la administración de cualquier empresa existen dos fases: una estructural, en la que a partir de uno o más fines se determina la mejor manera para obtenerlos; y otra operacional, en donde se ejecutan todas las actividades necesarias para lograr lo establecido en el periodo de estructuración (Münch, Flores, & Cacho, 2014).

Existen diferentes criterios sobre el número de etapas que constituyen al proceso administrativo. No obstante, la mayoría de los autores coincide en que existen cinco: planeación, organización, integración, dirección y control. Estas se detallan a continuación:

- Planeación. En la etapa de planeación se determinan los escenarios futuros y el rumbo de a dónde se dirige la empresa. Además, se definen los

resultados que se pretenden obtener, a fin de definir las estrategias alcanzarlos, minimizando los riesgos que pudieran presentarse.

La etapa de planeación permite encaminar y aprovechar de una mejor manera los esfuerzos y recursos, reduce los niveles de incertidumbre en un proyecto y permite hacer frente a las contingencias. También, es un sistema racional para la toma de decisiones, evitando la ambigüedad, y sirve de punto de partida para llevar a cabo el control del proceso administrativo (Münch, Flores, & Cacho, 2014).

Lleva consigo los siguientes elementos, fundamentales para toda organización: filosofía de la empresa, valores, misión, misión, propósitos, premisas, investigación, objetivos, estrategias, políticas y programas. Además, el proceso está enfocado en la construcción de un objetivo, unificando diferentes ideas en busca de una meta concreta, y en el cambio de estrategias, permitiendo cierta flexibilidad para su implementación.

Para llevar a cabo la etapa de planeación se utilizan diferentes técnicas, pudiendo ser cuantitativas, como la elaboración de un análisis de tiempo denominado *Técnica de Revisión y Evaluación de Programas* (PERT, por sus siglas en inglés), como cualitativas, por ejemplo, un análisis de Fortalezas, Oportunidades, Debilidades y Amenazas (FODA) (Münch, Flores, & Cacho, 2014).

En un proceso administrativo existen diferentes tipos de planeación, sin embargo, es habitual referirse a dos de ellos: la planeación tradicional y la planeación estratégica (Bernal, Sierra, Cortés, & García, 2013).

En una planificación tradicional, algunos de los objetivos más trascendentales se plantean para un determinado tiempo, sin que existan análisis rigurosos dentro o fuera de las organizaciones, y sin que exista tampoco una definición precisa de los criterios que hay que seguir. Los directivos determinan estos objetivos en espera que, una vez se les hayan comunicado a los empleados, estos realicen el trabajo necesario para alcanzarlos.

Por otro lado, la planeación estratégica es un tipo de planeación que lleva consigo un proceso sistemático, fundamentado en conocimiento riguroso de la organización, interno y externo. Con ello se busca formular objetivos y estrategias, permitiendo a la organización diferenciarse de la competencia. La visión y misión de la empresa son elementos clave para este tipo de planeación.

- Organización. Aquí, se diseñan y determinan las estructuras, procesos, funciones y responsabilidades, así como el establecimiento de métodos y la aplicación de técnicas, buscando simplificar el trabajo a desarrollar.

  En la etapa de organización, se suministran los métodos para que se puedan llevar a cabo para realizar las actividades del proceso administrativo eficientemente. Con ello, los costos pueden disminuir, incrementando la productividad. Por último, reduce o incluso elimina la duplicidad de funciones que pudieran haberse considerado en un inicio.

  En la organización se busca dividir el trabajo mediante la jerarquización del personal y la división en departamentos, la descripción de las funciones y realizando un análisis de puestos. Además, en este proceso también se aboga por el trabajo coordinado eficientemente (Münch, Flores, & Cacho, 2014).

  Para esta etapa, se utilizan diversas técnicas, como los organigramas, manuales, diagramas de flujo procedimentales, documentos para la distribución del trabajo y el análisis de puestos.

- Integración. Dentro de este proceso se eligen y obtienen los recursos necesarios para poner en marcha los planes de algún proyecto. Comprende recursos materiales, humanos, tecnológicos y financieros.

  Los resultados que se obtienen durante la elaboración de un proyecto dependen de manera directa de los insumos del mismo, por lo que el realizar una integración correcta, estimando los recursos y los costos implicados, se vuelve fundamental. También, es preciso llevar a cabo la selección correcta de los recursos humanos, la especificación de las características, cantidades

y calidad de los recursos, incidiendo en el incremento de la optimización (Münch, Flores, & Cacho, 2014).

Para llevar a cabo la etapa de integración, se definen las necesidades y requerimientos de los recursos, los estándares de calidad y tiempos. También, se identifican a las fuentes de abastecimiento de recursos, se eligen proveedores y se seleccionan los recursos de acuerdo a diferentes estándares.

Esta sección tiene sus principios en el cumplimiento de los requisitos, obteniendo los recursos de proveedores confiables y sometiéndose a un proceso de adecuación, teniendo siempre en cuenta los estándares de calidad definidos (Münch, Flores, & Cacho, 2014).

Sus técnicas se centran en los recursos humanos, aplicando exámenes psicométricos, entrevistas y el análisis de puestos, buscando elementos para que sean parte de la organización.

- Dirección. Es un proceso que consiste en ejecutar todas las demás fases, mediante la conducción y orientación de recursos. Además, pone en práctica el ejercicio del liderazgo, aplicando los fundamentos de la misión y visión de la empresa.

  La dirección es la responsable de la toma de decisiones para ejecutar actividades en consecuencia, quien vigila y practica el buen comportamiento con las actitudes y conductas adecuadas, y tiene las cualidades para comunicar, influir, guiar y dirigir a diferentes grupos de trabajo (Münch, Flores, & Cacho, 2014).

  El papel que juega la dirección lleva consigo el aprovechamiento y resolución de conflictos, gracias a la supervisión constante de los diferentes elementos pertenecientes a la organización.

  Para llevarlo a cabo hace uso de herramientas cuantitativas, tales como modelos matemáticos o la construcción de rutas críticas en la elaboración de proyectos, y de cualitativas, como las tormentas de ideas, aprovechando la visión que tienen los diferentes elementos dentro de la organización en algún momento dado.

- Control. Aquí, se establecen estándares para poder evaluar los resultados obtenidos, con el objetivo de corregir desviaciones, prevenirlas y mejorar las operaciones continuamente (Münch, Flores, & Cacho, 2014).

  Esta etapa sirve para comprobar la efectividad de la gestión, asegura que los estándares de calidad se cumplan, protege los activos de la empresa y establece las medidas para revenir errores, reducir costos y el tiempo en la elaboración de proyectos.

  El establecimiento de estándares y de indicadores, su medición y corrección, así como la retroalimentación de los trabajadores son elementos clave en esta etapa del proceso administrativo. Asimismo, hace uso de diversas técnicas, tales como los sistemas de información, gráficas y diagramas, estudio de métodos, análisis de indicadores, control interno y diferentes métodos cuantitativos (Münch, Flores, & Cacho, 2014).

El proceso administrativo es una metodología básica para aplicar cualquier enfoque de administración de proyectos. El desglose anterior que describe a este proceso es uno de los más usados, pues ofrece una mayor claridad para su entendimiento. Un diagrama general de las cinco etapas, describiendo a su vez los elementos que conforman a cada una y la fase de la que forman parte, se ilustra en la Figura 1.

Definición de proyecto

De acuerdo con la Guía del *Project Management Body of Knowledge* (PMBOK), un proyecto se define como un "esfuerzo temporal que se lleva a cabo para crear un producto, servicio o resultado único" (Project Management Institute, 2013). Los proyectos pueden tener impactos sociales, económicos o ambientales que pueden perdurar mucho más que la elaboración en sí del proyecto.

Debido a que es temporal, este tiene un inicio y un final. El proyecto llega a su término cuando se cumplen sus objetivos, pero también cuando no se no se cumplirán o cuando la necesidad que dio origen al proyecto deje de existir. Asimismo, es posible que el proyecto llegue a su fin cuando el cliente, patrocinador o el líder de proyecto decida que sea momento de terminarlo.

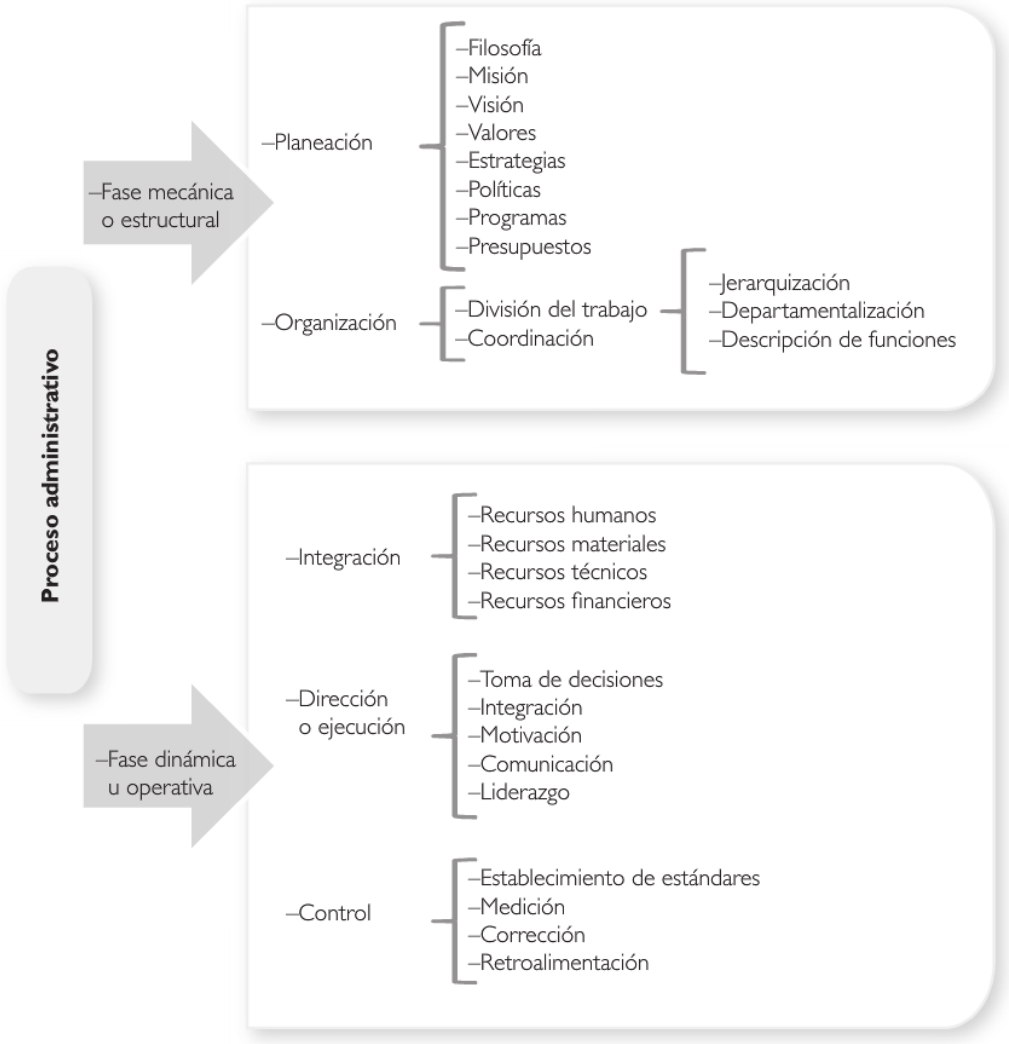

Figura 1. Fases, etapas y elementos del proceso administrativo. Fuente: Münich, Flores & Cacho (2014)

Todo proyecto tiene como finalidad generar un producto, servicio o resultado único, tangible o intangible. Pese a que puedan existir dentro del proyecto elementos repetibles e idénticos, estos no afectarán la naturaleza única y las características fundamentales del proyecto (Project Management Institute, 2013).

Ya que todo proyecto es único, existe cierta incertidumbre en su realización, debido a que en ellos se generarán productos y servicios diferentes. Por ende, el equipo de trabajo podría realizar actividades distintas en cada proyecto, lo cual requerirá una mayor planificación para su desarrollo. Además, este equipo de desarrollo puede

variar en tamaño, yendo desde un individuo hasta un grupo de organizaciones trabajando en común.

Proyecto de software

Según el IEEE [IEEE93a], la *ingeniería de software* se puede definir como: 1) “La aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software, es decir, la aplicación de la ingeniería de software.”. 2) “El estudio de enfoques según el punto anterior” (IEEE Computer Society, 2004).

En este sentido, se define a la ingeniería de software como una herramienta para el desarrollo sistemático de proyectos, así como para su mantenimiento, gestión y seguimiento. Bajo estos criterios, un *proyecto de software* es un tipo de proyecto en el cual se utiliza una metodología, se aplican diferentes procesos a cada elemento del desarrollo, y en donde cada uno de estos es mesurable.

Al tiempo empleado para la elaboración de un proyecto, comprendido desde la fecha de inicio hasta la de finalización, se le conoce como *ciclo de vida de un proyecto*, que es independiente del ciclo de vida del producto desarrollado. En él, se realizan las modificaciones, pruebas, mantenimiento, entre otras actividades, que conllevan a la correcta y eficaz realización del proyecto.

Los proyectos pueden variar en cuanto a su tamaño y complejidad. No obstante, para construir una definición de un proyecto es conveniente definir una estructura que cumpla para todos ellos. En la Guía del PMBOK® se definen los siguientes elementos, los cuales están presentes dentro ciclo de vida de un proyecto de software:

- Inicio del proyecto,
- Organización y preparación,
- Ejecución del trabajo y
- Cierre del proyecto.

Además, dentro de cualquier proyecto, existe riesgo e incertidumbre, así como costos asociados a su realización y cambios, y aunque es claro que estos elementos pueden variar dependiendo de diversas causas, todos ellos tienen comportamientos que pueden ser esquematizados y tomados como punto de referencia su elaboración. De manera general, los niveles de costo y dotación de personal son bajos en un inicio, pero estos van creciendo conforme se avance con la elaboración de un proyecto hasta alcanzar un punto máximo en donde decrecen rápidamente. Lo anterior entra en contraste con la incertidumbre y los riesgos, los cuales comienzan con niveles altos, pues no existe certeza de que se alcancen a cumplir los objetivos y no se sabe qué complicaciones se pudieran presentar, pero que conforme se avanza con el proyecto, estos decrecen. En la Figura 2 se ejemplifica lo anterior.

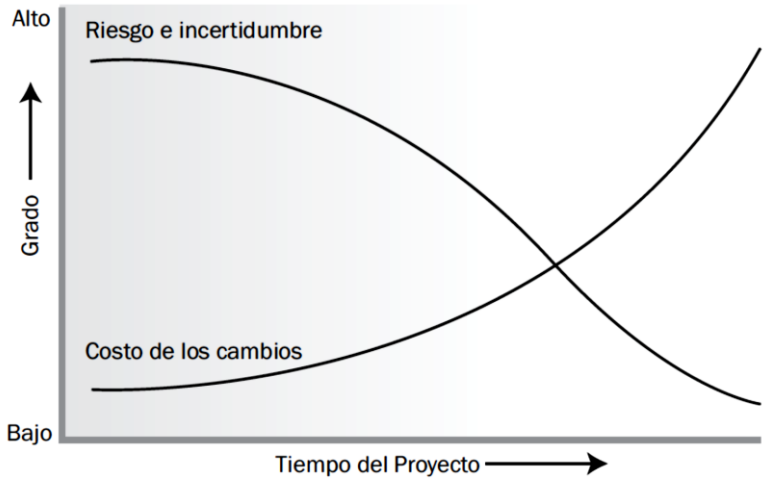

Figura 2. Impacto del riesgo, incertidumbre y el costo de los cambios en función del tiempo en la realización de proyectos. Fuente: Guía del PMBOK® (2013)

Fases de un proyecto

Sin importar el tipo de proyecto desarrollado, existen periodos de tiempo en donde se realizan actividades específicas, relacionadas de manera lógica, logrando con ellas un avance progresivo. A estos lapsos temporales se les denominan *fases de desarrollo*. Lo anterior aplica también para los proyectos de software, en los que es preciso definirlas y delimitarlas, y que en conjunto constituyen al ciclo de vida del proyecto en cuestión.

Las fases son comúnmente utilizadas cuando es necesario desarrollar algún nuevo producto, y su aplicación culmina generalmente con un entregable importante. Asimismo, una fase puede estar enfocada a procesos muy específicos, sin embargo, normalmente será necesario aplicar no solo unos cuantos, sino la mayoría de los procesos para cada una de las fases (Project Management Institute, 2013).

Gracias a las fases, se puede segmentar el trabajo en subconjuntos lógicos para mejorar su gestión, control y monitoreo, y pese a que el número de fases que se requieren para la elaboración de algún proyecto varía, dependiendo de las necesidades del mismo, existen algunas características comunes en todos los proyectos de ingeniería de software. Dentro de estas cualidades se enlista que para cada fase: el trabajo tiene un enfoque diferente y a menudo involucra a equipos y sectores de la organización distintos; del mismo modo, los objetivos o entregables que se buscan para el final de esta son únicos, por ello los grupos de desarrolladores y de procesos aplicables pueden cambiar; su cierre termina cuando se genera un entregable de la misma, en este punto es buen momento para realizar una revisión general de las actividades en curso, conocido también como *revisión de etapa*; en muchos casos, será necesario que su finalización sea aprobada para darla por cerrada. Además, por la naturaleza y la gestión de un proyecto, las fases de desarrollo utilizadas por diferentes organizaciones tienden a ser dispares, e incluso en una misma organización, estas varían dependiendo del producto que se busca elaborar.

Procesos aplicados a la administración de proyectos

Para gestionar debidamente un proyecto es necesario aplicar conocimientos, habilidades, herramientas y técnicas a las actividades de este. El llevar un control metodológico de estos elementos garantizará que un proyecto se desenvuelva de manera eficaz. Dicho desarrollo se llevará a cabo a través de procesos.

Un proceso es un “conjunto de acciones y actividades, relacionadas entre sí, que se realizan para crear un producto, resultado o servicio predefinido” (Project

Management Institute, 2013). Cada uno de estos lleva consigo entradas, herramientas y técnicas aplicables a este, y sus salidas obtenidas.

La correcta elección de qué procesos a utilizar será clave para que un proyecto sea exitoso. Además, otros factores importantes que son necesarios para concluir un proyecto son: utilizar un enfoque adecuado para cumplir los requisitos, establecer buena comunicación con los interesados, cumplir los requisitos para satisfacer las necesidades planteadas y delimitar adecuadamente las restricciones del proyecto.

En general, existen dos tipos de procesos; los orientados a la dirección de proyectos y los orientados a productos. Los primeros son aplicables al proyecto en sí, para que se desarrolle de manera eficaz a lo largo de su ciclo de vida, mientras que los segundos están enfocados directamente al producto desarrollado. Estos últimos están definidos y delimitados en el ciclo de vida del proyecto, y se debe tener en cuenta que para realizar una correcta definición de su alcance es necesario contar con la comprensión básica del cómo generar el producto final (Project Management Institute, 2013).

Cabe señalar que se hará énfasis en los procesos orientados a la dirección de proyectos más que a los enfocados a productos, debido al tipo de información documentada. No obstante, y aunque no se mencione de manera detallada, los procesos orientados a productos no deberían ignorarse en el desarrollo de un proyecto.

En todas las industrias se aplican procesos de dirección de proyectos, y si bien existe el concepto de *buenas prácticas*[3] para la gestión de estos, no quiere decir que se les deban aplicar todos los procesos si obedecen a necesidades y conceptos distintos. El administrador del proyecto y su equipo de trabajo deberán establecer cuáles son los procesos necesarios para la realización de uno nuevo, así como el

[3] Acuerdo general que indica que la aplicación de procesos en un proyecto aumenta sus posibilidades de éxito.

rigor con el que van a aplicar. A este proceso de análisis detallado y selección de los procesos se le conoce como *adaptación*.

Para su conceptualización y empleo, la Guía del PMBOK® establece cinco categorías en las que se agrupan todos estos procesos. A estas secciones se les conoce como Grupos de Procesos para la Dirección de Proyectos (o simplemente Grupos de Procesos). En ellas se detallan los siguientes:

1. Grupos de Procesos de Inicio. Procesos que se llevan a cabo para definir un nuevo proyecto o una nueva fase de este.
2. Grupos de Procesos de Planificación. Los cuales son utilizados para establecer cuál es el alcance del proyecto, así como demarcar el curso de acción para alcanzar los objetivos.
3. Grupos de Procesos de Ejecución. Procesos que se aplican a la realización de trabajos que a su vez sirven para alcanzar objetivos.
4. Grupos de Procesos de Monitoreo y Control. Que se utilizan para revisar, rastrear y regular el progreso, así como gestionar los cambios pertinentes del proyecto.
5. Grupo de Procesos de Cierre. Empleados para finalizar las actividades de todos los Grupos de Procesos y así cerrar formalmente el proyecto o una fase del mismo.

Estos procesos no son independientes entre sí; de manera general, las salidas de un proceso regularmente pasan a ser, ya sea, la entrada de otro u otros, o bien constituyen entregables intermedios llamados también *entregables incrementales*. Esto ocurre para todos los Grupos de Procesos a excepción de los de Monitoreo y Control, los cuales se encuentran presentes en cada uno de los demás, considerandos a estos como procesos “de fondo”. Una descripción gráfica de cómo se relacionan todos ellos se muestra en la Figura 3.

No obstante, no es necesario que un proceso termine para darle paso a los siguientes. De hecho, normalmente en un proyecto estos procesos se superponen y se repiten varias veces durante su ciclo de vida. Por ello, cabe aclarar que los

Grupos de Procesos no son equivalentes a las fases del ciclo de vida de un proyecto e incluso es probable que, por su naturaleza evolutiva, en cada una de las fases se desarrollen todos estos Grupos de Procesos. La interacción de los procesos dentro de una fase o un proyecto se identifica en la Figura 4.

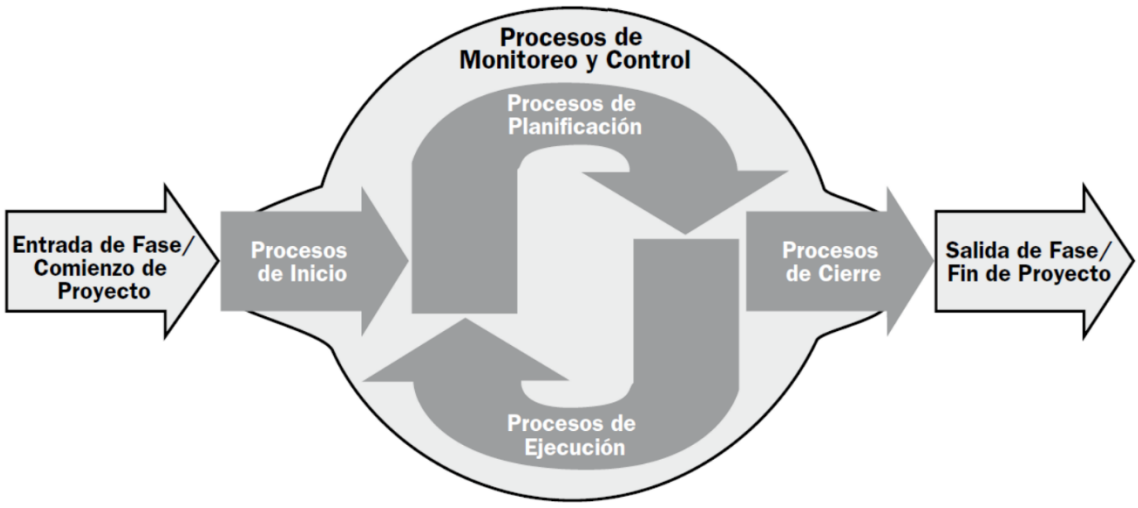

Figura 3. Interacción de los Grupos de Procesos. Fuente: Guía del PMBOK® (2013)

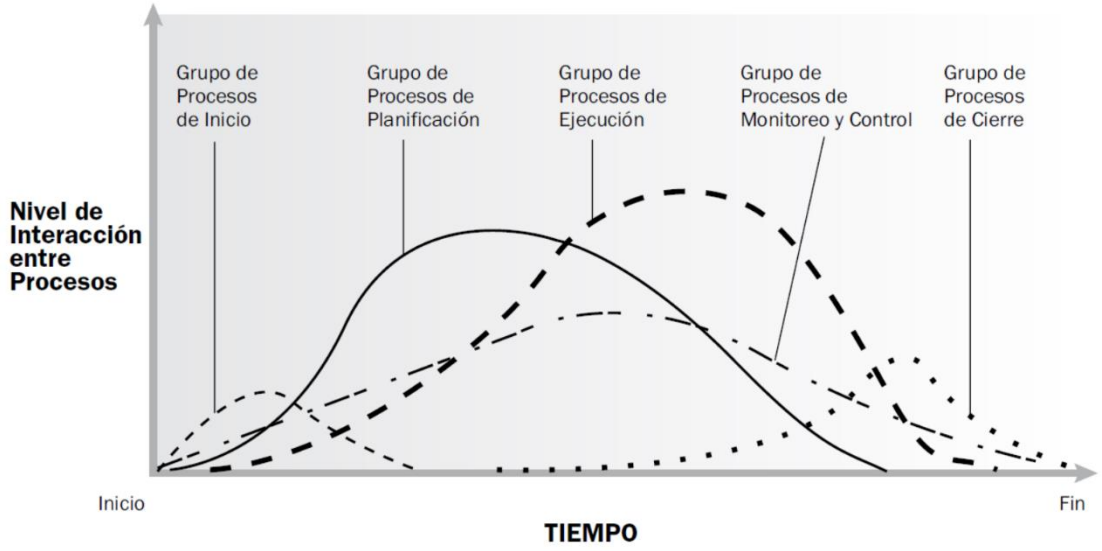

Figura 4. Interacción de los procesos dento de una fase o de un proyecto. Fuente: Guía del PMBOK® (2013)

Procesos de inicio

Este Grupo de Procesos está conformado por los procesos destinados a iniciar un nuevo proyecto o una nueva fase del mismo una vez se autorice. Aquí, se establece

el alcance inicial y se ponen a trabajar los recursos financieros iniciales. Además, se definen quiénes serán los interesados, tanto internos como externos, y si en este punto aún no se hubiera seleccionado al director de proyecto, se elige a uno. La información precisada se registra en el Acta de Constitución del proyecto[4] y en el registro de interesados (Project Management Institute, 2013). Cabe señalar que un proyecto se considera autorizado cuando se aprueba el Acta de Constitución del proyecto y aunque el equipo de dirección del proyecto pueda formar parte de la redacción del acta, la evaluación, financiamiento y aprobación del caso de negocio deben ser manejados fuera de los *límites del proyecto*[5].

Cuando se realizan proyectos de gran tamaño, es buena práctica dividir al proyecto en fases, como se mencionó anteriormente. Dentro de cada una de estas, se deben llevar a cabo los procesos de Inicio para mantener centrada la necesidad planteada, aprovechando también para verificar el trabajo realizado hasta el momento y definir entonces si se debe continuar, posponer o suspender el proyecto (Project Management Institute, 2013).

El involucramiento de todos los interesados desde el comienzo del proyecto asegura una mayor integración y por tanto un mayor entendimiento de los requerimientos. Además, un punto importante a destacar es que, dentro de estos procesos, se le otorga autoridad al director de proyecto para que utilice los recursos necesarios de la organización en las actividades posteriores al proyecto.

Procesos de Planificación

Los Procesos de Planificación desarrollan el plan de dirección del proyecto[6] y detallan los documentos requeridos para llevarlo a cabo. Normalmente, a consecuencia de la complejidad del análisis en esta etapa se necesite el uso de

[4] Proceso de desarrollar un documento que autoriza formalmente la existencia de un proyecto y confiere al director del proyecto la autoridad para asignar los recursos de la organización a las actividades del proyecto.
[5] El límite de un proyecto se define como el momento en que se autoriza el inicio o la finalización de un proyecto o de una fase de un proyecto.
[6] Se define un plan integral conjuntando diferentes planes secundarios para la dirección general del proyecto.

varios ciclos de retroalimentación para detallarlo adecuadamente. Además, conforme se avanza en la realización del proyecto, se le conoce con mayor detenimiento, e inclusive se realicen modificaciones de aspectos importantes, posiblemente exista la necesidad de realizar una planificación adicional. A esto último se le conoce como *elaboración progresiva* (Project Management Institute, 2013).

Pese a que las especificaciones realizadas en esta etapa probablemente sean modificadas frecuentemente, la ventaja de los Procesos de Planificación es que, como su nombre lo indica, *planifican* las tácticas y estrategias, así como la ruta que hay que seguir, para alcanzar los objetivos especificados al inicio.

La interacción con todos los interesados es fundamental, específicamente para obtener retroalimentación y detallar los documentos de planificación del proyecto. Sin embargo, este acto no puede prolongarse de manera indefinida, por lo que los procedimientos establecidos para una organización son los que dictan cuándo terminar la planificación inicial.

Procesos de Ejecución

En este Grupo, los procesos están destinados al trabajo realizado para alcanzar los objetivos del proyecto. Esto involucra coordinar a las personas y recursos, gestionar las expectativas de los interesados y en general, integrar todas las actividades a fin de cumplir y alcanzar las metas planteadas (Project Management Institute, 2013). Aquí, gran parte del presupuesto del proyecto será utilizado para su realización.

Procesos de Monitoreo y Control

Los procesos de esta etapa se encargan de analizar, rastrear y dirigir el progreso y desempeño del proyecto, a fin de identificar áreas en donde se requiera algún cambio para su mejora progresiva. Este tipo de procesos se realizan de manera regular cada cierto tiempo, buscando enfocar objetivos para que se alcancen las metas sin contratiempos.

El monitoreo continuo realiza observaciones en todos los niveles, resaltando cuáles son las áreas a las que se les debe prestar más atención, tanto en el equipo como

al proyecto, y sugiere entonces que se realicen en ellas técnicas para la mitigación de riesgos (Project Management Institute, 2013). Además, cuando se tienen proyectos de varias fases, este Grupo de Procesos ayuda también con su coordinación y seguimiento.

Procesos de Cierre

Este Grupo de Procesos es aplicable para cuando un proyecto o una fase del mismo terminan, para así cerrarlo formalmente. Para finalizar un proyecto será necesario revisar que todas las fases y procesos anteriores se hayan completado y una vez hecho esto, los procesos de Cierre lo declararán formalmente cerrado, considerando así que el proyecto o la fase han finalizado (Project Management Institute, 2013).

No obstante, pueden ocurrir circunstancias en las cuales el cierre del proyecto no sea como se espera, por ejemplo, por un cierre prematuro. Podría incluso ser tal que algunos procesos no puedan cerrarse formalmente (v. g. a consecuencia de reclamaciones y desapego del cliente). En tales casos, y debido a que los eventos son externos al ámbito del proyecto, estos se transfieren a otras unidades de la organización quienes se encargarían de ello, y una vez hecho esto será posible dar por finalizado al proyecto.

Elaboración de un proyecto

Un proyecto se puede realizar de diversas maneras. Sin embargo, para que existan buenas prácticas en el desarrollo de este, se debe contar con procesos en la construcción y el posterior mantenimiento del proyecto. Además, si se cuenta con una metodología, la planificación para llevar a cabo el proyecto se hará más metódica y por ende comprensible, al igual que su documentación.

Como se ha venido detallando, al desarrollar un nuevo proyecto, la administración de proyectos será un pilar fundamental para la realización del mismo. Dicha administración debería estar definida en toda organización que haya decidido adoptar un buen control de procesos, en donde la elección del modelo a seguir puede ser variada.

En el presente documento se han hecho referencias frecuentes a la Guía del PMBOK® del PMI en alusión a un buen modelo de administración. Sin embargo, existen otras alternativas. Una de las más representativas es el *Software Engineering Body of Knowledge* (SWEBOK) definido por el *Institute of Electrical and Electronics Engineers* (IEEE), el cual se utilizará como modelo para definir la estructura general de un proyecto.

Existen diez áreas se abordan en el SWEBOK tituladas *Áreas de Conocimiento* relacionadas al software, las cuales abarcan los requerimientos, diseño, construcción, pruebas, mantenimiento, gestión de configuración, gestión de la ingeniería, métodos y herramientas de la ingeniería, y calidad, todo lo anterior aplicado al software. En la Figura 5 se muestran de manera general las características de las primeras cinco áreas, mientras que la Figura 6 se visualizan las restantes. (IEEE Computer Society, 2004).

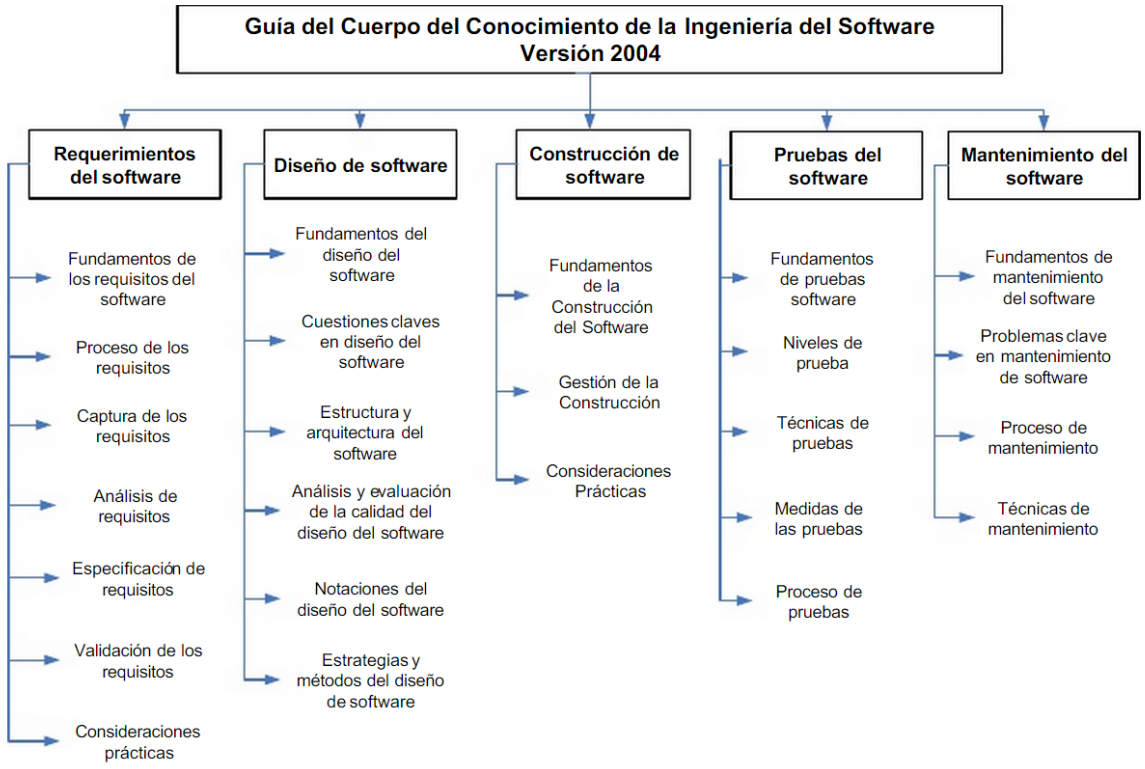

Figura 5. Cinco Áreas de Conocimiento. Fuente: SWEBOK 2004

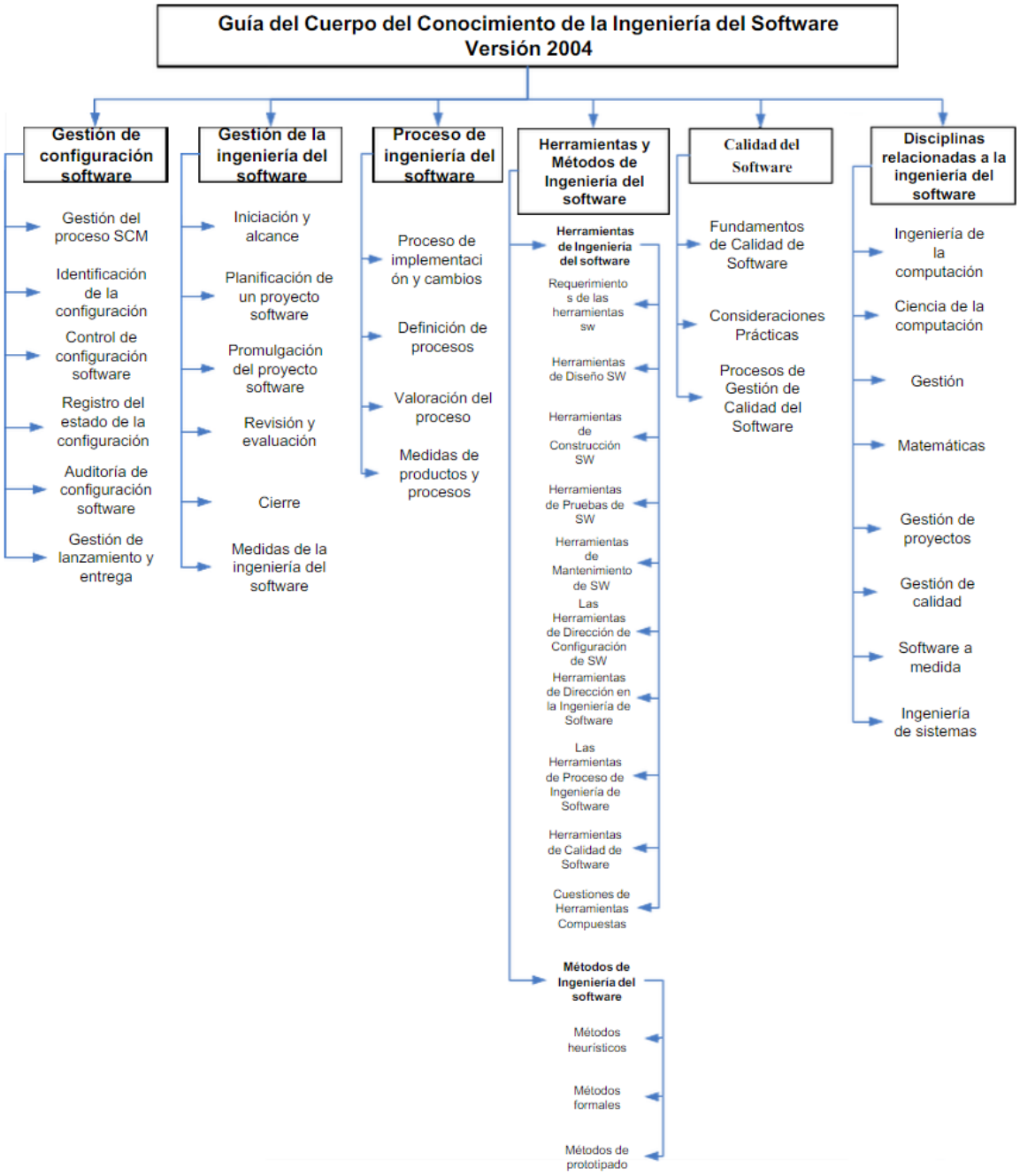

Figura 6. Seis Áreas de Conocimiento restantes (incluyendo a las disciplinas relacionadas con la ingeniería de software).Fuente: SWEBOK 2004

Metodologías para el desarrollo de un proyecto

Durante la planificación de un proyecto, es importante realizar el diseño de una metodología, pues en ella se especifica qué es lo que se va a producir y qué es lo que se hará para llevarlo a cabo. Además, se le puede considerar como una forma

estratégica de trabajo, por lo que se tienen que detallar los aspectos clave, para así sacarles el mejor provecho durante el desarrollo de dicho proyecto.

Bajo este contexto, una metodología se puede definir como "un sistema de prácticas, técnicas, procedimientos y normas utilizado por quienes trabajan en una disciplina" (Martínez & Chávez, 2010). Por lo que la elección de estándares se volverá crucial en el momento de definirla. Al hacerlo, se podrá tener un mejor control en las etapas subsecuentes de este proceso.

Para definir una metodología, se deben cohesionar dos términos importantes: método; forma de trabajo que implica un arreglo ordenado de manera lógica, generalmente con pasos a seguir (Martínez & Chávez, 2010); y ciclo de vida. De esta forma, será posible secuenciar todas las fases de un método de acuerdo con el orden establecido en el ciclo de vida del proyecto.

Metodologías ágiles

Un problema que surge a raíz de la aplicación de las metodologías tradicionales a los nuevos proyectos, consecuencia del riguroso manejo del tiempo e inflexibilidad, es que muchos equipos de desarrollo opten por dejar de prescindir de ellas, pese al riesgo que conlleva, pues las metodologías en cuestión no satisfacen las necesidades de los nuevos productos desarrollados.

Manifiesto Ágil

Ante tal preocupación, en febrero de 2001, un grupo enfocado a la mejora de modelos acuñó el término *ágil*, aplicado al desarrollo de software. Estos profesionales en el análisis y aplicación de modelos formaron un documento denominado *Manifiesto Ágil*. Su objetivo fue el construir una estructura que permitiera a los equipos de desarrolladores hacer frente a las necesidades cambiantes de los nuevos proyectos, así como a los tiempos cada vez menores para su construcción.

En el manifiesto se exponen alternativas a modelos más rígidos. Sus principios se basan en la entrega continua de software utilizable, en periodos que normalmente abarcan desde un par de semanas hasta varios meses, además de tener la capacidad de ser adaptable para así poder adaptar cambios en los requisitos del proyecto sin importar la fase de desarrollo en la que se encuentre.

Según el Manifiesto Ágil se valora más al individuo y a sus interacciones con los demás miembros, por lo que este se vuelve el principal elemento de éxito o fracaso en un proyecto de software. En este sentido, será necesario construir un buen equipo antes que definir el entorno en el que se llevará a cabo el desarrollo del proyecto (Toro López, 2013).

Particularmente para el área de desarrollo de software, existen gran cantidad de metodologías ágiles. Básicamente, estas se fundamentan en la sinergia *método - ciclo de vida*. Por lo cual, en ellas se combina el método de desarrollo de nuevos productos con el crecimiento sistematizado del mismo, a lo largo de todo el proyecto.

En contraste con la administración de proyectos aplicada por el Project Management Institute (PMI), con una administración de proyectos ágil se hará hincapié en la definición del objetivo de negocios del proyecto, así como en su propuesta de valor. Además, la metodología ágil llevará consigo una planificación ligera y una actitud adaptativa, su ciclo de vida será evolutivo y estará enfocado principalmente al desarrollo de nuevos productos (Martínez & Chávez, 2010). De esta forma se pretende disminuir sistemáticamente la incertidumbre y mitigar así posibles riesgos.

Las metodologías ágiles han demostrado ser más eficaces en el control del avance y en el manejo de expectativas, por lo que se han colocado como una conveniente alternativa para el desarrollo de proyectos informáticos. Además, son más flexibles en cuanto al control de todos los procesos, el modelado y la documentación, cosa que las distingue de las metodologías tradicionales, quienes al ser muy rigurosas suelen resultar efectivas con proyectos de gran tamaño, pero tienen dificultades al tratar de adaptarse a los nuevos proyectos con naturaleza cambiante, los cuales necesitan ser desarrollados en un lapso de tiempo menor y respetando los

entregables de calidad (Toro López, 2013). Los contrastes entre las metodologías ágiles y las tradicionales se ilustran de manera general en la Tabla 1.

Tabla 1. Comparando métodos Ágiles contra métodos tradicionales. Fuente: Toro López (2013)

Metodologías Ágiles	Metodologías tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (menos de 10 integrantes) trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Más artefactos
Pocos roles	Más roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos

La prioridad de las metodologías ágiles es satisfacer las necesidades del cliente, entregando avances o subproductos tempranos y continuos, ligados al desarrollo de software. Además, según las afectaciones que se presenten en el proyecto, son aceptados los cambios, pues con ellos el cliente busca obtener una ventaja respecto a su competencia.

El software presentado al cliente debe ser funcional, desde un par de semanas o un par de meses posteriores al inicio del proyecto, según sea el caso. Es preciso que las entregas de software al cliente sean en el menor tiempo posible. Para lograrlo, tanto la empresa de desarrollo como el cliente deben estar en constante comunicación a lo largo de todo el proyecto (Toro López, 2013).

En cuanto a los equipos de desarrollo, estos deben estar formados por individuos motivados, con un entorno de trabajo adecuado y el apoyo que necesiten. La mejor manera para dialogar con ellos, así como para comunicarles información, es mediante el diálogo cara a cara.

Debido a que la filosofía de las metodologías ágiles se basa en el desarrollo sostenible, es fundamental la existencia de un ambiente de paz constante y estable en el equipo de desarrolladores, así como con los promotores y usuarios (Toro López, 2013).

Se prioriza el software funcional en estas metodologías, por lo que se convierte en la medida principal del progreso del proyecto. En este sentido y por su naturaleza incremental, la simplicidad juega un papel sumamente importante en el proceso. Asimismo, la atención constante a la calidad técnica y al buen diseño, mejora la agilidad de su construcción.

Ya que los equipos se organizan por sí mismos, y al existir una sinergia entre cada individuo y la empresa de desarrollo, surgen mejores arquitecturas, requisitos y diseños por parte de estos. Por lo anterior, el mismo equipo es quien reflexiona respecto a cómo ser más efectivo, y según sus conclusiones adapta su comportamiento (Toro López, 2013).

Para que este tipo de metodologías puedan darse, será necesario que el gerente del proyecto adopte ciertas cualidades, como: cuestionar todo, al realizar preguntas que cuestionen la situación actual del proyecto para saber si se va por un buen camino; relacionarse con la innovación, ampliando sus conocimientos adaptándose a las nuevas tecnologías, al mismo tiempo de incentivar la creatividad e innovación; experimentar, probando elementos nuevos e innovadores para descubrir lo que funciona y lo que no, mejorando su capacidad de adaptarse, siempre buscando conseguir el éxito; ser creativo, buscando facilitar procesos y sobre todo, presentar precedentes con ideas nuevas; ofrecer valor, convirtiendo en hábito todo lo que se considere indispensable para así generar valor a la empresa; realizar cambios incrementales, para así lograr saltos cuantificables de productividad y agilidad constantemente; tener siempre en mente propósitos, de nueva cuenta para mejorar la creatividad y agilidad, así como para generar trabajo para las reservas del equipo (Toro López, 2013).

eXtreme Programming (XP)

El *eXtreme Programming* es otra metodología que surge a partir del Manifiesto Ágil, conocida también como *Fast Programming*. Cuenta con cuatro principios: 1) comunicación, 2) simplicidad, 3) retroalimentación (*feedback*) y 4) coraje; y 12 prácticas: 1) planeación, 2) pruebas (*testing*), 3) programación por pares, 4) refactorización (*refactoring*), 5) diseños sencillos, 6) propiedad colectiva del Código, 7) integración permanente, 8) el usuario al lado del equipo programador, 9) entregas frecuentes y pequeñas, 10) 40 horas de trabajo a la semana, 11) estandarización del Código y 12) metáfora (Toro López, 2013).

El equipo de desarrollo en la metodología XP está conformado por diferentes actores (Toro López, 2013), los cuales son:

- Programador. Quien produce el código del sistema y realiza las pruebas unitarias al mismo.
- Cliente. El que escribe las historias de usuario[7] y las pruebas funcionales para validar el correcto funcionamiento del sistema.
- Encargado de pruebas (*Tester*). Ayuda al cliente a escribir las pruebas funcionales. Ejecuta este tipo de pruebas regularmente, mantiene al tanto al equipo de los resultados obtenidos y es el responsable de las herramientas de prueba pertinentes.
- Encargado de seguimiento (*Tracker*). Proporciona realimentación al equipo. Evalúa el avance que se ha tenido con respecto a los objetivos y al tiempo estimado, para mejorar en futuras ocasiones. Es el responsable de realizar todo el seguimiento del progreso en cada iteración.
- Entrenador (*Coach*). Responsable del proceso global. Proporciona guías al equipo para reafirmar la metodología XP y para seguir al proceso tal como se tenía contemplado.

[7] Herramienta complementaria de las metodologías ágiles que especifica los requisitos funcionales y operativos de un producto de software.

- Consultor. Es un agente externo al grupo de desarrollo. Posee conocimiento específico para asesorar al equipo por si fuese precise.
- Gestor (*Big boss*). El elemento que relaciona a los clientes y usuarios con los programadores. Su labor es esencial en la coordinación pues busca que el equipo trabaje efectivamente y con las condiciones adecuadas.

Existen herramientas complementarias utilizadas para mejorar este tipo de metodologías, de entre ellas se encuentran las historias de usuario, quienes proporcionan información sobre el flujo de información y comunicación entre los módulos solicitados por parte del cliente y ayudan a los desarrolladores a entender de una mejor manera los requerimientos especificados. Estas historias, tienen que ser muy flexibles, haciendo posible su modificación y actualización, con un grado de complejidad no muy complejo, permitiendo que se realicen en periodos cortos de tiempo (Toro López, 2013).

Scrum

Es un modelo ágil que se centra en la gestión del proyecto más que en las prácticas de programación, en contraste con Extreme Programming (XP), por ejemplo. El avance del proyecto se basa en iteraciones para el desarrollo del mismo en las cuales se pretende realizar un avance progresivo, revisando y replanteando objetivos, buscando optimizar la forma de trabajo en cada ciclo.

Scrum es una metodología ágil de adaptación, iterativa, rápida y flexible, lo que la convierte en una de las más populares. Entre las fortalezas clave están los equipos multifuncionales, autoorganizados, con el poder de dividir el trabajo internamente, para realizarlo en ciclos de trabajo cortos y concentrados llamados *Sprints* (SCRUMstudy, 2013).

El ciclo Scrum comienza con una reunión entre los socios o *stakeholders*, en donde se crea la visión del proyecto. Es entonces cuando el propietario del producto, o *Product Owner*, realiza una *Prioritized Product Backlog*, que es una lista priorizada de todos los requerimientos en forma de historias de usuario. Cada *Sprint* comienza

con una Reunión de Planificación (*Sprint Planning Meeting*) en donde las historias de usuario de alta prioridad son consideradas para incluirlos dentro de dicho *Sprint*.

Un *Sprint* suele durar entre una y seis semanas, en donde el Equipo (*Scrum Team*) trabaja para la creación de entregables, derivados de avances significativos en la realización del producto. En la metodología se establece que, durante cada *Sprrint*, se deben llevar a cabo reuniones diarias, conocidas como *Daily Standup Meetings*, en donde los miembros del equipo exponen progresos diarios. Al término de cada ciclo, se realiza una *Sprint Planning Meeting*, para hacer una demostración al *Product Owner* y a los *stakeholders* de los bienes y servicios que se han alcanzado. Entonces, el *Product Owner* acepta las entregas si estas cumplen con los criterios de aceptación predefinidos. Una vez concluye esto, se realiza una reunión de retrospectiva, o *Spring Retrospective Meeting*, en donde el equipo analiza el trabajo llevado a cabo hasta el momento, pensando en la forma de mejorar los procesos y el rendimiento a medida que avanzan al siguiente *Sprint* (SCRUMstudy, 2013). Un diagrama general que representa estas interacciones se presenta en la Figura 7.

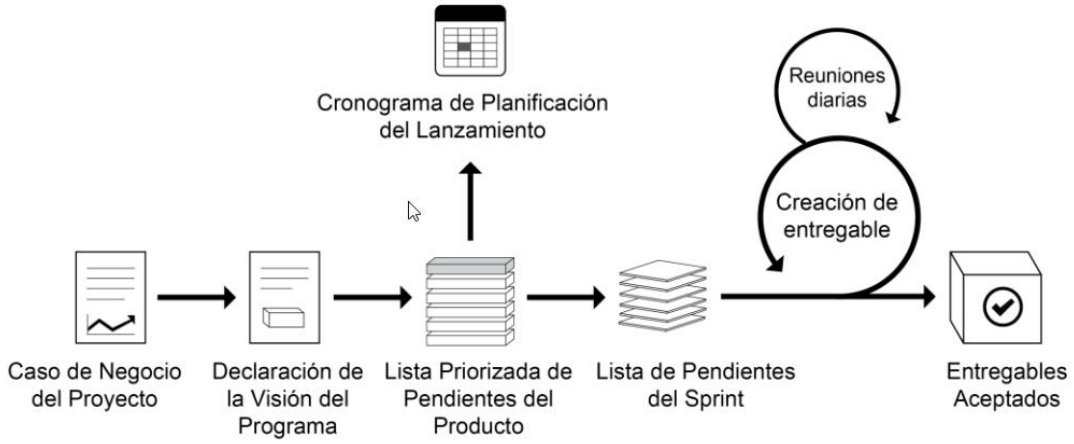

Figura 7. Flujo de trabajo en Scrum. Fuente: Guía Scrum™ (2013)

Para que sea eficaz, el *Scrum Team* debe tener de entre seis y diez miembros. No obstante, esto no quiere decir que utilizando este marco de trabajo solo sea posible desarrollar proyectos pequeños. La filosofía de esta metodología permite escalarla a proyectos más grande, en donde el equipo puede exceder fácilmente diez

personas. En este caso se conforman múltiples equipos de Scrum, proceso conocido como *Scrum of Scrum*.

Las áreas que abarca la metodología Scrum son:

- Seis principios.
- Cinco aspectos.
- Diecinueve procesos.

Principios

Los principios de Scrum se pueden aplicar a cualquier tipo de proyecto en cualquier organización. Estos no son negociables y se deben establecer tal como lo especifica la Guía SBOK™, no obstante, de acuerdo a los requerimientos del proyecto, los principios se pueden modificar para adaptarlos a las necesidades (SCRUMstudy, 2013). A continuación, se presenta cada uno:

1. Control de proceso empírico. Basado en las experiencias previas. Se basa en tres ideas principales: transparencia, inspección y adaptación.
2. Auto-organización. Implica que los equipos son auto-organizados, consiguiendo así un aporte de valor significativamente mayor por parte de estos.
3. Colaboración. Se centra en las tres dimensiones básicas relacionadas con el trabajo colaborativo: conciencia, articulación y apropiación, y aboga por la gestión de proyectos como un proceso de creación de valor compartido.
4. Priorización basada en valor. Como mecanismo para obtener el máximo valor del negocio priorizando entregables en cada *Sprint*.
5. Tiempo asignado. Restricción limitante para cada *Sprint*, empleada para manejar eficazmente la planificación y ejecución del proyecto.
6. Desarrollo iterativo. Lo que ayuda a gestionar de una mejor manera los cambios que se soliciten, además de desarrollar constantemente entregables para satisfacer las necesidades del cliente.

Aspectos

Al igual que los principios, estos se deben abordar y gestionar a lo largo de un proyecto para garantizar la correcta elaboración de este. La Guía SBOK™ detalla los siguientes:

1. Organización. En donde es preciso comprender cuáles son los roles y responsabilidades dentro de un proyecto Scrum. Existen dos clasificaciones generales: *Core Roles* y *Non-core Roles*.
    - Roles principales. Conocidos también como *Core Roles*, son papeles que obligatoriamente se requieren para desarrollar un producto. Quienes conforman a estos deben estar plenamente comprometidos con el proyecto y son responsables de su éxito. Dentro de estos están:
        - El *Product Owner*. Es la persona responsable de alcanzar el máximo valor empresarial del proyecto, de articular los requisitos del cliente y de mantener la justificación del negocio para dicho proyecto. El *Product Owner* representa la voz del cliente (*voice of the customer - VOC*).
        - El *Scrum Master*. Es un facilitador quien se asegura de que el equipo del proyecto cuente con el ambiente propicio para el desarrollo del mismo. Esto guía, facilita y enseña las prácticas de Scrum a todos los involucrados. Además, elimina impedimentos que pudieran presentarse, siguiendo los procesos Scrum.
        - El *Scrum Team*. Es un grupo de personas responsables de los requisitos especificados por el *Product Owner*, así como de la creación de los entregables en cada *Sprint* del proyecto.
    - Roles secundarios. Conocidos como *Non-core Roles*, son papeles que no son obligatorios, no ocupan ningún papel formal en el equipo del proyecto y aunque pueden interactuar con el equipo, no son responsables de su éxito. No obstante, estos roles se deben tener en cuenta en todo proyecto Scrum. En esta clasificación se encuentran los siguientes:
        - Interesados. Conocidos en inglés como *stakeholders*, en donde se incluyen a los clientes, usuarios y patrocinadores que interactúan con

frecuencia con el equipo principal de Scrum (*Scrum Core Team*) y tienen influencia en el proyecto a lo largo de su desarrollo. Lo más importante a tener en cuenta es que el proyecto produzca beneficios de colaboración para los *stakeholders*.

- Cuerpo de asesoramiento de Scrum. Consiste en un grupo de documentos y/o un grupo de expertos, estos guían el trabajo llevado a cabo por el *Product Owner*, el *Scrum Master* y el *Scrum Team*.
- Vendedores. Agentes externos quienes le ofrecen productos y servicios que no están dentro de la organización a esta última.
- *Chief Product Owner.* Este rol se utiliza cuando se desarrollan proyectos grandes. El *Chief Product Owner* es el encargado de coordinar el trabajo de los diferentes *Product Owners*.
- *Chief Scrum Master.* Es quien coordina todas las actividades de Scrum en grandes proyectos, donde se requiere que varios *Scrum Teams* trabajen en paralelo.

El diagrama de la Figura 8 muestra cómo se realiza la interacción entre los diferentes roles dentro de la metodología Scrum.

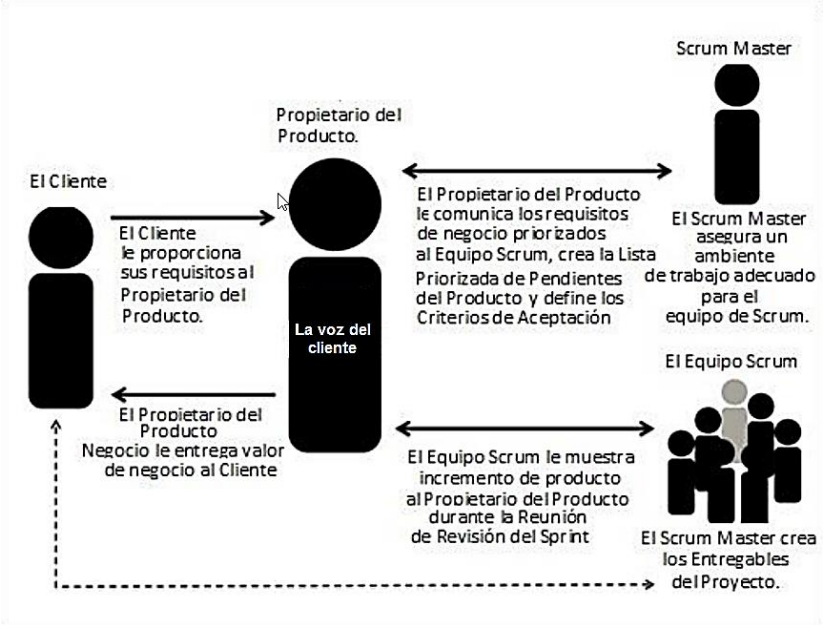

Figura 8. Interacción de los principales roles dentro de un proyecto Scrum. Fuente: Guía SBOK™ (2013)

2. Justificación del negocio (*Business Justification*). Antes de comenzar con cualquier proyecto es importante llevar a cabo una evaluación del negocio. Esto ayuda a comprender las necesidades de cambio en la empresa o en un producto, así como para tener siempre en cuenta por qué seguir adelante con el proyecto. Este aspecto toma en cuenta que en todo proyecto existe cierta incertidumbre que hace imposible predecir si el proyecto tendrá éxito. Para afrontar esto, Scrum realiza las entregas de resultados lo antes posible, consiguiendo así una oportunidad para consolidar un trabajo eficiente, ya que el cliente va conociendo el estado actual del producto, al mismo tiempo de aportar y demostrar el valor de dicho proyecto a los *stakeholders*.

   Además, debido a su naturaleza adaptativa, la metodología permite que los objetivos y procesos del proyecto cambien si cambia su justificación del negocio. El *Product Owner* es el principal responsable de dicha justificación.
3. Calidad. La Guía SBOK™ la define como la “capacidad del producto o de los entregables de cumplir con los criterios de aceptación y de alcanzar el valor del negocio que espera el cliente”. Para ello, Scrum incorpora un enfoque de mejoramiento continuo, en donde el equipo aprende de sus experiencias, y es posible actualizar el *Prioritized Product Backlog* con cualquier cambio.

   Debido a que el trabajo se realiza en *Sprints*, los errores y defectos son notados con facilidad a través de pruebas de calidad repetitivas, que se deben elaborar como parte de las labores en un *Sprint*. Estas pruebas de calidad para los avances del proyecto que son potencialmente entregables, se les conoce como *Done*.

   Por lo tanto, las pruebas repetitivas optimizan la probabilidad de alcanzar los niveles de calidad esperados. Las discusiones entre el equipo y so socios, junto con los incrementos reales, aseguran que la diferencia entre las expectativas y los entregables se reduzca constantemente.

4. Cambio. Todos los proyectos están expuestos a cambios, sin embargo, el marco de desarrollo de Scrum está diseñado para aceptarlos. Así, se trata de maximizar

los beneficios de dichos cambios y minimizar los impactos negativos que estos conllevan.

Estos cambios se deben a que los *stakeholders* pueden cambiar de opinión a lo largo del desarrollo del proyecto, y a que es muy difícil que dichos socios definan todos los requisitos desde un inicio. Los *Sprints* cortos y repetitivos aseguran que exista una retroalimentación del cliente en cada entrega, permitiendo que este interactúe regularmente con el equipo, vea entregables y modifique los requisitos si es preciso.

5. Riesgo. Este se define como “un evento incierto o un conjunto de eventos que pueden afectar a los objetivos del proyecto y pueden contribuir para su éxito o fracaso”. En este sentido un riesgo puede ser benéfico, en cuyo caso se le denomina “oportunidad”, pues tiene un impacto positivo en el proyecto. En contraparte, si el riesgo tiene un impacto negativo para el proyecto se le denomina “amenaza”.

   La gestión de los riesgos se debe hacer de forma proactiva dentro de un proceso iterativo. Siguiendo algunos pasos estandarizados, los riesgos pueden ser identificados y evaluados, para después responder a ellos basados en dos factores: la probabilidad de ocurrencia, y el posible impacto en el caso de que ocurran.

Procesos

Los procesos abordan las actividades y el flujo específico de un proyecto. En el caso de Scrum, existen diecinueve, agrupados en cinco fases, dentro de las cuales se describe cada uno con sus entradas, herramientas y salidas. Todas ellas se especifican en la Guía de SBOK™, en donde se distinguen las obligatorias de las opcionales. Las primeras, son importantes para la implementación exitosa de Scrum, mientras que las segundas dependerán del proyecto en cuestión. Las cinco fases que se describen en la Guía se enumeran a continuación:

1. Iniciar. En donde se Crea la visión del proyecto, se identifica a los integrantes del *Core Team*, se elabora(n) la(s) *Epic(s)*[8]*,* se crea una lista priorizada de pendientes y se realiza un plan de lanzamiento.
2. Planear y Estimar. Se elaboran y aprueban las Historias de Usuario[9] y el Criterio de Aceptación[10], diseñados para asegurar que los requisitos del cliente estén claramente representados y puedan ser claramente comprendidos por todos los *stakeholders*. Además, se elaboran, estiman e incluyen las tareas a realizar en una lista priorizada de pendientes (*Prioritized Product Backlog*) para un *Sprint.*
3. Implementar. El *Scrum Team* realiza entregables, se llevan a cabo los *Daily Standup Meeting* y se le da mantenimiento a la lista priorizada de pendientes acorde a los cambios.
4. Revisión y retrospectiva. Si se está elaborando un proyecto grande en donde interviene más de un equipo Scrum, se realiza una reunión (*Scrum of Scrum - SoS*) con los representantes de cada uno. El equipo realiza una demostración de los entregables del *Sprint* al *Product Owner* para que este los apruebe y acepte. Además, el *Scrum Team* realiza una retrospectiva de las lecciones aprendidas en dicho *Sprint*.
5. Lanzamiento. Se presentan los entregables ya aceptados a los socios relevantes, y se realiza una retrospectiva de todo el proyecto.

Los procesos pertenecientes a cada una de las fases descritas se presentan en la Tabla 2.

Tabla 2. Resumen de los procesos de Scrum. Fuente: adaptado de la Guía SBOK" (2013)

Fase	Procesos
Iniciar	1. Crear la visión del proyecto (*Create Project Vision*). 2. Identificar al Scrum Master y al socio(s) (*Identify Scrum Master and Stakeholder(s)*).

[8] Especificaciones generales, normalmente definidas al comienzo del proyecto, cuando los requisitos se han establecido de manera muy amplia. Se les puede considerar historias de usuario sin refinar.
[9] Manera simplista de la documentar requisitos y la funcionalidad deseada. Los requisitos expresados en forma de Historias de Usuario son afirmaciones breves, simpes y fáciles de entender.
[10] Marco establecido para validar las Historias de Usuario. Cada una de estas tiene su Criterio de Aceptación, proporcionando la objetividad necesaria para definir cuándo se han cumplido (*Done*).

	3. Formación de un equipo Scrum. 4. Desarrollo de Epic(s). 5. Creación de la lista priorizada de pendientes del producto (*Create Prioritized Product Backlog*). 6. Realizar el plan de lanzamiento (*Conduct Release Planning*).
Planear y Estimar	7. Elaborar historias de usuario (*Create User Stories*). 8. Aprobar, estimar y asignar historias de usuarios (*Approve, Estimate, and Commit User Stories*). 9. Elaboración de tareas (*Create Tasks*). 10. Estimar tareas (*Estimate Tasks*). 11. Elaboración de la lista de pendientes del Sprint (*Create Sprint Backlog*).
Implementar	12. Crear entregables (*Create Deliverables*). 13. Llevar a cabo el Standup diario (*Conduct Daily Standup*). 14. Mantenimiento de la lista priorizada de pendientes del producto (*Groom Prioritized Product Backlog*).
Revisión y Retrospectiva	15. Convocar Scrum de Scrums (*Convene Scrum of Scrums*). 16. Demostración y validación del Sprint (*Demonstrate and Validate Sprint*). 17. Retrospectiva de Sprint (*Retrospect Sprint*).
Lanzamiento	18. Envío de entregables (*Ship Deliverables*). 19. Retrospectiva del proyecto (*Retrospect Project*).

Big data

Big data, según su traducción más literal al español, significa “grandes datos”, y precisamente dentro de la Era de la Información, en donde cada vez es más frecuente escuchar y emplear este término, se vuelve fundamental entender todo lo que representa, las herramientas que ayudan a su interpretación y procesamiento, y, sobre todo, el valor que aporta a las diferentes organizaciones e individuos.

Cabe señalar que este término no se refiere a una ciencia o a una tecnología particular, sino a una tendencia, misma que se ha consolidado gracias a la enorme cantidad de datos generada día a día por cada ser humano en el planeta. Para dimensionar cuán grande es esto, la *International Business Machines Corporation* (IBM) realizó un estudio en donde se concluye que se están generando alrededor

de 2.5 quintillones de bytes en escala corta[11] día tras día (International Business Machines Corp., 2012). Y aunque *big data* no se refiere a una cantidad de datos en particular, la expresión es comúnmente utilizada al hablar de petabytes o hexaytes. En la Tabla 3 se ilustra la equivalencia en bytes de estas y algunas otras nomenclaturas.

Tabla 3. Equivalencias en bytes de algunas unidades. Fuente: elaboración propia

Unidad	Equivalencia en bytes
kilobyte (kB)	10^3 = 1,000
megabyte (MB)	10^6 = 1,000,000
gigabyte (GB)	10^9 = 1,000,000,000
terabyte (TB)	10^{12} = 1,000,000,000,000
petabyte (PB)	10^{15} = 1,000,000,000,000,000
exabyte (EB)	10^{18} = 1,000,000,000,000,000,000
zettabyte (ZB)	10^{21} = 1,000,000,000,000,000,000,000
yottabyte (YB)	10^{24} = 1,000,000,000,000,000,000,000,000

El nacimiento de esta expresión es inexacto, empero, algunas fuentes sugieren que se originó en la década de los 90, en una conversación dentro de la empresa Silicon Graphics Inc.,[12] (Gandomi & Haider, 2014). No obstante, fue hasta 2011 cuando su utilización comenzó a ser habitual. En la Figura 9 se presenta el número de documentos que contenían la palabra *big data* hasta 2013, incrementándose de manera exponencial en los últimos años.

Desde que se popularizó el anglicismo, se han empleado dos conceptos clave como una primera definición de lo que abarca; el almacenamiento y el análisis de datos. Adicionalmente, argumentos posteriores han ayudado a explicar de una forma mucho más completa lo que abarca este fenómeno.

[11] Sistema numérico en donde cada término es mil veces mayor que el anterior. En esta escala un quintillón equivale a la potencia 10^{18}.
[12] Compañía dedica a fabricar hardware y software de innovación.

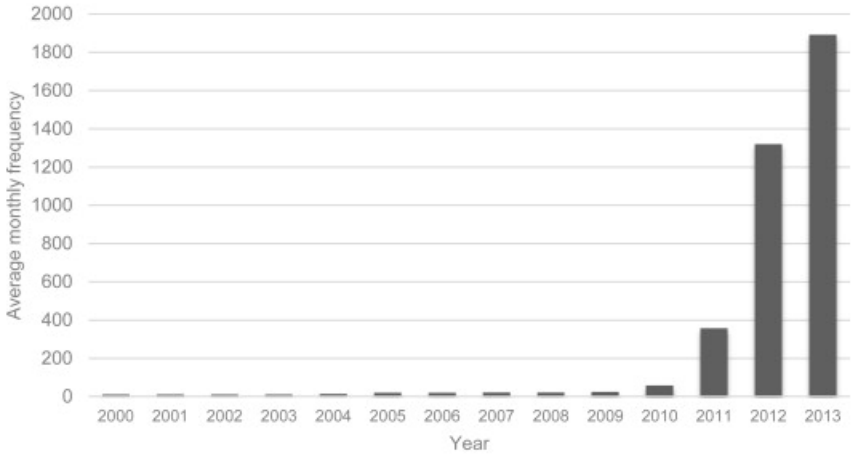

Figura 9. Distribución de frecuencia de documentos que contienen el término "big data". Fuente: ProQuest LLC[13]

Sin embargo, la misma velocidad con la que se extendió la palabra *big data* y los intentos subsecuentes por explicarla, dieron lugar a diferentes, muy variadas y comúnmente incompletas conceptualizaciones. Un análisis realizado en 2012 por SAP SE[14] en el que se les pregunta a diferentes ejecutivos su significado, dio lugar a lo siguiente:

- 28% de ellos dijeron que se refiere a un masivo crecimiento de transacciones de datos.
- 24% de los encuestados estuvieron de acuerdo en que se alude a nuevas tecnologías diseñadas ante el crecimiento de volumen, variedad y velocidad de información.
- 19% comenta que es un requisito de almacenamiento y archivado de datos siguiendo normas de regulación y cumplimiento.
- 18% dice que es la explosión de nuevas fuentes de datos, tales como redes sociales, dispositivos móviles y algunos otros.
- El 11% restante dio alguna otra definición.

[13] Compañía editorial que publica y suministra contenidos electrónicos a bibliotecas, promueve la investigación, el aprendizaje, la difusión y la gestión de información.
[14] Empresa alemana dedicada al diseño de software para el manejo de operaciones de negocios y relaciones con clientes.

Claramente, el tamaño de los datos es la primera característica a relucir dentro de estas descripciones, no obstante, no es la única. De hecho, entre las acepciones conocidas, una de las más comúnmente aceptada es la de las tres V. En esta se hace alusión al volumen, variedad y velocidad como parámetros fundamentales del *big data*. A continuación, se presenta una breve explicación de cada uno:

- El volumen se refiere al tamaño de los datos, comúnmente medidos en terabytes y petabytes, aunque también da lugar a otros factores tales como el tiempo de procesamiento o el tipo de información a analizar. Es preciso señalar también que, conforme avanzamos en el tiempo, las cantidades de información consideradas “grandes” no serán las mismas que las de ahora e inclusive dentro de diferentes industrias este término es relativo.
- La variedad apunta a la “heterogeneidad estructural en un conjunto de datos” (Gandomi & Haider, 2014), en donde se incluyen tanto a los estructurados, almacenados en hojas de cálculo o bases de datos y que comprenden solo el 5% del total de información existente; como a los no estructurados, que comprenden el 95% restante, tales como textos, imágenes, audio y video. En medio de estos se encuentra otra clasificación; los datos semiestructurados, quienes no se ajustan a las normas estrictas definidas para el almacenamiento de información estructurada, pero tampoco presentan el desorden de aquella no estructurada. Un ejemplo común son los documentos definidos con lenguaje de marcas extensibles (XML, por sus siglas en inglés), las cuales constan de etiquetas definidas por el usuario y que a su vez son interpretados por computadoras.

  Esta variedad de información no es nueva dentro de las organizaciones, pues la mayoría ha ido acumulando documentos procedentes de sus empleados y clientes. Y, gracias a la aparición de tecnologías capaces de gestionar y analizar estas fuentes documentales, se ha comenzado con su exploración y utilización, buscando mejorar e innovar dentro de sus procesos internos y de negocio.
- La velocidad, referente tanto a lo rápido con lo que se está generando información como al tiempo necesario para analizarla y aprovecharla. Esto

es derivado de la proliferación de dispositivos móviles (principalmente teléfonos inteligentes), de la utilización de sensores para el diagnóstico de fenómenos, entre otras cosas. Lo anterior conlleva a la necesidad de realizar análisis en tiempo real y una planificación basada en evidencia. Ejemplos de lo anterior se pueden encontrar en diferentes sectores; por citar alguno, la empresa Wal-Mart Stores, Inc., dedicada a la venta de productos de uso común, procesaba en 2010 más de un millón de transacciones por hora (Gandomi & Haider, 2014). Asimismo, múltiples corporaciones han aprovechado la generación de información en tiempo real para hacer llevar ofertas personalizadas a sus usuarios día tras día. Entre los datos recopilados se encuentra la ubicación geoespacial del usuario, los datos demográficos y los patrones anteriores de navegación y compras.

Si bien, se tiene bien definido a cada componente de las tres V, se debe entender también que estos elementos pueden estar interactuando entre sí. Adicionalmente, se han añadido algunas otras dimensiones que cubren áreas adicionales del *big data* (Gandomi & Haider, 2014), entre estas se encuentran:

- Veracidad, que fue incluida por IBM como la cuarta V y representa la fiabilidad de las fuentes de datos. Por ejemplo, el análisis sentimental realizado por las opiniones que realizan las personas a través de las redes sociales podría considerarse incierto, ya que es generado completamente por seres humanos. Sin embargo, esta sigue siendo información valiosa, por lo que la gestión y exploración de datos inciertos se convierte en otra característica primordial para el *big data*.
- Variabilidad y complejidad. La primera se refiere a una variación notable de datos en diferentes periodos de tiempo, generando en algunas épocas picos y valles de información (derivados de algún suceso sorpresivo e imprevisto), mientras que la segunda hace alusión al hecho de que la generación de información proviene de muchas y muy diversas fuentes. Ante esto, se vuelve fundamental realizar una recolección exhaustiva de datos para después limpiarla y homogeneizarla.

- Valor, concepto agregado por la empresa Oracle, quien lo define como un atributo definitorio del *big data*. La compañía argumenta que la información almacenada es comúnmente mucho mayor en cantidad de lo que puede aportar directamente, es decir, que cuenta con un bajo valor con relación a su volumen. Sin embargo, dicho valor puede incrementarse directamente al realizar un análisis posterior de estos conjuntos de información.

La descripción de las tres V es una de las más aceptadas para explicar el fenómeno, empero, se tienen también otros enfoques, inclusive a nivel industrial. Recorriendo las descripciones realizadas por diferentes organizaciones dentro del área de las Tecnologías de la Información, tenemos a Gartner Inc., quien refiere al *big data*, como el conjunto de volumen, velocidad, variedad y veracidad de datos; Oracle, quien lo interpreta como la derivación de valor a partir de la toma de decisiones, guiadas por la interpretación de diversas fuentes de información; Microsoft, quien lo describe como la aplicación de una significativa potencia de computación (v. g. aprendizaje de máquina e inteligencia artificial) a grandes y complejos conjuntos de datos; el proyecto MIKE (siglas en inglés de *Method for an Integrated Knowledge Environment*)[15] sostiene que tiene que ver con la complejidad más que con la cantidad de datos; y el Instituto Nacional de Estándares y Tecnología (NIST, por sus siglas en inglés) de EE. UU., afirma que es todo aquello que supera la capacidad de procesamiento de las herramientas computacionales actuales convencionales (Ward & Barker, 2013).

Reuniendo las aseveraciones anteriores para tratar de consensar una definición, *big data* se refiere a un gran volumen de datos, capaz de ser procesado a una velocidad aceptable para encontrar información de valor a través de herramientas y algoritmos computacionales cada vez más complejos, acotando la brecha entre los tipos de problemas computables y los no computables. Esto último debido a que una gran cantidad de problemas que se han querido resolver con ayuda de máquinas de cálculo con un gran poder de procesamiento (computadoras) pero que todavía

[15] Metodología de entrega, pensada para una eficaz gestión de información empresarial.

tenían como limitante el “tiempo máquina” implicado para realizar tales análisis, se han venido abordando con herramientas de *big data*, las cuales utilizan fundamentalmente la cooperación de decenas, cientos o incluso miles de computadoras (denominados clústeres de datos), algoritmos y mecanismos de software para solventar tales complicaciones.

La necesidad de búsqueda de nuevas técnicas para el análisis de datos es producto de la gran cantidad de información existente y que han sido generada sobre todo en esta última década. Un estudio realizado por IBM apunta que de 2010 a 2012 se produjo el 90% del total de datos que se tenían en este último año (International Business Machines Corp., 2012).

Empresas del sector público y privado, gobiernos, escuelas e institutos son algunos ejemplos de los involucrados en la producción de datos, quienes realizan operaciones de compra/venta día tras día, reúnen información acerca de sus clientes, tienen bases de datos con censos de población, registros médicos, impuestos, entre otros. Los bancos e instituciones financieras participan también, las redes sociales, los registros de compañías celulares; en prácticamente cualquier lugar del mundo, e inclusive fuera de él como en satélites, transbordadores o estaciones espaciales, se están generando datos, segundo a segundo, para múltiples y muy variados propósitos.

Cabe señalar que los seres humanos no somos los únicos responsables del crecimiento tan acelerado de información, pues con la comunicación máquina a máquina, que va desde la utilización de sensores para medición, monitoreo y control, hasta diversos mecanismos de inteligencia artificial, se ha convertido en una realidad que las computadoras también desarrollen por sí solas de datos de valor.

Datos estructurados y no estructurados

Esta generación masiva, veloz y hasta cierto punto descontrolada de información conlleva al manejo de herramientas de almacenamiento e indexación diferentes a las tradicionales, es decir, además de contar con las bases de datos relacionales,

se han comenzado a utilizar las no relacionales, las cuales están especializadas en los grandes volúmenes de datos, permitiendo realizar una escalabilidad sencilla y casi transparente, además de estar adaptadas para una fácil integración con clústeres de datos.

Bajo este supuesto surge otra interrogante: *¿de qué forma es preciso clasificar toda esta información?* Debido su diversidad, existen múltiples propuestas para realizarla y, dentro de estas, IBM propone la siguiente:

- Web y redes sociales. Contenido que se puede encontrar principalmente dentro de la red informática mundial (WWW, por sus siglas en inglés) y el generado dentro de redes sociales tales como Facebook, Twitter e Instagram.
- Maquina a máquina (M2M, por sus siglas en inglés). Tecnologías que permiten la interconexión entre dispositivos que realizan mediciones, se comunican entre sí a través de redes, las procesan y generan resultados.
- *Big Transaction Data*. Información en donde se incluyen las transacciones bancarias y de facturación, registros de llamadas de teléfono, entre otros. Este tipo de contenido es habitualmente guardado en formatos semiestructurados y no estructurados.
- Información biométrica. Aquí se incluyen patrones corporales propios de cada ser humano, por ejemplo, huellas digitales, retinas escaneadas, reconocimiento facial, e inclusive información genética.
- Datos generados por los propios seres humanos. Aquellos que se producen minuto a minuto en múltiples formatos, tales como notas de voz, correos electrónicos y documentos, entre otros.

Acorde a la creciente producción masiva de datos por parte de organizaciones, dispositivos autónomos e individuos, y debido también a que el tipo de información generada tiende hacia contenidos de video y de *streaming* (Gandomi & Haider, 2014), los diferentes sectores tienen que estar preparados para consumir y analizar estas fuentes de datos en tiempo real. Sin embargo, los sistemas de gestión de

datos tradicionales no están preparados para realizar esta labor de manera eficiente con tales cantidades de información. Ante esto, las tecnologías de *big data* entran en juego, permitiendo crear la infraestructura necesaria y dotar de inteligencia a los sistemas tradicionales, innovando a su vez en este sector, y abriendo paso a una nueva era para el análisis y procesamiento de información.

Analíticos con Big Data

Por sí solos, los grandes conjuntos de datos no tienen ningún valor. Su potencial sale a la luz cuando se realiza un análisis posterior que impulsa una toma de decisiones basada en la evidencia. El proceso para esto se puede dividir en cinco etapas (Labrinidis & Jagadish, 2012), ilustradas en la Figura 10, que a su vez están agrupadas en dos procesos: administración y análisis de datos. El primero es referente a las técnicas y tecnologías de apoyo para adquirir y almacenar datos, preparándolos para su consulta posterior, mientras que el segundo se refiere a la utilización de técnicas de análisis y aprendizaje a partir de los datos recolectados.

Figura 10. Procesos utilizados para big data. Fuente: adaptado de Gandomi & Haider (2014)

A continuación, se presentan las técnicas de análisis de datos comúnmente utilizadas en las que se involucran tanto datos estructurados como no estructurados.

Análisis de texto (*text minning*)
Se refiere a la extracción de información de datos textuales, tales como noticias, blogs, foros, encuestas, documentos corporativos, registros textuales de llamadas,

entre otros. Utiliza estudios estadísticos sobre el uso de palabras y frases, lingüística computacional[16] y aprendizaje, permitiendo a las empresas pasar grandes volúmenes de texto a resúmenes, apoyando así a la toma de decisiones basada en evidencia. Por mencionar una aplicación, esta técnica puede usarse para la predicción del mercado de valores mediante información extraída de periódicos y noticias financieras (Chung, 2014).

En seguida, se detallan algunos métodos utilizados para ejecutar esta técnica de análisis:

- Extracción de información (IE, por sus siglas en inglés). Se utiliza para obtener datos estructurados de texto no estructurado. Por ejemplo, con IE es posible recuperar la dosis, frecuencia y el nombre de un medicamento dentro de una receta clínica.

  El algoritmo empleado realiza, entre otras, dos sub-tareas; reconocimiento de entidades y extracción de relaciones (Jiang, 2012). Por ejemplo, dada la frase: “Steve Jobs, cofundador de Apple Inc. en 1976”, un sistema IE puede encontrar algunas relaciones entre entidades, tales como: *fundadorDe* [Steve Jobs, Apple Inc.] o *fundadoEn* [Apple Inc., 1976].
- Técnicas de resumen de texto. Empleadas para generar automáticamente un resumen de uno o varios documentos con la información clave de la(s) fuente(s) original(es). Se aplica para artículos científicos, noticias, anuncios, blogs y correos electrónicos, y básicamente sigue alguno de dos enfoques; extracción o abstracción (Hahn & Mani, 2000).

  En la extracción se crea un resumen a partir del texto original, compuesto generalmente de frases, para después recuperar las unidades de texto sobresalientes y encadenarlas. Estas son identificadas a través de su ubicación en el texto y su frecuencia, por lo que puede decirse que no se requiere una “comprensión” de qué es lo que se está diciendo. En contraste,

[16] Campo de estudio computacional que busca reducir la brecha entre los lenguajes basados en reglas y los que utilizan el lenguaje natural.

para la abstracción es preciso identificar la información semántica, produciendo además un resumen con unidades de texto que generalmente no están incluidas en la fuente original, por lo cual es preciso incorporar algoritmos avanzados de procesamiento de lenguaje natural (PLN) y conllevando a la generación de resúmenes más coherentes que lo de extracción. No obstante, dado su nivel de complejidad, frecuentemente se prefiere la utilización del primer enfoque sobre la del segundo.

- Búsqueda de respuestas (en inglés *Question Answering*, QA). Proporciona respuestas a preguntas que se formulan en lenguaje natural. Ejemplos de ello son Siri de Apple o Watson de IBM. Son muy útiles para la asistencia en diferentes sectores, como sanitaria, de finanzas o de educación. Al igual que la técnica de resumen por abstracción, requiere la utilización de mecanismos PLN complejos. Tiene tres vertientes; enfoque basado en recuperación de información, enfoque basado en conocimiento y enfoque híbrido (Gandomi & Haider, 2014).

  En el enfoque basado en la recuperación de información (IR, por sus siglas en inglés) se tienen a su vez tres componentes: el procesamiento de la pregunta, que determina detalles de la misma, tales como el tipo y el enfoque que tiene, para así generar una consulta interna; el procesamiento de documentos, quienes contienen elementos pre-escritos como posibles fragmentos de solución; y el procesamiento de la respuesta, en donde se extraen los posibles resultados de los documentos consultados, se clasifican y se devuelve la mejor solución encontrada.

  Para el enfoque basado en conocimiento se genera una descripción semántica de la pregunta para luego ejecutarse dentro de un conjunto de datos estructurados. Esto es particularmente útil en áreas para las que no existen grandes volúmenes de datos y, al contrario del primer enfoque en donde es necesario tener datos duplicados que dependen del contexto, en este no existe redundancia de información. Siri es un ejemplo de tecnología QA basado en conocimiento.

En enfoques de QA híbridos como Watson de IBM se entrelazan a los dos primeros; mientras que una pregunta se realiza semánticamente, los candidatos de respuestas se buscan a través de métodos de IR.

Análisis de sentimiento

Técnica utilizada para analizar texto que contiene opiniones de gente hacia productos, organizaciones, personas, entre otras entidades. Las empresas están utilizando cada vez más la información generada por sus clientes, con el fin de analizar el tipo de opinión que tienen sobre la organización y áreas como las de marketing, finanzas, ciencias políticas y sociales han visto una utilidad sumamente importante a este tipo de análisis.

Para entender cómo se implementa, es conveniente dividirlo en tres secciones dependiendo de su alcance, los cuales son: nivel de documento, nivel de frase y basado en aspectos. El nivel de documento es utilizado para determinar si un escrito completo conforma una opinión positiva o negativa, pudiendo también subdividirse en otras tendencias intermedias (Feldman, 2013). Para este nivel, se parte del supuesto de que dicho documento contiene sentimientos sobre una sola entidad. El segundo; nivel de frase, es empleado para determinar el sentimiento acerca de alguna entidad, analizando la fuente de datos enunciado por enunciado. Para ello, se debe distinguir entre las frases objetivas de las subjetivas, por lo que esta clasificación puede ser más compleja que la anterior. Por último, las técnicas basadas en aspecto identifican las diferentes tendencias de sentimiento dentro de un documento, asociándolas a su vez con las entidades (pudiendo ser más de una) a las que hace alusión el texto. Esto último es útil dado que, en muchas ocasiones, los comentarios realizados por los usuarios apuntan a más de una entidad, o bien, tienen diferentes observaciones para una misma. Con este último nivel se pueden encontrar las mejores características de un producto, a diferencia del primero, por ejemplo, en donde se perderían los puntos a favor dentro de un texto mayoritariamente negativo.

Dado el valor que aporta el análisis de sentimiento, más adelante se profundiza al respecto desde una perspectiva aplicada, partiendo de datos históricos de las

opiniones realizadas por usuarios de redes sociales y medios de comunicación, para realizar una predicción del sentimiento futuro, generando información de valor basada en evidencia, sumamente útil para el diseño de estrategias de negocio subsecuentes.

Análisis de audio

Como su nombre lo indica, consiste en la extracción de información de archivos de audio, aunque también se aplica al estudio del habla humana, técnica conocida como análisis de conversación. Los centros de atención telefónica o atención sanitaria son las fuentes primarias de datos en donde se aplican estas herramientas (Gandomi & Haider, 2014).

En los *call centers*, gracias al registro de miles o incluso de millones de horas de llamadas grabadas, es posible realizar análisis de audio para mejorar la atención al cliente y crear una mejor experiencia, evaluar el desempeño de los agentes, monitorear el cumplimiento de las políticas institucionales, entre otras actividades. Estos sistemas de análisis pueden emplearse incluso para llamadas en vivo, generando recomendaciones basadas en las interacciones pasadas, proporcionándole al agente información de valor en tiempo real.

Para el caso de la asistencia sanitaria, el análisis de audio es útil para el diagnóstico y tratamiento de ciertas afecciones médicas tales como la depresión, la esquizofrenia o el cáncer (Hirschberg, Hjalmarsson, & Elhadad, 2010). Inclusive, es posible analizar el llanto de un bebé para obtener información de valor sobre su salud y estado emocional (Patil, 2010).

Para realizar el análisis de audio dos enfoques son comúnmente utilizados; el primero, basado en la transcripción, conocido como *large-vocabulary continuous speech recognition* (LVCSR), y el segundo, fundamentado en la fonética (Gandomi & Haider, 2014). A continuación, se explica de manera general cada uno de ellos:

- Sistemas LVCSR. Siguen un proceso de dos fases: la indexación y la búsqueda. Para la primera, se intenta transcribir todas las palabras provenientes de audio a texto, mediante un proceso denominado

Reconocimiento Automático del Habla SR, por sus siglas en inglés), en donde se identifican los sonidos asociados con palabras, apoyándose en un diccionario predefinido, para devolver ya sea la palabra exacta o la más parecida encontrada. El resultado de esta fase es un índice de que contiene información sobre las palabras y la secuencia de las mismas. Para la segunda, se utilizan métodos de análisis de texto para encontrar información de valor sobre el índice generado.

- Sistemas basados en fonética. Se utilizan fragmentos de sonido o fonemas, que son "unidades perceptualmente distintas de sonido en un idioma determinado que distinguen a una palabra de otra" (Gandomi & Haider, 2014), por ejemplo, en el idioma inglés se tienen los fonemas /k/ y /b/ para diferenciar las palabras de *bat* y *cat*. En este tipo de sistema también se utilizan dos fases, la de indexación y la de búsqueda fonética. Para la primera, se descompone el audio en los fonemas que lo componen, y para la segunda, el sistema busca la representación fonética de lo que se pretende analizar dentro de la base de datos de audio generada.

Análisis de contenido de video (VCA, por sus siglas en inglés)

Utiliza diferentes técnicas para monitorizar y extraer información de archivos de video. Este tipo de estudio todavía se encuentra en una etapa prematura a comparación con otros. Empero, se han desarrollado diversas técnicas para rebatir a esta primera conjetura, empleando tecnologías *big data*, incluyendo el análisis de videos pregrabados como de aquellos transmitidos en tiempo real.

Este esfuerzo es derivado del aumento de cámaras de circuito cerrado (CCTV, siglas en inglés de *closed circuit television*), así como de la creciente tendencia por compartir este tipo de contenido a través de sitios web y de redes sociales. Sin embargo, un problema primario de este tipo de archivos es el gran espacio de almacenamiento demandado. Para poner esto perspectiva, se dice que un segundo de video en alta definición es equivalente a alrededor de 2 mil páginas de texto plano. Pese a esto, la utilización de este tipo de contenido no ha cesado; en abril de

2015, por ejemplo, se estimaba que la plataforma Youtube tenía una taza de subida de 100 horas de video cada minuto (Gandomi & Haider, 2014).

Una aplicación relacionada con la captura de video para la obtención de datos se da en los centros comerciales, en donde, gracias a la utilización de CCTV se pueden observar patrones de comportamiento de los clientes, desde que entran al recinto, conforme van realizando las compras, hasta llegar a las cajas registradoras para realizar el cobro. Este análisis se puede extender hacia el análisis de la edad o el sexo de los compradores.

La creciente generación de este formato de archivos ha derivado en el uso de técnicas que permiten realizar indexación automática en el momento del guardado y así poder recuperarlos fácilmente a futuro. Para ello, se utilizan diferentes niveles de información, que van desde los metadatos hasta el análisis visual de video, pasando por la recuperación de bandas sonoras y la transcripción textual del audio (Gandomi & Haider, 2014).

Básicamente, existen dos arquitecturas empleadas para realizar esta búsqueda de información; basadas en el servidor (*server-based*) y basadas en borde (*edge-based*). Para las primeras, la captura realizada se envía a un servidor centralizado y dedicado quien realiza el análisis. Aunque es el más económico, comúnmente se deben utilizar videos comprimidos, debido a las limitaciones de ancho de banda, lo que deriva en la pérdida de calidad conllevando a un análisis menos eficaz. En complemento, las de tipo *edge-based* el análisis se aplica en el "borde" del sistema, es decir, dentro de la misma cámara que realiza la grabación. Como resultado, el contenido en bruto está disponible para su análisis, sin embargo, este tiende a ser menos eficaz que el realizado por los del tipo *server-based*, debido a que el procesamiento computacional se ve mermado debido a las limitantes de no contar con un servidor dedicado.

Análisis de medios sociales

Se refiere a la obtención de información a partir de conjuntos de datos provenientes de medios sociales, entendiéndose como todas aquellas plataformas que permiten

a los usuarios generar e intercambiar contenido (Gandomi & Haider, 2014). Estos abarcan, entre otros, a: redes sociales, como Facebook o LinkedIn; blogs, como WordPress; microblogs, entre los que está Twitter y Tumblr; marcadores sociales, como Digg; de uso o alojamiento compartido de medios, en los que se encuentra Instagram y Youtube; wikis, como Wikipedia; sitios de preguntas y respuestas, como Yahoo! Respuestas y StackOverflow; y de revisión, como TripAdvisor. Además, aplicaciones móviles que sirven también para realizar una integración social pueden también considerarse como medios sociales, siendo WhatsApp una de las más utilizadas.

Aunque históricamente se puede rastrear el uso de una primera red social hasta mediados del siglo 20, su análisis subsecuente es relativamente reciente, nacido en la primera década del siglo XIX. La investigación es realizada por múltiples disciplinas, entre las que se encuentran: psicología, sociología, antropología, informática, matemáticas, economía e inclusive física.

El enfoque principal que se le ha dado al análisis de medios sociales ha sido orientado hacia el marketing de productos y servicios, debido a que en estos últimos años ha existido una adopción generalizada y en continuo aumento de las redes sociales a nivel mundial (He, Zha, & Li, 2013). Dentro de este, el contenido generado por el usuario (opiniones, imágenes, videos) y las interacciones entre entidades (personas, entidades y productos), han sido las fuentes primarias de información en las que se basa este análisis.

Dentro de esta área, existen dos clasificaciones en las que comúnmente se divide el estudio de datos; la analítica basada en el contenido y aquella basada en la estructura (Gandomi & Haider, 2014). A continuación, se detalla cada una de ellas:

- Análisis basado en el contenido. Se centra en los datos producidos por los usuarios dentro de las plataformas de medios sociales, como los comentarios que realizan hacia los productos que adquieren, así como a las imágenes y a los videos publicados. Las técnicas de análisis de texto, imágenes, audio y video son útiles para este primer grupo.

- Análisis basado en la estructura. Está enfocado en la información que se produce en las redes sociales, pudiendo identificar los tipos de relaciones que tienen los usuarios con sus conocidos, sus gustos, intereses, entre otros. El modelo resultante es una estructura conformada por nodos y enlaces, representando a los actores participantes y a las relaciones entre sí, respectivamente (Heidemann, Klier, & Probst, 2012).

  Dicho gráfico puede resultar muy útil para representar diversos tipos de información. En seguida, se analizan dos: los denominados gráficos sociales, y aquellos titulados gráficos de actividad. En los primeros, el enlace entre un par de nodos significa la existencia de algún tipo de vínculo, por ejemplo, la amistad, lo cual puede resultar benéfico en análisis en donde se busca encontrar actores con un número relevante de vínculos, directos o indirectos. Para los segundos, los bordes representan las interacciones reales, lo cual implícitamente refleja que ha existido un intercambio de información entre los actores. Debido a la información concreta que muestra este tipo de gráfico, es preferible en comparación con los sociales.

Derivado de la creciente necesidad de análisis de medios sociales, han aparecido nuevas técnicas capaces de realizarlo. A continuación, se describen brevemente las más utilizadas:

- Detección de comunidades. Es también llamado “descubrimiento de comunidades”, en donde se buscan patrones de comportamiento afines dentro de una red, obteniendo subconjuntos con usuarios que comparten características en común. Los tipos de usuarios son fácilmente identificados, pues interactúan entre si de una manera mucho más frecuente, comparada con el del resto de la red.

  Este tipo de búsqueda es utilizado para segmentar grandes conjuntos de datos, en los que normalmente se tienen millones de nodos y bordes, en pequeñas comunidades, ayudando a descubrir las conductas particulares de cada una, brindando también la posibilidad de realizar una predicción futura

de su comportamiento colectivo para solventar las necesidades emergentes que pudieran surgir (Aggarwal, 2011).

El descubrimiento de dichas comunidades se basa en la división de un conjunto en otros más pequeños, basándose en la similitud de los puntos de datos. Gracias a este, es posible hacer llegar a los usuarios productos eficaces, especializados en sus áreas de interés y despreciando aquellos que no les interesan (Parthasarathy, Ruan, & Satuluri, 2011).

- Análisis de influencia social. Se refiere a las técnicas aplicadas al análisis del comportamiento de los usuarios, influenciado por las opiniones de diferentes actores y de las conexiones dentro de la red, partiendo del supuesto de que el punto de vista de cada individuo se ve afectado por los comentarios y acciones de los demás. Ante esto, el análisis se esfuerza en evaluar el nivel de influencia de los participantes, verificando el grado de afianzamiento de las conexiones para así encontrar los puntos más relevantes e impactantes para la red.

  El marketing digital está haciendo uso de las técnicas de influencia social, buscando aprovechar la difusión de información publicitaria y mejorando en consecuencia la consciencia de la marca para una rápida y eficiente adopción (Gandomi & Haider, 2014).

- Predicción de enlaces. Aborda el problema de predecir futuros vínculos entre los diferentes nodos de una red. Esto debido a que un medio social no permanece estático, sino que se va actualizando y modificando a lo largo del tiempo, creando nuevos nodos y enlaces. El objetivo fundamental de este punto es contar con la información y métodos de análisis suficientes para comprender y predecir la dinámica de la red. Entre otros, se busca conocer la ocurrencia de interacción, colaboración, o influencia entre las entidades para un periodo de tiempo.

  Cabe señalar que, aunque mayoritariamente se están utilizando este tipo de tecnologías, dentro del área de medios sociales y de comunicación, estas pueden extenderse hacia otras áreas. Por ejemplo, dentro del campo de estudio de la biología, se pueden utilizar algoritmos de predicción de enlaces

para identificar asociaciones entre redes biológicas (v. g. interacciones entre proteínas), lo cual ayuda a reducir el costo dedicado a la investigación por pruebas experimentales que ya no tendrían que hacerse. En seguridad, se pueden identificar posibles relaciones entre redes terroristas o criminales, Finalmente, dentro de herramientas dedicadas al entretenimiento, la predicción de enlaces aporta sugerencias a los usuarios, dentro de sistemas como Youtube o Netflix (Liben-Nowell & Kleinberg, 2003).

Análisis predictivo

El análisis predictivo lleva consigo a una serie de técnicas que buscan realizar una estimación futura de resultados, basados en datos históricos y actuales (Gandomi & Haider, 2014). Prácticamente, se puede aplicar a casi todas las disciplinas, desde el comportamiento de dispositivos, hasta el de los compradores, tomando como base la información de lo que buscan, adquieren y cuándo lo hacen. Inclusive, medios de comunicación social e informativos también hacen uso de este tipo de metodologías.

Básicamente, se intentan descubrir patrones y relaciones entre los datos, utilizando técnicas que generalmente se dividen en dos; aquellas que intentan tomar como base a los patrones históricos de una variable para extrapolarlos a futuro, y las cuales en donde se recuperan las interdependencias entre la(s) variable(s) de salida y las explicativas[17], aprovechando tales relaciones para generar predicciones.

No obstante, existen otras clasificaciones utilizadas para definir el tipo de mecanismo empleado para un análisis predictivo. Por ejemplo, pueden identificarse también entre técnicas de regresión y técnicas de aprendizaje de máquina, o bien, entre las que entregan resultados continuos y discretos (Gandomi & Haider, 2014).

Independientemente de la clasificación y de sus técnicas subsecuentes, la mayoría está fundamentada en métodos estadísticos. Sin embargo, existen algunos puntos de debate entre la conceptualización tradicional de la estadística y la utilizada para

[17] Variables independientes.

grandes volúmenes de datos. En primer lugar, la premisa con la que se parte para los métodos estadísticos convencionales es que, basados en una muestra significativa de la población, es posible obtener resultados y conclusiones que pueden ser aplicados para el total de la misma. Lo cual, llevarlo a cabo dentro de un ambiente *big data*, puede que ya no se cumpla, pues se está trabajando con la mayoría o inclusive con la totalidad de la población. En segundo lugar y en términos de eficiencia computacional, los métodos utilizados para muestras pequeñas pueden resultar ineficientes al escalarlos hacia los grandes volúmenes de datos. En tercer lugar, existen características distintivas inherentes a los grandes conjuntos de datos, tales como la heterogeneidad, la acumulación de ruido, las correlaciones espurias, y la endogeneidad incidental, eventualidades que apuntan hacia nuevas técnicas estadísticas y probabilísticas para el análisis de información (Gandomi & Haider, 2014).

Procesos estocásticos y modelos auto-regresivos

Si bien es cierto que el estudio matemático aplicado al *big data* lleva consigo una continua mejora, esto no quiere decir que la fundamentación teórica y algoritmos derivados sean inútiles. Por ello, es preciso señalar que en la presente tesis no se pretende desarrollar un nuevo modelo, sino que se parte de la teoría existente, descrita a continuación.

Qué es un proceso estocástico

Un proceso estocástico se puede entender como un proceso de probabilidad, esto quiere decir que su evolución se puede analizar en términos de variaciones aleatorias, ya sea total o parcialmente. Para ejemplificarlo, se exponen dos casos presentes en la Ciudad de México (Mascareñas, 2013). El primero tiene que ver con los cambios de temperatura, la cual aumenta a lo largo del día y desciende durante la noche, siento este fenómeno prácticamente aleatorio pues depende de diversos factores meteorológicos, y es parcialmente determinista dado que se puede conocer su comportamiento general a través de la posición del sol. Lo mismo ocurre con el número de accidentes automovilísticos diarios, sucedidos en la Ciudad.

Un último ejemplo, útil para entender el tipo de fenómenos que se pueden modelar como procesos estocásticos, es el ciclo epidemiológico de alguna enfermedad; si un individuo infectado transmite dicha enfermedad a una comunidad aislada, durante un periodo de tiempo estas personas no presentan síntomas y, conforme estos se hacen visibles, se aíslan a quienes los demuestran hasta que se curan y se vuelven inmunes o mueren, la metodología y logística implícita en este proceso hace posible ubicar, con cierto grado de seguridad, la fase en la que se encuentra cada individuo en el mes n. Lo anterior es a lo que se le conoce como un proceso estocástico.

Como se aprecia, tales procesos sirven para representar una gran cantidad de sucesos de interés general, por lo que su estudio y utilización es cada vez más común. Además, son de particular interés pues no solo modelan el pasado, sino que pueden utilizarse para representar la influencia que los eventos pasados tienen sobre el presente, y hacer una aproximación de su influencia para el futuro. La definición formal (Rincón, 2012) se presenta a continuación:

"Un proceso estocástico es una colección de variables aleatorias $\{X_t : t \in T\}$ parametrizada por un conjunto T, denominado espacio parametral[18], en donde las variables toman valores en un conjunto ε, llamado espacio de estados".

Para el último ejemplo presentado anteriormente, el espacio parametral utilizado es el conjunto discreto $T = \{1,2, \dots\}$, representando los meses que han pasado después del brote. En este caso, se dice que el proceso es a tiempo discreto, los cuales, de manera general, son denotados como:

$$\{X_n : n = 0, 1, 2, \dots\}$$

O, explícitamente:

$$X_0, X_1, X_2, \dots$$

[18] Valores que puede tomar el tiempo.

Lo cual representa que, para cada n, X_n es el valor del proceso o el estado del sistema en el tiempo n.

Existen también otro tipo de modelados, en donde es necesario analizar el valor para cada instante de tiempo (Bressolf, 2014). En estos casos, el espacio parametral se toma como el conjunto continuo $T = [0, \infty)$ refiriéndose a este tipo de proceso como uno a tiempo continuo, quien se denota como:

$$\{X_t : t \geq 0\}$$

Tipos de procesos estocásticos

Con base en lo anterior, es posible identificar diferentes escenarios a analizar, los cuales son abordados con procesos estocásticos específicos. A continuación, se describen detalladamente algunos de los diferentes tipos de procesos:

Procesos de ensayos independientes

Para este tipo de modelos, el espacio parametral es discreto y las variables aleatorias son independientes entre sí. En este caso, se representa una sucesión de ensayos independientes de un mismo experimento aleatorio, donde el resultado de la observación realizada en el tiempo t es independiente de cualquier otra, pasada o futura. Un ejemplo de lo anterior ocurre al analizar el resultado de lanzar una moneda en repetidas ocasiones.

Procesos de Markov

En este caso, el estado de un sistema en el tiempo $t + 1$ depende únicamente del tiempo t y es independiente de los estados anteriores; $0, 1, \ldots, t - 1$. Los mismos son estudiados mediante cadenas de Markov, cuya definición (Ortega, 2014) se presenta a continuación:

Una Cadena de Markov a tiempo discreto es una sucesión de variables aleatorias X_n que toman valores en un conjunto finito o numerable ε, (espacio de estados), y que satisfacen que:

$$P(X_{n+1} = j \mid X_0 = i_0, \ldots, X_n = i_n) = P(X_{n+1} = j \mid X_n = i_n)$$

Esta ecuación es conocida como propiedad de Markov.

La probabilidad de que X_{n+1} esté en el estado j dado que X_n está en el estado n es llamada probabilidad de transición en un paso de n a j y se denota $P_i^{n\,n+1}_j$.

Para ejemplificar este tipo de proceso, se presenta el caso de una línea telefónica, en donde, de manera general, cuenta con dos estados: libre (0) u ocupada (1). Si llega una llamada mientras se encuentra ocupada, la llamada no se recibe, pero si llega cuando la línea está libre, esta se toma y pasa a estar ocupada. En este sentido, la probabilidad de que llegue una llamada es $\alpha \epsilon [0,1]$, mientras que la probabilidad de que la línea se desocupe es $\beta \epsilon [0,1]$. Para cada intervalo de tiempo, puede llegar una llamada o desocuparse la línea, más no ambas. Al modelar esta situación con una cadena de Markov, el espacio de estados es $E = \{0,1\}$, y las probabilidades de transición están dadas por:

$$P(X_{n+1} = 0 \mid X_n = 0) = 1 - \alpha = P_{0\,0}$$

$$P(X_{n+1} = 1 \mid X_n = 0) = \alpha = P_{0\,1}$$

$$P(X_{n+1} = 1 \mid X_n = 1) = 1 - \beta = P_{1\,1}$$

$$P(X_{n+1} = 0 \mid X_n = 1) = \beta = P_{1\,0}$$

Procesos estacionarios

De manera general, un proceso estocástico estacionario es aquel que se mantiene constante a través del tiempo, dentro de un margen estable de oscilación (Ruíz, 2006). Hay dos tipos de procesos estacionarios:

1. Procesos estacionarios en sentido estricto. Conocidos comúnmente como procesos estacionarios. En este tipo de procesos, al realizar un mismo desplazamiento en el tiempo de todas las variables de cualquier distribución conjunta finita, resulta que esta distribución no varía, es decir:

$$F\left(X_{i_1}, X_{i_2}, \dots, X_{i_r}\right) = F\left(X_{i_{1+j}}, X_{i_{2+j}}, \dots, X_{i_{r+j}}\right)$$

para todo $j \geq 0$ y todo conjunto de índices $i_1, i_2, \ldots, i_r$ donde $i_1 < i_2 < \cdots < i_r$, y $F(X_i)$ representa la función de distribución conjunta de las variables aleatorias.

2. Procesos estacionarios en sentido débil. En este tipo de procesos se cumple que, para todo t en el espacio parametral,

$$\mu_t = \mu$$
$$\sigma_t^2 = \sigma^2$$
$$Cov(t, t+j) = Cov(s, s+j) = \gamma_j \text{ para } j \in \mathbb{Z}$$

Donde:

μ_t es la función de medias, dada por $\mu_t = E(X_t)$

σ_t^2 es la función de varianzas, $\sigma_t^2 = Var(X_t)$

$Cov(t,s) = Cov(s,t) = Cov(X_s, X_t)$ es la función de auto-covarianzas

Esto quiere decir que las características del proceso se mantienen constantes a lo largo del tiempo.

Procesos con incrementos estacionarios

En estos procesos, dado $t \in T$ fijo, la distribución de $X_{s+t} - X_s$ es la misma para todo $\{s \in T \mid s + t \in T\}$. Esto significa que cambios de igual tamaño son iguales en distribución.

Procesos de ruido blanco

Son un tipo de proceso estacionario que cumple las siguientes propiedades:

- Tiene media cero, i.e. $\mu = E(\varepsilon_t) = 0$
- Su varianza es constante: $Var(\varepsilon_t) = \sigma^2$
- Las variables aleatorias son independientes a lo largo del tiempo:

$$Cov(\varepsilon_t, \varepsilon_s) = 0 \text{ siempre que } t \neq s$$

Es decir, un proceso de ruido blando es una sucesión de variables no relacionadas entre sí, que oscilan alrededor de un margen constante, donde los valores pasados no tienen ninguna influencia en los futuros, por lo que conocerlos no proporciona información sobre el siguiente estado de tiempo: el proceso es “puramente aleatorio”

(Ruíz, 2006). De este modo, todos los coeficientes de correlación serán 0, excepto en el caso de un retardo, donde será 1.

Procesos con incrementos independientes

Son procesos estocásticos a tiempo continuo en los que para cualesquiera de los tiempos $0 \leq t_1 < t_2 < \cdots < t_n$, las variables $X_{t_1}, X_{t_2} - X_{t_1}, \ldots, X_{t_n} - X_{t_{n-1}}$son independientes. Esto quiere decir que los desplazamientos que tiene el proceso en estos intervalos disjuntos de tiempo son independientes unos de otros.

Es posible ahondar en este tema tanto como se requiera: existen un sinnúmero más de tipos, a veces formados como híbridos de dos o más de los descritos anteriormente: por ejemplo, los procesos de Levy, que son una mezcla entre procesos de incremento estático e incremento independiente. Otros tipos de procesos surgen a partir de las funciones de distribución de las variables estudiadas, tal es el caso de los procesos Gaussianos.

Modelos auto-regresivos

Los procesos estocásticos son utilizados para analizar la influencia del pasado en el presente y futuro. Empero, los modelos auto-regresivos (AR) también son útiles para este tipo de análisis (de Arce, 2006). En ellos, se supone que el valor actual del proceso es una combinación lineal de valores pasados del mismo proceso, pero con un término de error e_t , que suponemos normalmente distribuido, que evita que el proceso sea determinista. Es decir,

$$X_t = \delta + a_1 X_{t-1} + a_2 X_{t-2} + \cdots + a_p X_{t-p} + e_t \qquad (1)$$

donde δ representa una constante.

Este proceso es conocido como AR(p), es decir, un modelo auto-regresivo de orden p.

Modelos auto-regresivos como procesos estacionarios

Si el término de error cumple las propiedades que pedimos a los procesos estocásticos de ruido blanco (media cero, varianza constante y covarianza cero

entre errores correspondientes a observaciones diferentes), ese término es llamado ruido blanco. Así, tomado la esperanza de ambos lados en (1), como $E(X_t) = \mu$, y $E(e_t) = 0$ tenemos que

$$\mu = \delta + a_1\mu + a_2\mu + \cdots + a_p\mu$$

y así,

$$\mu = \frac{\delta}{1 - a_1 - a_2 - \cdots - a_p}$$

de modo que, para que μ exista, se debe cumplir que $1 - a_1 - a_2 - \cdots - a_p \neq 0$.

Normalmente se trabaja con modelos AR de órdenes bajos (1 o 2), o con órdenes coincidentes con la periodicidad de los datos observados, por ejemplo, AR(4) para estudios trimestrales, o AR(12) para estudios mensuales.

No es difícil entender que los modelos auto-regresivos sean muy populares para hacer mediciones y predicciones, pues tienen un fuerte fundamento teórico en los procesos estocásticos, una rama muy importante de las matemáticas aplicadas.

Aplicación en el método ARIMA

Las siglas ARIMA significan *Auto-Regressive Integrated Moving Average*, es decir, modelo auto-regresivo integrado de media móvil. Este es un modelo estadístico que, mediante el estudio de modelos dinámicos de series temporales, permite encontrar patrones para hacer predicciones sobre el futuro. Estas predicciones se basan en los datos pasados, y no en una variable independiente.

Su popularidad radica en que, al usarlo, ya no es necesario que el investigador especifique el modelo a usar, sino que es el programa quien, al tener los datos temporales de la variable a estudiar, indica la estructura probabilística subyacente.

Las series temporales usadas en ARIMA son, claramente modelos auto-regresivos, en los que el ruido es blanco, de modo que utiliza procesos estocásticos estacionarios de ruido blanco aplicados a un modelo auto-regresivo.

Tecnologías

Java

Java apareció en el año de 1991, cuando un grupo de ingenieros de Sun Microsystems[19] decidió crear un lenguaje de programación enfocado hacia los electrodomésticos. Por ello, este debía ser muy sencillo y compacto, además de que tenía que ser ejecutado por una gran variedad de hardware con deferentes características y en constante cambio.

Debido a esto, los desarrolladores optaron por realizar código "neutro", que no dependiera de ningún electrodoméstico en particular, para luego ejecutarlo a través de una "máquina virtual", denominada *Java Virtual Machine* (JVM). La JVM era quien interpretaba el código, convirtiéndolo en uno particular para cada CPU utilizada. No obstante, y pese a los esfuerzos realizados, ningún fabricante de electrodomésticos se interesó por este nuevo lenguaje (García, y otros, 2000).

Fue a finales de 1995 cuando reapareció Java, siendo ahora enfocado hacia la programación de computadoras. Su adopción fue posible gracias a que se incorporó dentro del navegador de Netscape, produciendo una verdadera revolución en Internet. A principios de 1997, apareció Java 1.1, mejorando considerablemente a la primera versión, y a finales de 1998, nació Java 1.2, que más adelante se rebautizó como Java 2.

Hoy día, Java no solo es utilizado en el entorno Web, sino que se extiende hacia muchas aplicaciones y dispositivos utilizados a diario. Desde los teléfonos móviles hasta herramientas que proveen soluciones de comercio y de negocio electrónico,

[19] Empresa dedicada al desarrollo de software, absorbida por Oracle Corporation en 2009.

incluyendo algoritmos de big data, utilizados para el procesamiento de grandes cantidades de información.

Servlets

El término servlet se refiere a una tecnología de Java para la ampliación y mejora de servidores Web, proporcionando un método basado en componentes independiente de la plataforma, permitiendo una mayor flexibilidad comparada con la de otros programas de comunicación cliente-servidor que hacían uso del estándar de interfaz de entrada común (CGI)[20], teniendo así la libertad para elegir servidores, plataformas, y herramientas útiles para el desarrollo de aplicaciones (Oracle Corporation, 2016).

La tecnología fue desarrollada por Sun Microsystems en el año de 1996, buscando competir contra el estándar CGI, utilizado para la generación de contenido Web dinámico. El principal problema con este último era el hecho de que se tenía que crear un proceso dentro del servidor por cada petición realizada, lo que ocasionaba un desmesurado uso de memoria y una alta demanda de procesamiento. Como consecuencia, los desarrolladores se enfrentaban a una gran cantidad de problemas al desarrollar soluciones complejas o escalables que hacían uso de estándar.

En contraste, los servlets adoptaron una metodología distinta que les permitió hacer un uso más eficiente del servidor; se mantenían en memoria después de haber sido llamados la primera vez. Gracias a este mecanismo, los servidores fueron capaces de disminuir el tiempo de respuesta y optimizar sus recursos ante las peticiones por parte de los clientes (Kurniawan, 2015).

[20] Tecnología que define un estándar para la comunicación a través de la Web, permitiendo la comunicación cliente-servidor.

El mayor beneficio de los servlets es la conexión directa con la familia de las APIs[21] de Java, las cuales permiten interactuar con otras herramientas y que en conjunto son capaces de producir aplicaciones robustas, seguras y escalables. Esto se debe también a la alta compatibilidad que tienen con las características maduras del lenguaje Java, incluyendo la portabilidad, el rendimiento y la reutilización de código.

Estas características han hecho que los servlets sean hoy en día una opción popular para la creación de aplicaciones Web interactivas, siendo también considerados como una tecnología madura, teniendo a la fecha alrededor de nueve actualizaciones, de la que la versión 3.1 es la última disponible. La misma está definida en el *Java Specification Request* (JSR) 400.

El trabajo conjunto de esta tecnología Web del lenguaje Java con el código HTML[22] de uso común, trajo consigo la aparición de la tecnología JavaServer Pages (JSP), quien, en sus inicios, definía dentro de los mismos archivos de HTML el código de Java, proveyéndole así la lógica de la aplicación y la funcionalidad para comunicarse con otros módulos.

Sin embargo, se identificaron problemas inherentes a la combinación de los dos tipos de código, pues se generaban elementos de una difícil comprensión y complicada documentación, por lo que los desarrolladores optaron por separar el código Java del de HTML. Como resultado, las versiones posteriores de JSP hicieron explícita esta sugerencia e inclusive, a partir de la versión 2.0, se proporcionó una manera para desactivar el código de Java que se encontraba embebido (Kurniawan, 2015).

Hoy en día, la forma recomendada para acoplar este tipo de aplicaciones es utilizando la arquitectura de JSP *model 2*. Con ella, se escribe el código de Java dentro de sus propias clases (que, siguiendo con las buenas prácticas, pasarían a

[21] Interfaz de programación de aplicaciones; conjunto de herramientas disponibles para software de terceros, incluidas a manera de bibliotecas, ofreciendo así una capa de abstracción.
[22] *HyperText Markup Language*. Lenguaje de marcas para la elaboración de páginas web.

ser JavaBeans[23]) y se utilizan las páginas JSP para escribir tanto el contenido HTML como para presentar los objetos de Java al usuario final. Otras arquitecturas, tales como la del Modelo-Vista-Controlador, y *frameworks*[24] de desarrollo, como Java Server Faces o Spring MVC, se pueden conjuntar con *model 2*, para permitir una mayor flexibilidad y facilidad de uso al desarrollar aplicaciones.

Un servlet es un programa Java, al igual que una página JSP. En tiempo de ejecución, una página JSP se traduce y se compila dentro del servlet. Sin embargo, una aplicación desarrollada con servlets no puede ejecutarse por sí sola, por lo que hace uso de los denominados contenedores de servlets.

Los contenedores de servlets reciben las peticiones realizadas por un usuario a través de un navegador Web y se las pasan al servlet correspondiente. Posteriormente, regresan al mismo usuario las respuestas de dicho servlet. En la Figura 11 se presenta la interacción descrita.

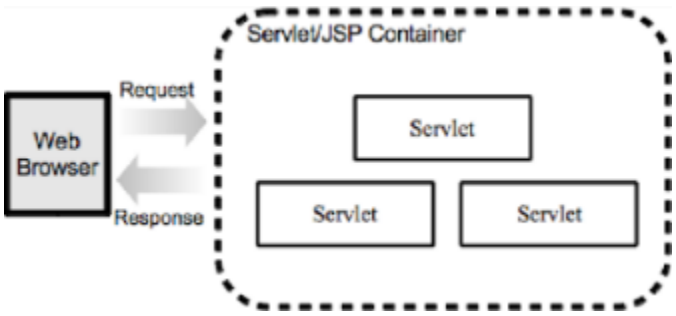

Figura 11. Interacción de una aplicación que utiliza servlets/JSP. Fuente: Kurniawan, B. (2015)

La Apache Software Foundation

La *Apache Software Foundation* (ASF) es una organización sin fines de lucro registrada en 2009, enfocada en dar soporte a proyectos Apache. Está conformada por una comunidad descentralizada de desarrolladores quienes dan mantenimiento a los diferentes proyectos de la Fundación de manera colaborativa. Dichos

[23] Modelo de componentes reutilizables de Java.
[24] Infraestructuras para el desarrollo de aplicaciones que definen un conjunto de prácticas y criterios para construir aplicaciones de forma ordenada, rápida, escalable, flexible y segura.

proyectos son distribuidos bajo la licencia Apache; una licencia abierta y pragmática, es decir, los proyectos Apache son caracterizados por ser software libre.

Para gestionar cada uno de los proyectos de la ASF, el equipo en cuestión elige a los desarrolladores encargados, quienes debieron haber contribuido previa y significativamente dentro de dicho proyecto. Así, los miembros de la *Apache Software Foundation* forman parte de una meritocracia (Arenas Delgado, y otros, 2016).

Uno de los objetivos principales de la ASF es proteger legalmente a los participantes de sus proyectos, así como prevenir que el nombre Apache sea utilizado por otras organizaciones indebidamente.

Historia de la ASF

La historia de la ASF se remonta al año 1995, en donde el servidor web más popular era uno desarrollado por el Centro Nacional de Aplicaciones de Supercomputación (NCSA), denominado NCSA HTTPd. No obstante, desde que el principal desarrollador Rob McCool dejó el proyecto en 1994, la evolución del software había quedado seriamente comprometida. A partir de entonces, el mantenimiento del mismo quedó en manos de los responsables de los sitios web, quienes realizaron algunas mejoras. Un grupo de ellos, utilizando el correo electrónico como medio principal de comunicación, se pusieron de acuerdo para conjuntar dichas mejoras y fueron dos de ellos; Cliff Skolnick y Brian Behlendorf, quienes iniciaron una lista de correo, un espacio para compartir información y un servidor en California para que los desarrolladores pudiesen trabajar. Al año siguiente, ocho de los programadores involucrados fundaron Grupo Apache (Arenas Delgado, y otros, 2016).

En abril de 1995, una vez conjuntadas las correcciones y mejoras al servidor NCSA HTTPd, se publicó lo que había de ser la primera versión oficial del servidor Apache (0.6.2). Curiosamente, fue por esas fechas que la NCSA decidió reemprender el desarrollo de su servidor.

En este momento, el desarrollo de Apache se bifurcó: por un lado, los desarrolladores siguieron trabajando en el Apache 0.6.2, llegando al 0.7; por el otro,

los programadores reescribieron por completo el código de la primera versión creando una arquitectura modular. Finalmente, en junio de 1995 se conjuntaron las dos ramas de Apache, haciéndose público como Apache 0.8.

A finales de 1995, específicamente el 1 de diciembre, apareció Apache 1.0, que incluía amplia documentación y mejoras modulares. Poco a poco, Apache sobrepasó al servidor NCSA como el más popular, puesto que ha conservado hasta la fecha (2016). En el año de 1999, los miembros de Grupo Apache fundaron la ASF con el objetivo de seguir brindando soporte al servidor HTTP de Apache y actualmente la Fundación tiene su sede en Los Ángeles (Arenas Delgado, y otros, 2016).

Apache Tomcat

Apache Tomcat es un contenedor web y servlet basado en Java, que se utiliza para alojar aplicaciones basadas del mismo lenguaje. Fue desarrollado por primera vez para Jakarta Tomcat[25], sin embargo, debido al aumento de la demanda, fue posteriormente alojado como un proyecto separado llamado *Apache Tomcat*, el cual es apoyado por la ASF.

Fue inicialmente desarrollado por James Duncan Davidson, un arquitecto de software de Sun Microsystems, quien posteriormente ayudó a que este fuera de tipo *open source*. Tomcat implementa las especificaciones Java Servlet y JavaServer Pages (JSP), y proporciona un entorno de servidor web HTTP "puro" permitiendo la ejecución de aplicaciones basadas plenamente en Java (Khare, 2012).

La última versión estable de esta herramienta a la fecha (febrero de 2016) es la 8.0 e implementa las especificaciones Servlet 3.1 y Java Server Pages 2.3. Incluye además un gran número de características adicionales que la convierten en una plataforma útil para desarrollar e implementar aplicaciones y servicios web.

[25] Proyecto que crea y mantiene software de código abierto para la plataforma Java.

Apache Tomcat se introdujo por primera vez en el grupo de código abierto en 1999 y su primera versión se lanzó con la versión 3.0.x. Desde entonces, ha sido apoyado por la comunidad de desarrolladores y ampliamente aceptado en la industria de las tecnologías de la información. Actualmente, Tomcat es ejecutado en entornos de producción, además de que ha sido utilizado para proyectos de misión crítica[26] dentro de diversas industrias (Khare, 2012).

Dentro de las mejoras progresivas que ha tenido esta herramienta, desde su versión inicial hasta la actual, se encuentran los descritos en la Tabla 4, destacando las características más representativas dentro cada una.

Tabla 4. Versiones de Apache Tomcat. Fuente: adaptado de Apache Software Foundation (2016)

Versión	Año	Características principales
3.x	1999	Versión inicial. Fusión del código Sun Java Web Server y las especificaciones ASF. Soporta las especificaciones de Servlet 2.2 y JSP 1.1.
4.x	2002	Soporta las especificaciones de Servlet 2.3 y JSP 1.2.
5.x	2004	Soporta las especificaciones de Servlet 2.4, JSP 2.0 y EL 1.1. Diseñado para J2SE 5.0. Se incluye Eclipse Java development tools (JDT) permitiendo que Tomcat funcione sin un Kit de desarrollo Java instalado previamente.
6.x	2007	Soporta las especificaciones de Servlet 2.5, JSP 2.1 y EL 2.1.
7.x	2011	Soporta las especificaciones de Servlet 3.0, JSP 2.2 y EL 2.2.
8.x	2014	Soporta las especificaciones de Servlet 3.1, JSP 2.3, EL 3.0, y WebSocket.

Al ser Apache Tomcat una implementación *open source* de la ASF pensada el manejo de Servlets de Java, extendiéndose además para soportar otras tecnologías tales como JavaServer Pages, *Java Expression Language*[27] y Java WebSocket, es preciso detallar también a su arquitectura interna. De entre los elementos que la conforman, se encuentran los descritos a continuación (Apache Software Foundation, 2016):

[26] Aplicaciones que, en caso de fallo, repercutirán operativa y económicamente a una empresa.
[27] Lenguaje de expresiones regulares utilizado dentro de las páginas web de Java.

- Servidor. En el mundo de Tomcat, un servidor representa al contenedor completo. Por defecto, la herramienta cuenta con una implementación predeterminada que a menudo no requiere configuración personalizada de los usuarios.
- Servicio. Es un componente intermedio que vive dentro del servidor y vincula a uno o más conectores directamente con un “motor”. Este elemento tampoco requiere una configuración adicional ya que por defecto funciona de una manera simple y eficiente.
- Motor. El motor representa una un canal de procesamiento de las solicitudes recibidas para un servicio en específico. Con un servicio puede tener diversos conectores, el motor recibe y procesa todas las solicitudes de los mismos, devolviendo la respuesta al conector adecuado para su transmisión hacia el cliente. Es poco común personalizar alguno de estos motores, pero Tomcat lo permite.
- Host. Se define como una asociación de un nombre de red (por ejemplo, www.company.com) con el servidor Tomcat. Un motor puede contener a varios hosts y cada uno de ellos también admiten alias de red como company.com y abc.company.com.
- Conector. Es quien gestiona las comunicaciones con el cliente. Existen varios conectores disponibles con Tomcat, incluyendo el HTTP, que se utiliza para canalizar la mayoría de tráfico de este protocolo, especialmente cuando se utiliza a la aplicación como un servidor autónomo. También se encuentra el conector Apache JServ Protocol (AJP), quien es el responsable de conectar al servidor Tomcat con un servidor web como Apache HTTPD. Es posible también crear conectores personalizados, como para administrar el tráfico hacia el protocolo seguro de transferencia de hipertexto (HTTPS).
- Contexto. Es quien representa a una aplicación web. Un host puede contener múltiples contextos, cada uno con una ruta única. La interfaz de contexto se puede implementar para contextos personalizados, sin embargo, Tomcat cuenta con el *StandardContext*, quien proporciona funcionalidades base significativas.

Apache Maven

Apache Maven es una herramienta para la gestión y construcción de proyectos Java. Fue desarrollada por la ASF y actualmente tiene la licencia Apache 2.0, convirtiéndose en una herramienta de software libre. El elemento fundamental utilizado por Maven es un archivo denominado *project object model* (POM), en donde se especifican todas las características que posee cualquier proyecto.

Maven, que en idioma yiddish significa "acumulador de conocimiento", fue originalmente desarrollado como un intento para simplificar los procesos de construcción dentro del *framework* Jakarta Turbine (software pensado para el desarrollo rápido de aplicaciones). Existieron diferentes alternativas realizadas anteriormente, utilizando como base el software Ant[28], cada una con sus propios archivos de construcción, pero que, al no existir un estándar para elaborarlas, terminaron por ser diferentes entre sí (Apache Software Foundation, 2016).

Ante esto, los desarrolladores optaron por establecer un método mediante el cual se pudiera especificar claramente al proyecto, es decir, la definición precisa de en qué consistía, así como un camino cómodo para publicar su información elemental. Además, se buscaba que este pudiera ser accesible desde otros proyectos (Apache Software Foundation, 2016).

El resultado de estos requerimientos fue una herramienta que hoy día (2016) es utilizada para la construcción y manejo de cualquier proyecto que se desarrolle con el lenguaje de programación Java. Maven se ha convertido en un *framework* elemental para los desarrolladores, ya que proporciona elementos de fácil manejo e integración dentro de diferentes proyectos. Acorde a lo establecido en su página oficial, el *framework* tiene, entre otras, las características descritas a continuación:

- Realiza la construcción de un proyecto de manera sencilla: a diferencia de otras herramientas (v. g. Ant), en Maven no se debe especificar el proceso

[28] Librería Java, desarrollada por la *Apache Software Foundation*, encargada del manejo de procesos, a través de las especificaciones descritas dentro de un archivo de construcción.

de construcción de un proyecto, ya que esta se basa en una estructura previamente definida, en la que no interviene el programador.

- Proporciona un sistema de construcción uniforme: en un proyecto Maven, la construcción de un proyecto se realiza a través de un archivo denominado *project object model* (POM). Además, se permite que las dependencias y complementos sean compartidas a través los diferentes proyectos.
- Aporta información de calidad sobre el proyecto: esto es en parte, gracias a la descripción proporcionada dentro del archivo POM, pero también debido al código generado automáticamente dentro de dicho proyecto. Por ejemplo, entre otras cosas Maven puede proporcionar:
    - El registro de cambios dentro del código.
    - Referencias cruzadas entre proyectos.
    - Listas de correo.
    - La lista de dependencias utilizadas.
    - Reportes de las pruebas unitarias realizadas.

  Todo esto es transparente para el usuario, lo que representa un menor esfuerzo para la construcción e implementación de un proyecto y, por ende, un desarrollo rápido de aplicaciones.
- Proporciona líneas guía para implementar mejores prácticas de programación: la estructura de Maven está definida acorde a los principios de las buenas prácticas para el desarrollo de software. Por ejemplo, la especificación, ejecución y reporte de pruebas unitarias forman parte del ciclo de construcción de un proyecto. Para ello, dichas buenas prácticas fueron utilizadas como base, manteniendo separado el código fuente de las mismas. Además, se utilizan convenciones de nombre, con el fin de localizarlas y ejecutarlas eficientemente.

  Maven también proporciona directrices sobre cómo diseñar la estructura del directorio del proyecto, permitiendo navegar de la misma forma dentro de diferentes proyectos. No obstante, esta se puede modificar si algún proyecto no se adapta a dicha estructura.

- Permite una migración transparente de nuevas características: es muy fácil realizar una actualización del software Maven, y con ello, obtener las características que se van sumando en cada nueva versión.

Spring

Spring es un framework de desarrollo montado en Java que proporciona soporte e infraestructura completo para el desarrollo de aplicaciones empresariales, pensada para que el usuario pueda enfocarse en el desarrollo de la aplicación sin preocuparse por la configuración inherente.

La herramienta permite la construcción de aplicaciones desde objetos simples de Java (POJOs, por sus siglas en inglés), implementando en ellos una serie de servicios empresariales no invasivos (SpringSource, 2016). La herramienta está pensada para utilizarse tanto en ediciones estándar de Java (JSE) como para empresariales (JEE).

Este framework surge como una alternativa ligera a la compleja plataforma de Java EE, siendo además de software libre y desarrollado por la Spring Source. Se puede utilizar en contenedores web como Apache Tomcat, Glashhfish y JBoss debido a la compatibilidad para el desarrollo de este tipo de aplicaciones, pero no se limita solo a ello; puede utilizarse también para aplicaciones de escritorio.

Spring parte de la teoría básica para la construcción de aplicaciones, en donde se elaboran diferentes objetos, pensados para trabajar de manera conjunta. De esta forma, estos se vuelven dependientes entre sí.

La solución fue pensada debido a que Java, aunque es capaz de proporcionar una gran cantidad de funcionalidades para el desarrollo de software, carece de los medios efectivos para organizar los diferentes bloques de construcción en un todo coherente, dejando esta tarea a los arquitectos y desarrolladores. Y si bien, existen diferentes patrones de diseño para abordar problemáticas comunes dentro de escenarios específicos, estos son simplemente “mejores prácticas” que se han

descrito, pero depende plenamente del programador el llevarlas hacia su implementación.

El framework, en un esfuerzo para solventar tales complicaciones, implementa los diferentes patrones de diseño como objetos que se pueden integrar fácilmente dentro de múltiples aplicaciones. Además, hace uso de un principio denominado "Inversión de Control" (IoC, por sus siglas en inglés) con el que provee de un medio para el desarrollo de componentes dispares dentro de una aplicación completamente funcional. Es por ello que numerosas organizaciones lo utilizan, buscando una solución para desarrollar aplicaciones robustas y sostenibles.

La herramienta cuenta con diferentes características organizadas en alrededor de 20 módulos (SpringSource, 2016). Estos están agrupados dentro los bloques siguientes: contenedor principal (*Core Container*), integración y acceso a datos, módulo web, programación orientada a aspectos (AOP, por sus siglas en inglés), instrumentación. Mensajería y pruebas. El diagrama de la Figura 12 ilustra lo anterior.

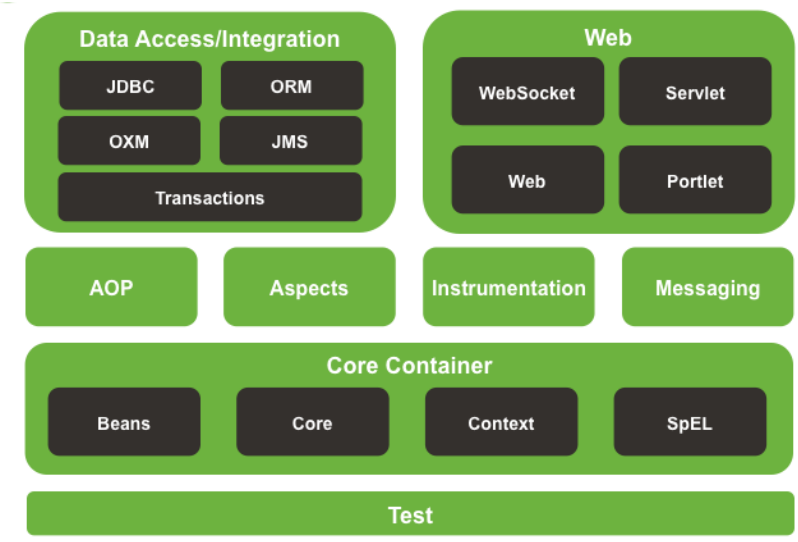

Figura 12. Módulos de Spring Framework. Fuente: SpringSource (2016)

Algunas de las funcionalidades de interés más utilizadas por los desarrolladores, incluidas en el diagrama de módulos presentado anteriormente, son:

- Spring MVC (Modelo-Vista-Controlador). Para el desarrollo de aplicaciones Web.
- Spring Security. Útil para brindar seguridad robusta en la aplicación desarrollada, incluyendo herramientas para la autenticación y autorización.
- Spring Data. Pensado para el consumo y actualización de datos, alojados tanto en bases de datos relacionales como en esquemas de tipo NoSQL y servicios en la nube.
- Spring Web Services. Para la creación de servicios web de tipo REST (*Representational State Transfer*) o SOAP (*Simple Object Access Protocol*).
- Spring Roo. Pensado como una herramienta ágil para el desarrollo de aplicaciones, pudiendo implementar soluciones sencillas en cuestión de minutos.

Modelo-Vista-Controlador

El Modelo-Vista-Controlador (MVC) es un patrón de arquitectura de software en el que se busca reducir el esfuerzo de programación de un sistema, estandarizando la construcción de los componentes, dividiéndolos en tres bloques elementales; el Modelo, las Vistas y los Controladores. Con esto, es posible implementar por separado cada uno de dichos elementos, permitiendo una mejor organización del trabajo (lo que conlleva a reducir los tiempos de desarrollo), además de un aumento en el código reutilizable, y una mayor especialización por parte de los programadores y los diseñadores involucrados en el desarrollo de un módulo específico.

Este patrón fue descrito por primera vez en 1979 (Fernández & Díaz, 2012), pensado para reducir el esfuerzo de programación requerido para el desarrollo de sistemas múltiples que hacían uso de los mismos datos. En esta especificación se detalla al Modelo, a las Vistas y a los Controladores como entidades separadas, con lo cual fue posible que, al realizar cualquier cambio dentro del Modelo, pudiera verse reflejado en todas las Vistas.

Con el MVC se tienen muchas ventajas, dentro de las cuales destacan las siguientes:

- Separación clara y concisa de los componentes del sistema, permitiendo su implementación por separado.
- Se puede abstraer dentro de cualquier Interfaz de Programación de Aplicaciones (API), por lo que cualquier usuario que haga uso de esta, podría implementar el MVC sin aparente dificultad.
- La conexión entre el Modelo y sus Vistas es dinámica; produciéndose en tiempo de ejecución, no en tiempo de compilación.

Gracias a esta arquitectura es posible construir cada uno de los bloques por separado, uniéndolos posteriormente. Además, si llegase a existir algún tipo de comportamiento indeseado dentro de un módulo, este podría reemplazarse sin que los demás elementos se vean afectados (Fernández & Díaz, 2012).

Para entender mejor cómo está constituido el MVC, se detallan sus partes a continuación y se ilustran en la Figura 13:

- Modelo. Se conforma por los datos que son pertenecientes al sistema. Es el responsable de manejar toda la información; los accesos y actualizaciones pertinentes. Es un ente que no tiene conocimiento específico de los Controladores y de las Vistas, es decir, no tiene ninguna referencia hacia estos.
- Vista. Dentro de esta se presentan los datos proporcionados por el Modelo en un formato adecuado para que el usuario interactúe con ellos. Para adquirir dichos datos la Vista se comunica preferentemente con el Controlador (quien entonces realiza una petición al Modelo), pero también es posible que tome la información directamente del Modelo a través de un enlace hacia este.
- Controlador. Es quien proporciona un significado de las órdenes del usuario, realizando peticiones al Modelo cuando es necesario. Además, es posible que el Controlador envíe actualizaciones de la información a sus Vistas

asociadas. Entonces, el Controlador puede verse como el intermediario entre la Vista y el Modelo, entrando en acción ya sea por cambios de la información dentro de este último, o bien, por alteraciones a la Vista realizadas por el usuario.

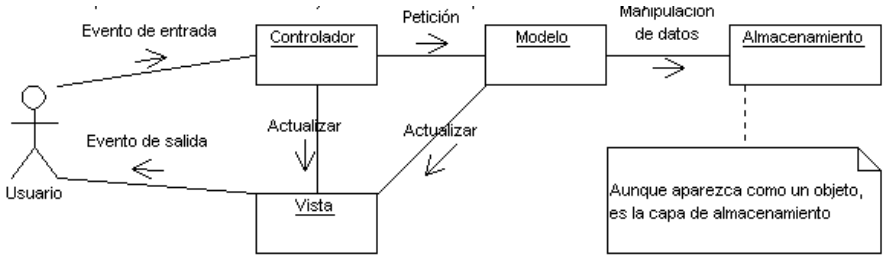

Figura 13. Interacción del modelo, la vista y el controlador. Fuente: Fernández & Díaz (2012)

Java Server Faces

Java Server Faces (JSF) es un framework tipo *front-end* para el desarrollo rápido de aplicaciones Web, pensado para una implementación veloz, principalmente de peticiones asíncronas entre el servidor y la página (peticiones AJAX[29]), sin tener que preocuparse por la codificación inherente y así orientarse hacia la funcionalidad principal.

JSF utiliza JavaServer Pages (JSP) como una tecnología para desplegar a las páginas web, pero también puede acoplarse con otros tipos de soluciones, tales como XUL (acrónimo de *XML-based User-interface language*; lenguaje basado en XML para la interfaz de usuario).

Entre las características que incluye JSF se encuentran las listadas a continuación:

- Cuenta con un conjunto de APIs para presentar componentes en una interfaz de usuario y administrarlos. Permite el manejo de eventos, validar lo que

[29] *Asynchronous JavaScript And XML*. Técnica de desarrollo Web que permite la creación de aplicaciones interactivas a través de peticiones y respuestas asíncronas entre el cliente y el servidor. De esta forma, se puede actualizar cierta parte del contenido de una página sin tener que recargarla.

ingrese dicho usuario, definir el esquema de navegación entre páginas y cuenta con soporte para internacionalizar sitios web.

- Posee bibliotecas de etiquetas que permiten incluir componentes tanto de terceros como personalizados.
- Permite manejar eventos del lado del servidor.

Una de las mayores ventajas de JSF es que el desarrollador solo tiene que aprender el modelo de programación para la construcción de la interfaz de usuario una vez, y luego aplicar este conocimiento a cualquier elemento compatible con JSF que pretenda utilizar. Lo anterior hace posible una estandarización del lenguaje, útil para el desarrollo veloz de soluciones futuras.

Entonces, JSF es una tecnología que trabaja con componentes, los cuales se conforman de atributos y una serie de comportamientos asociados. Debido a su estructura generalizada, es posible reutilizarlos o usar alguno proveniente de un tercero sin tener que aprender su estructura particular. Además, es posible utilizar varios para conformar a otro más complejos, permitiendo gran flexibilidad al momento de generar la aplicación web del usuario.

Gracias a esta conceptualización del modelo basado en componentes, el desarrollador se ve altamente beneficiado. Además, gracias a la separación estructural entre la lógica de la aplicación y las páginas web desarrolladas, unidas solo por el flujo de información existente entre una y otra, es posible generar un código limpio y de fácil mantenimiento (Dudney, Lehr, Willis, & Mattingly, 2004).

El modelo basado en componentes permite a los programadores generar características orientadas hacia la lógica del negocio, en lugar de centrarse en modelos de desarrollo que limitan la generación de características de valor. Además, garantiza que los componentes desarrollados serán reutilizables en el futuro.

JSF utiliza el patrón de diseño modelo-vista-controlador, permitiendo una segmentación en bloques de las diferentes secciones de código (Dudney, Lehr, Willis, & Mattingly, 2004). En este, el framework los cataloga como sigue:

- Modelo. Elemento que proporciona el acceso a los datos de los que hace uso la aplicación.
- Vista. Elemento que representa los datos de la aplicación y su comportamiento en una forma gráfica para la interacción con del usuario. Este la única interfaz con la que el usuario interactúa directamente.
- Controlador. Elemento que procesa los eventos dirigidos por el usuario, lo que puede resultar en actualizaciones del modelo o en la manipulación directa de la vista.

Además, de entre las ventajas de utilizar el framework JSF, en comparación con otros existentes, se encuentran las siguientes (Dudney, Lehr, Willis, & Mattingly, 2004):

- Normalización. JSF cuenta con un marco de desarrollo estándar basado en componentes para la web, lo que estandariza y hace más fácil su utilización por parte de los desarrolladores, comparado con otras herramientas que buscan cubrir la misma necesidad.
- Soporte de herramientas. Cuenta con un API bien documentada, además de incorporar elementos de *Java Enterprise Edition* que hacen más robusto al framework,
- Componentes. De todos los marcos existentes, muy pocos incorporan el concepto de componentes de interfaz de usuario reutilizables (más allá del concepto de una página Web). JSF es sólo un intento de este tipo, y está destinado a ser para las interfaces de usuario Web lo que Swing es para las interfaces de usuario más tradicionales.

R

R es un lenguaje de programación de alto nivel y un ambiente para análisis de datos. Liberado por primera vez en agosto de1993, desarrollado por Ross Ihaka y Robert Gentleman. Su diseño fue influenciado en gran medida por otros dos lenguajes; S, desarrollado en los Laboratorios Bell; y Scheme. El resultado fue un lenguaje muy similar en apariencia a S, pero la implementación subyacente y la semántica fueron derivadas de Scheme (Crawley, 2007).

Aunque tenga ciertas similitudes con sus antecesores, R cuenta con dos diferencias fundamentales (Ihaka, 1998), las cuales se especifican a continuación:

- Gestión de memoria. En R, se designa una cantidad fija de memoria para su arranque y es gestionado en tiempo de ejecución por un recolector de basura. Esto significa que la cantidad de memoria utilizada no crece de una manera súbita, sino que es controlada de una forma más estable, evitando problemas de paginación subsecuentes, representando una mejora sustancial en contraste con el leguaje S.
- *Scoping* o alcance de una variable. En S, las variables utilizadas en funciones pueden ser locales o globales. En cambio, dentro de R se permite que las funciones accedan a las variables presentes en memoria previas a la definición de la función. Un ejemplo de lo anterior se presenta en la Figura 14, en donde se aprecia que a una variable `total` se le asigna el valor de 10. Enseguida, se define la función `make.counter,` quien espera recibir un valor para instanciar a una variable llamada también `total`. A continuación, se define a una función dentro de `make.counter`, encargada de incrementar a la variable `total` propia de su *scope* (o alcance) y regresar el resultado obtenido.

  La ejecución del código anteriormente descrito se presenta en la Figura 15 y se explica a continuación: cuando se llama a la función `make.counter` para devolver un resultado (en este caso el resultado es su función interna), este se asigna a la variable `counter`, y en este momento se crea una nueva

instancia de la variable `total`. Posteriormente, al realizar las llamadas a la función establecida dentro de `counter`, esta opera con su propia variable `total` sin afectar a la definida previamente (ya que está fuera de su *scope*), lo cual se corrobora al mostrar en pantalla el valor actual de la primera variable declarada (`total`).

```
> total <- 10
> make.counter <- function(total = 0) {
+     function() {
+         total <<- total + 1
+         total
+     }
+ }
```

Figura 14. Código que ejemplifica el scope de una variable en R. Fuente: adaptado de Ihaka (1998)

```
> counter <- make.counter()
> counter()
[1] 1
> counter()
[1] 2
> counter()
[1] 3
> total
[1] 10
```

Figura 15. Salida del código mostrado en la Figura 14. Fuente: adaptado de Ihaka (1998)

En S, para tener el mismo comportamiento, la variable `total` debería declararse como global. En R, en cambio, se toma a la variable previamente definida dentro de su *scope*, teniendo como resultado una variable que solo puede ser vista y manipulada por la función definida dentro de `make.counter`.

En general, las reglas de *scoping* han definido a R como un lenguaje de programación con un estilo muy limpio, por lo que se han conservado pese a que se ha vuelto más compleja su implementación dentro de diferentes intérpretes.

Programación en paralelo

Los constantes aumentos de procesamiento computacional que se han estado desarrollando durante las últimas décadas han sido los detonantes de muchos de los avances en campos tan diversos como la ciencia, el Internet y el entretenimiento. Por ejemplo, la decodificación del genoma humano, la obtención de imágenes médicas cada vez más precisas, las búsquedas Web asombrosamente rápidas y precisas y los juegos de computadora cada vez más realistas habrían sido imposibles sin estos aumentos. De hecho, los aumentos más recientes en procesamiento computacional habrían sido difíciles, si no imposibles, sin los aumentos anteriores (Pacheco, 2011).

A medida que aumenta este poder computacional, el número de problemas que son considerados para ser resueltos aumentan de igual manera. Los siguientes son algunos ejemplos (Pacheco, 2011):

- Modelado de clima. Con el fin de comprender de una mejor manera el cambio climático, es preciso desarrollar modelos informáticos precisos que incluyan interacciones y análisis de los comportamientos entre la atmósfera, los océanos, la tierra sólida y las capas de hielo en los polos. Asimismo, es preciso hacer estudios detallados de cómo varias intervenciones humanas pueden afectar en demasía el clima global.
- Plegamiento de proteínas. Se cree que las proteínas mal plegadas pueden estar involucradas en enfermedades como Huntington, Parkinson y Alzheimer, pero la capacidad del ser humano para estudiar este tipo de configuraciones de moléculas complejas está severamente limitada por el poder computacional actual con el que se cuenta.
- Descubrimiento de medicamento. Hay muchas maneras en que el poder computacional aumentado se puede utilizar en la investigación en nuevos tratamientos médicos. Por ejemplo, hay muchos fármacos que son eficaces en el tratamiento de una fracción relativamente pequeña de los que sufren de alguna enfermedad. Es posible que se puedan idear tratamientos

alternativos mediante un cuidadoso análisis de los genomas de los individuos para quienes el tratamiento conocido es ineficaz. Esto, sin embargo, implicará el análisis computacional extenso de tales genomas.

- Investigación energética. El aumento de la potencia computacional permite programar modelos mucho más detallados de tecnologías como turbinas eólicas, células solares y baterías. Estos programas pueden proporcionar la información necesaria para construir fuentes de energía limpia mucho más eficientes.
- Análisis de datos. El ser humano genera enormes cantidades de datos. Según algunas estimaciones, la cantidad de datos almacenados en todo el mundo se duplica cada dos años, pero la gran mayoría de ellos es en gran medida inútil a menos que se analice. Como ejemplo, conocer la secuencia de nucleótidos en el ADN humano es, por sí mismo, de poco uso. Entender cómo esta secuencia afecta el desarrollo y cómo puede causar la enfermedad requiere un análisis extenso. Además de la genómica, las grandes cantidades de datos son generadas por los colisionadores de partículas, tal como el Gran Colisionador de Hadrones de la Organización Europea para la Investigación Nuclear, las imágenes médicas, la investigación astronómica y los motores de búsqueda en la Web, son ejemplos de estos datos.

Estos y una serie de problemas adicionales se han venido resolviendo conforme ha aumentado el poder de procesamiento computacional, a la vez de que se han ido creado nuevas estrategias y metodologías para abordarlos. Es aquí en donde se han implementado herramientas de big data en conjunto con las de programación en paralelo, buscando con ellas una solución ante tales complicaciones.

Para comenzar con ello, es preciso hablar de una herramienta computacional conocida como Hadoop, quien ha sido la tecnología de facto, utilizada para abordar tales retos.

La herramienta fue creada por Doug Cutting, creador de Apache Lucene; una interfaz de programación de aplicaciones (API, por sus siglas en inglés) utilizada para búsqueda y recuperación de texto. A su vez, Hadoop tuvo sus orígenes con Apache Nutch, que es un motor de búsqueda Web que utiliza como base precisamente a Lucerne (White, 2015).

Construir un motor de búsqueda Web desde cero era un objetivo ambicioso, pues no solo era programar el software necesario para realizar *crawling*[30] e indexación de sitios Web, sino también significaba el tener que ejecutar dicho software sin tener un equipo dedicado para ello, pues había de por medio muchas partes móviles. Además de esto, Doug Cutting, junto con el cofundador de Hadoop, Mike Caferella, estimaron que el hardware necesario para procesar a alrededor de mil millones de páginas tendría un costo cercano a los 500 mil dólares, con un coste de mantenimiento mensual de 30 mil dólares, por lo que el proyecto también era costoso. No obstante, tanto el creador como el cofundador lo consideraron como un objetivo digno, ya que daría paso a la democratización y estandarización de los algoritmos utilizados por los motores de búsqueda.

En 2002 se inició con la construcción de Nutch, pero poco tiempo después sus creadores se dieron cuenta de que la arquitectura del motor de búsqueda no podría escalar a los miles de millones de páginas Web estimadas. Sin embargo, al año siguiente Google publicó un artículo en el que se describe un sistema de archivos distribuido pensado para satisfacer las necesidades de almacenamiento este motor de búsqueda; Google File System (GFS). En general, este sistema buscaba reducir el tiempo utilizado para las tareas administrativas, tales como la gestión de los nodos en donde era necesario almacenar información. Para el año 2004, los desarrolladores de Nutch decidieron implementar un sistema de archivos distribuido de código abierto; el Nutch Distributed File System (NDFS) (White, 2015).

[30] Análisis y recolección de información dentro de la World Wide Web.

En este mismo año, Google dio a conocer otra publicación en el que se describe el modelo de programación MapReduce implementándolo dentro de GFS. Los desarrolladores de Nutch no tardaron en elaborar parte de estos algoritmos en su propio software, permitiendo que para el año 2005 se pudiera utilizar MapReduce en conjunto con NDFS.

Conforme se optimizaban los algoritmos dentro de Nutch fue posible emplear al NDFS más allá del ámbito de las búsquedas y para el año 2006 este sistema de archivos distribuido fue declarado como un subproyecto independiente llamado Hadoop. Al mismo tiempo, Doug Cutting se unió a Yahoo!, empresa que decidió proporcionar a un equipo dedicado y a los recursos necesarios para integrar al proyecto Hadoop dentro de su sistema. Esto fue demostrado en febrero de 2008 cuando Yahoo! anunció que su indexación de páginas web había sido generada por un clúster conformado por 10 mil procesadores (White, 2015).

En junio de 2008, se estableció a Hadoop como un proyecto de alto nivel de Apache, teniendo ya para entonces una activa y muy diversa comunidad. En esta época, Hadoop se comenzó a utilizar por muchas otras empresas además de Yahoo!, tales como Last.fm, Facebook y el New York Times.

Dentro de las acciones que se han conseguido gracias a Hadoop se encuentran: la digitalización de una serie de periódicos generando 4 terabytes de información en 24 horas, utilizando 100 computadoras en conjunto con la plataforma Amazon EC2; en abril de 2009 se anunció que el equipo de Yahoo! logró ordenar 1 terabyte de información en 62 segundos empleando un clúster de mil 460 nodos, superando lo reportado por la implementación de Google, quien requirió de 68 segundos para los mismos datos; y para el año de 2014, un equipo de la compañía Databricks[31],

[31] Empresa dedicada a ofrecer soluciones de Big Data.

utilizando 207 nodos de Spark[32] consiguió ordenar 100 terabytes de datos en 1,406 segundos a una taza de 4.27 terabytes por minuto (White, 2015).

[32]Framework de computación en clúster que forma parte del ecosistema Hadoop. Utiliza almacenamiento *in-memory* permitiendo un procesamiento de hasta 100 veces más rápido en comparación con el algoritmo de Hadoop tradicional, para aplicaciones específicas.

Metodología

Este trabajo se encuentra fundamentado en los conocimientos adquiridos dentro de la Facultad de Ingeniería, extrapolándolos hacia su utilización, culminando con la elaboración de una aplicación que demuestre los usos prácticos, tanto de la teoría matemática como de la computacional, aplicados a grandes cantidades de información.

Se parte de una investigación exploratoria con la que se busca conocer a profundidad los sistemas de software actuales, así como las buenas prácticas durante el desarrollo de algún proyecto, incluyendo tanto la documentación y organización teórica (que va de la mano con la administración de proyectos), como la práctica (técnicas sugeridas para la organización de código y utilización de *frameworks*), para así poder definir cuáles serán los utilizados durante el desarrollo. Además, se recolecta información sobre la utilidad de los procesos estocásticos, llevándolos hacia su utilización dentro de algoritmos que puedan proveer de información valiosa al usuario.

Posteriormente, se procede a la aplicación práctica de los conocimientos obtenidos, elaborando un módulo de software para realizar análisis a futuro de opiniones y tendencias grupales. Todo esto fundamentado matemáticamente y basándose en documentos históricos recopilados, tanto de las redes sociales como de diversas fuentes de noticias, por el software *Engine*.

Finalmente, se realiza la valoración de los datos obtenidos, verificando si se cumple o no con la hipótesis realizada. Se formula una conclusión basada en los resultados, identificando las posibilidades de crecimiento (su hubiere) y se generan las recomendaciones y propuestas subsecuentes.

El modelo de trabajo con el que se aborda la problemática sigue las prácticas recomendadas por la metodología Scrum, quien tiene los procesos descritos a continuación:

1. Inicio. En donde se genera la visión del proyecto y se priorizan los pendientes.
2. Planear y estimar. Se elabora un plan de trabajo con fechas y horas estimadas (v. g. diagrama de Gantt). Se identifican las historias de usuario para comenzar con la elaboración del proyecto.
3. Implementar. Se realiza la implementación del proyecto, verificando día a día los pendientes existentes buscando así minimizar un posible retraso. Se tienen reuniones ocasionales entre los principales interesados para mostrar los avances de dicho proyecto.
4. Revisión y retrospectiva. En donde se presentan y aprueban los entregables realizados, una vez terminado un *Sprint*.
5. Lanzamiento. Finalización del proyecto (que puede constar de uno o más *Sprints*). Se tiene un producto funcional listo para ser utilizado.

Con base en este esquema, el flujo de trabajo que se pretende seguir, y que es también parte del ciclo de vida de un proyecto en Scrum, se detalla puntualmente a continuación:

a. Problemática a abordar. Se parte de un caso o problema que se pretende resolver, siendo esta la verificación o refutación de la hipótesis planteada.
b. Visión del proyecto. A partir del caso, se vislumbra el valor que puede aportar cuando este se haya consumado. Particularmente, se pretende apoyar a los usuarios a realizar análisis de datos para generar un mayor valor, proveyendo predicciones de la información histórica recabada.
c. Priorizar los pendientes. Una vez identificado el o los objetivos del proyecto, se priorizan los más importantes a realizar, generando un mayor valor de la aplicación en el menor tiempo posible. Aquí, se generan las historias de usuario que se pretenden cubrir.
d. Planificar la ejecución. En donde se definen las tareas de desarrollo y documentación necesarias, así como los tiempos implícitos para su realización. Se genera entonces un diagrama de Gantt, conociendo los alcances y limitaciones del proyecto.

e. Generación de la lista de pendientes para el *sprint*. Aquí, se recolectan las tareas a realizar dentro de un *sprint*, cubriendo las historias de usuario descritas. Dado que la elaboración de este proyecto únicamente contará con uno, todos los pendientes para la realización de la tesis fueron incluidos aquí.
f. Creación del entregable. Periodo de ejecución del *sprint* (y en este caso, de todo el proyecto) realizando revisiones continuas, basadas en el avance programático estimado. Al término, se genera un entregable funcional que cubre las necesidades de las historias de usuario detectadas.
g. Entregables aceptados. Una vez que se haya validado por parte del principal interesado del proyecto que el entregable cubre todas las historias de usuario generadas, se aprueba y termina el *sprint* y, si es el último, se cierra el proyecto.

Capítulo 3. Desarrollo

Las enormes cantidades de información generada con los sistemas actuales y las características inherentes para su almacenamiento, análisis e interpretación, son el principal interés de este trabajo. Aprovechando diferentes herramientas para la recuperación de datos desde Internet, de limpieza de datos, almacenamiento, análisis, interpretación y presentación de resultados, aunado al sustento matemático, se busca construir un módulo adicional del software empresarial *Engine*, permitiendo con ello proporcionar una nueva forma de análisis de contenido, entregando información de valor innovadora para sus usuarios.

Como se ha mencionado anteriormente, el proyecto *Engine* perteneciente a la empresa utiliza los datos generados dentro de las redes sociales y de los diferentes medios de comunicación, para proveer estadísticas de valor a sus usuarios, como es el caso de índices de menciones, tendencias, opiniones y análisis de sentimiento de los diferentes actores analizados.

El objetivo principal de este trabajo consiste en el desarrollo de un módulo dentro de este software, para que, mediante el uso de un algoritmo de predicción, se pueda realizar un análisis cuantitativo de información, identificando con cierto grado de detalle, algunas de las tendencias de popularidad y opiniones futuras en torno a los actores de interés.

Selección de tecnologías

Para comenzar con el desarrollo, se definen las herramientas y métodos a ocupar, con base en lo obtenido a través de la investigación desarrollada dentro del marco teórico, buscando la mejor manera de abordar la problemática y encontrar una solución práctica y alcanzable. A continuación, se especifican detalladamente cuáles de estas tecnologías fueron elegidas, detallando el porqué de la selección para cada una.

Se decidió utilizar el lenguaje de programación Java en su versión 1.8, pues esta es la última versión estable y cuenta con amplia documentación y soporte. Se optó por este lenguaje debido a que, además de ser el utilizado mayoritariamente dentro del software *Engine*, se identificó como una tecnología madura y escalable, utilizada dentro del ámbito empresarial, teniendo inclusive una rama especializada en este sector; Java Enterprise Edition (Java EE). Esta cuenta con diversas especificaciones para la implementación y comunicación con Servicios Web, configuración XML, uso de anotaciones, herramientas para la conexión con bases de datos relacionales (*Java Database Connectivity*, JDBC), entre muchos otros. Además, dentro de Java EE está definida la especificación para el uso de Servlets y, dado que la interfaz de usuario de la herramienta *Engine* es de tipo Web, la utilización de estos hará posible una integración casi transparente con el módulo que se pretende desarrollar.

Algunas otras de las ventajas de Java son los *frameworks* con los que cuenta, muchos de ellos robustos, bien documentados y enfocados en el desarrollo ágil (v. g. Spring o Java server Faces), quitándole al desarrollador la imperiosa necesidad de configurar detalladamente el proyecto, para que se así pueda enfocarse en elaborar lo que verdaderamente da valor al usuario final. Asimismo, una gran cantidad de aplicaciones de terceros cuentan con mecanismos para interactuar con Java, ya sea a través de una API o de los denominados “kits de desarrollo de software” (SDK, por sus siglas en inglés), con los cuales se hace posible un acoplamiento inmediato entre estos lenguajes. Algunos ejemplos con los que se trabajará son el API de Facebook o el SDK de Twitter.

La arquitectura Modelo-Vista-Controlador para la solución a implementar estará implícita en todo momento, permitiendo así mantener de una forma ordenada y correctamente estructurada a los diferentes bloques de la misma. Los servlets de Java serán los responsables de implementar a cada uno de estos módulos. A continuación, se describen brevemente a cada uno:

- Modelo: fuentes de datos, tanto estructuradas como no estructuradas, en donde se almacenará la configuración del programa y la información a analizar.
- Vista: páginas web con las que podrá interactuar el usuario final, presentando los datos procesados de una forma atractiva y entendible, permitiendo además su manipulación para filtrar algún contenido específico buscado.
- Controlador: lógica de la aplicación, interactúa tanto con el modelo, para recuperar y almacenar los datos necesarios, como con la vista, para presentar dichos datos acorde a las consultas realizadas por parte del usuario final.

Para la construcción y gestión de los servlets, se pretende utilizar un conjunto de herramientas que permiten la cooperación conjunta entre sí, estos son: Spring MVC y Java Server Faces (JSF). El primero, es un módulo perteneciente a la paquetería de Spring, quien sugiere una arquitectura de programación en capas, implementa mecanismos de seguridad robusta, es fácilmente configurable y escalable, y quien será el encargado de operar toda la lógica de la aplicación; conectarse las fuentes de datos y con otros servicios para recuperar y procesar la información, manipulará los controladores de los servlets, entre otros.

El segundo, es un *framework* que se utilizará en la parte visual de la aplicación, comúnmente denominada *front-end*. La ventaja de utilizar a este último, es que facilita en gran medida la codificación de peticiones asíncronas entre el cliente y el servidor, utilizando *Asynchronous JavaScript And XML* (AJAX)[33]. Además, cuenta con una serie de extensiones que lo hacen más robusto, como es el caso de PrimeFaces, una librería de componentes enriquecidos para la construcción rápida

[33] Técnica de desarrollo Web que permite la creación de aplicaciones interactivas a través de peticiones y respuestas asíncronas entre el cliente y el servidor. De esta forma, se puede actualizar cierta parte del contenido de una página sin tener que recargarla.

y eficiente de aplicaciones Web, e inclusive, las últimas versiones permiten el acoplamiento con otros frameworks de HTML, tales como Bootstrap[34].

Para configurar y persistir la información recolectada y analizada, se pretende utilizar una base de datos relacional: MySQL, y un motor de búsqueda que permita el almacenamiento de información: Solr, respectivamente. Esto debido a que la primera se acopla a los datos de configuración iniciales de la aplicación y la segunda es óptima durante la búsqueda de contenido dentro de grandes cantidades de datos, debido a la naturaleza *full-text*[35] de la misma.

La definición y carga dentro del proyecto de los componentes antes mencionados, además de otros inherentes al desarrollo, se realizará mediante la herramienta Maven, perteneciente a la ASF. Este brinda algunas buenas prácticas para la organización de los archivos del proyecto y, sobre todo, permite llevar un manejo concreto sobre las versiones utilizadas en el mismo. Así, el seguimiento de las mismas para mantener actualizado al proyecto se convierte en una tarea sencilla para los desarrolladores futuros.

Para la ejecución de servlets del lado del servidor se pretende utilizar Apache Tomcat, pues a pesar de ser de código abierto, cuenta con una amplia documentación y soporte por parte de los desarrolladores del proyecto. También, es uno de los más utilizados dentro del mercado, fácil de configurar y ejecutar, siendo además multiplataforma.

En cuanto al tratamiento matemático de los datos, se realizará mediante el lenguaje de programación R, quien tiene a su disposición una amplia gama de librerías que proveen funcionalidades matemáticas avanzadas, adaptadas para ser utilizadas de una manera fácil y rápida. Asimismo, cuenta con algunas extensiones para

[34] Framework *front-end* que brinda una serie de estilos y extensiones predefinidos para el diseño visual de una página Web.
[35] Técnicas para el guardado y búsqueda de información dentro de una base de datos basada en texto.

comunicarse a través de sockets[36], permitiendo la comunicación entre los diferentes módulos del sistema. Particularmente, se eligió la extensión Rserve, pues su implementación tanto en R como en Java es relativamente sencilla.

Finalmente, el modelo matemático implementado para la predicción de datos dentro de R será Arima, que es catalogado como un modelo auto-regresivo. No obstante, este hace uso de procesos estocásticos de ruido blanco para su funcionamiento.

Proceso general de la aplicación

Las características técnicas relacionadas con la aplicación a desarrollar, incluyendo parte de la funcionalidad existente del software *Engine*, se describen de manera general a continuación. Más adelante, se profundiza sobre la analítica de datos, sección en donde se define el funcionamiento e interacción que irá teniendo el módulo implementado.

Primeramente, se define la arquitectura general del software, visualizando así la interacción que tienen entre sí los diferentes módulos, incluyendo el que se pretende desarrollar. A continuación, se detalla sobre cómo se realiza este trabajo conjunto entre aplicaciones, mismo que es presentado en el diagrama de bloques de la Figura 16.

[36] Técnica mediante la cual dos computadoras conectadas a través de una red pueden intercambiar mensajes de forma fiable y ordenada.

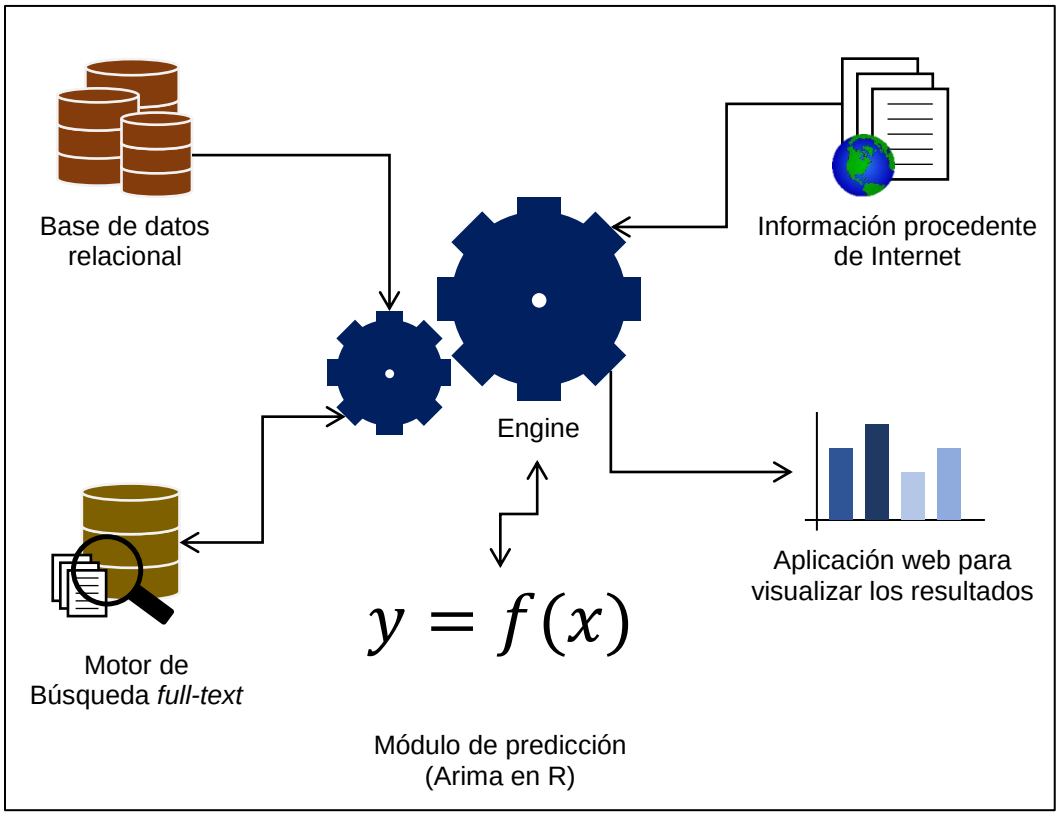

Figura 16. Arquitectura de la aplicación. Fuente: elaboración propia

Primeramente, la aplicación *Engine*, navega por Internet buscando información relevante, tanto de las redes sociales como de algunos medios de comunicación impresa. Los datos se recuperan, se obtienen sus metadatos (fuente, idioma, análisis sentimental, entre otros) y se guardan dentro de un motor de búsqueda, estando disponibles para realizar análisis y consultas posteriores. Enseguida, los usuarios finales, a través de un navegador web, consumen la información que ya ha sido guardada, realizando búsquedas y aplicando filtros. La herramienta puede presentar entonces gráficas, diagramas o análisis predictivos de una manera entendible y de fácil comprensión, aportando así una interpretación concisa y valiosa de aquellos datos de interés con base en los que fueron obtenidos desde internet. Todas las configuraciones internas para realizar esta interacción se encuentran almacenadas dentro de una base de datos relacional, por lo que las consultas hacia esta son realizadas durante todo el flujo del programa.

En cuanto al proceso de ejecución involucrado, que es presentado en la Figura 17, comienza con la administración de datos, en donde se adquieren, extraen, limpian y persisten los necesarios a analizarse. Posteriormente, se recupera información de valor de los mismos a través de técnicas de modelado y análisis, representando los resultados dentro del software de una manera entendible para el usuario. Enseguida, este puede aplicar filtros y búsquedas particulares, incluyendo la ejecución del método predictivo, útil para estimar el comportamiento futuro de la información que se esté analizando. Finalmente, el analista interpreta los resultados que le arroja el software, basados en la información histórica previamente recopilada.

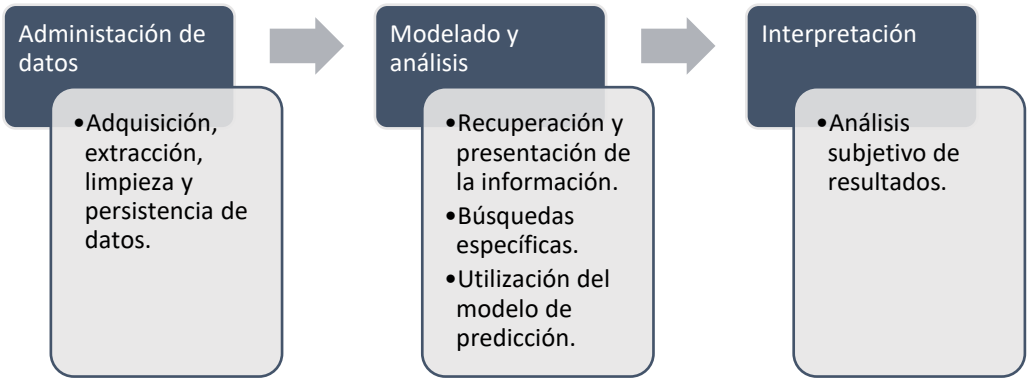

Figura 17. Proceso de ejecución de la aplicación. Fuente: elaboración propia

A continuación, se detallan las diferentes fases del proceso, abordando con particular interés a la del modelado y análisis de datos, a la vez que se profundiza sobre el desarrollo del módulo de análisis predictivo.

Administración de datos

Dentro de esta primera fase se define de manera general el proceso involucrado para la adquisición, clasificación y guardado de la información, misma que será utilizada por el módulo a desarrollar y que ya es recuperada por el software *Engine*.

Para comenzar con este proceso, se definen las fuentes de información desde donde se obtienen los datos, incluyendo páginas de noticias y redes sociales. En estas últimas se encuentran las aplicaciones de Facebook y Twitter.

Una vez identificadas, se procede a la extracción de datos desde Internet, utilizando diferentes métodos de recuperación de contenido. Para las redes sociales se utilizan las APIs respectivas, mismas que posibilitan la adquisición de datos de una manera sencilla y rápida. Para los sitios web de los diferentes medios de comunicación, en cambio, se tienen que utilizar algoritmos de introspección mediante arañas web[37], posibilitando así la obtención de los datos relevantes y una primera clasificación de contenido; se identifican enlaces, contenido multimedia y texto plano.

Ya que se ha realizado la adquisición de información, tanto de las redes sociales como de los medios de comunicación, se procede a realizar la limpieza y extracción de metadatos. Para la primera etapa, se comparan los datos que fueron obtenidos en el proceso previo con la información almacenada dentro de la base de datos de configuración, descartando así a aquellos que no tengan ningún tipo de contenido relevante. Posteriormente, se aplican algoritmos de inteligencia artificial para identificar elementos puntuales que no necesariamente forman parte del contenido principal, pero que ayudan, entre otras cosas, a establecer relaciones entre otros datos durante la fase de análisis de la información. A continuación, se presenta con un ejemplo de algunos de los metadatos que se pueden identificar del tweet presentado en la Figura 18.

A partir del tweet citado y con ayuda del API de Twitter, se pueden recuperar los atributos siguientes:

- Fecha y hora de publicación del tweet: 3 de marzo de 2016, 12:59 horas.
- Usuarios etiquetados: Eduardo Vázquez M. (VazquezMartin), Arquine (Arquine), MuseoCiudadMéxico (MuseoCDMX), AEP CDMX (aepCDMX), Desarrollo Económico (SedecoCDMX), Miquel Adrià (miqadria), El Colegio Nacional (ColegioNal_mx), FMPT CDMX (fmpt_cdmx)
- Actores involucrados: Teodoro González y Museo de la Ciudad de México.
- Menciones: @MuseoCDMX.

[37] Programa que permite la exploración de páginas Web de forma metódica y automatizada.

- Me gusta: 22.
- *Retweets*: 25.

Figura 18. Tweet publicado por la Secretaría de Cultura de la Ciudad de México. Fuente: Adaptado de Twitter (2016)

Cabe señalar también que, dentro de esta etapa, se utiliza un clasificador que identifica el tipo de sentimiento expresado en cada documento, estableciendo la tendencia que tiene, pudiendo ser positiva, neutra o negativa.

Una vez terminado el proceso de extracción y limpieza, se procede a realizar la persistencia de resultados dentro de alguna herramienta que permita el guardado de datos, así como la flexibilidad para realizar búsquedas y consultas ágiles, dentro de estas grandes cantidades de información. Es aquí en donde se utiliza un motor de búsqueda completamente basada en texto (*full-text*), debido a la forma en que se presentan los datos recolectados y a que esta herramienta se encuentra optimizada para indexar y realizar consultas de este tipo de contenido con un tiempo de respuesta rápido, comparado con otros tipos de soluciones.

Con este último paso, se concluye la primera fase del proceso general, teniendo como salida la información limpia, clasificada y almacenada, y que posteriormente será utilizada para entregar información de valor al usuario final. Este podrá entonces, a través de una aplicación web, interactuar con los datos y encontrar elementos de particular interés, teniendo además la posibilidad de utilizar el algoritmo de predicción para estimar el comportamiento que tendrán estos datos

con base en la información histórica recopilada. Dicho módulo se describirá detalladamente en los siguientes apartados.

Modelado y análisis

El comienzo de esta fase se da una vez que el usuario requiere obtener información de valor sobre los datos históricos recopilados previamente. Por ejemplo, en caso de ser una elección presidencial, el software *Engine* podría devolver indicadores sobre las tendencias que tendría la población sobre cada candidato e inclusive devolver el tipo de comentarios que se realizan al respecto (análisis sentimental). Por ello, lo primero que se tendría que hacer es navegar por las diferentes pantallas de la aplicación y elegir los filtros pertinentes para encontrar e identificar aquello que se esté buscando.

La herramienta cuenta con diferentes parámetros de búsqueda y métodos de filtrado, mismos que pueden utilizarse para especificar a las entidades e instituciones que se pretenda analizar. Asimismo, permite limitar el intervalo de fechas a uno de particular interés. El módulo de análisis predictivo desarrollado se adapta también a este tipo de criterios, permitiendo un análisis integral de la información.

Dentro de los diferentes módulos con los que cuenta la aplicación, son de particular interés los siguientes:

- Social Tracking. Aquí, se identifican los comentarios relacionados con algún actor o un grupo de actores dentro de las redes sociales y los medios de comunicación. A partir de estos, se presenta el tipo de sentimiento de cada uno y, recopilando todos los que se tienen, se obtiene y presenta un análisis de sentimiento generalizado por cada actor en un intervalo de tiempo.
- Menciones. En esta sección se visualiza el número de menciones dentro de un periodo establecido, pudiendo ver el detalle de cada una de las fechas y así identificar, sobre todo dentro de los puntos atípicos, qué sucesos desencadenaron dicho comportamiento.

Cabe señalar que, aunque el software *Engine* cuenta con alrededor de ocho módulos adicionales, el alcance de esta tesis abarca solo los dos descritos anteriormente. Sin embargo, se pretende que la aplicación desarrollada sea modular y pueda extenderse su funcionamiento a aquellas secciones en donde aplique.

Se eligieron los módulos de menciones y social tracking debido a que en ambos se dibuja la popularidad de un actor para un periodo de tiempo y, con base en este histórico, se puede estimar el comportamiento futuro que tendrá, despreciando los puntos atípicos que podrían surgir por un evento imprevisto. A continuación, se desglosa la implementación del modelo matemático elegido, la arquitectura particular del módulo y su funcionalidad.

Implementación del modelo

Como se abordó al comienzo del capítulo, se pretende utilizar el proceso auto-regresivo ARIMA para realizar la predicción de datos. Este, no está catalogado como un proceso estocástico, empero, utiliza uno fundamental para su funcionamiento; el proceso estacionario de ruido blanco.

Una vez que el analista ha accedido a la sección de Social Tracking o Menciones dentro del software *Engine*, este especifica los parámetros de búsqueda necesarios para recuperar la información de interés. Ya que lo ha hecho, se le presentan las gráficas de sentimiento o de menciones, respectivamente, en donde se puede visualizar el comportamiento general que han tenido los usuarios de redes sociales y la tendencia de comentarios que se han realizado en los diferentes medios de comunicación. El siguiente paso lógico dentro de la aplicación es realizar el análisis predictivo, mandando, con ayuda del módulo desarrollado, esta misma información que ha sido filtrada a un programa en R. Este último aplica entonces el modelo ARIMA, obteniendo los resultados esperados a futuro. Una vez realizado el análisis, R devuelve la información obtenida a la aplicación, quien dibujará la nueva información dentro de la gráfica que se le presentaba al usuario. Una vez hecho

esto, el proceso de análisis predictivo se habrá ejecutado exitosamente. El flujo de datos descrito anteriormente se ilustra en la Figura 19.

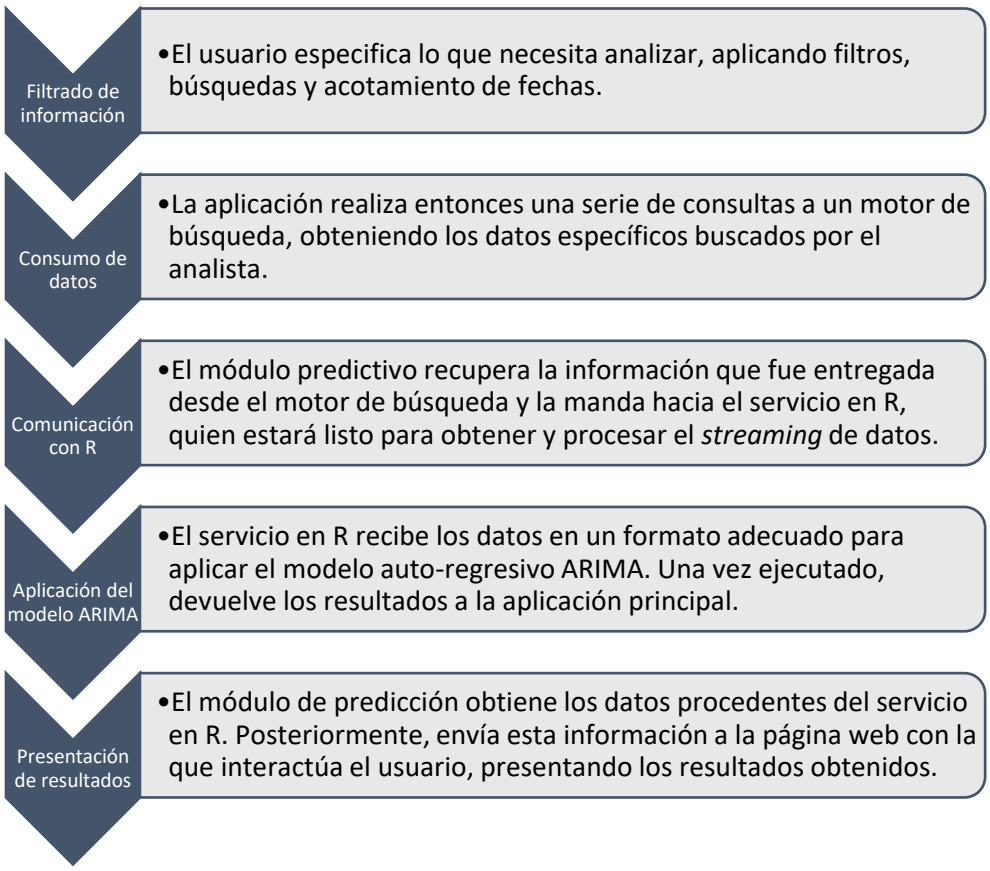

Figura 19. Flujo de información para realizar un análisis predictivo. Fuente: elaboración propia

Arquitectura

La arquitectura inherente, perteneciente a la comunicación entre la aplicación principal, el módulo desarrollado, los datos históricos necesarios y el procesamiento matemático ejecutado en R se presentan en la Figura 20.

Como se aprecia, el flujo de la aplicación, desde que el usuario comienza a utilizar el programa hasta que obtiene los resultados del análisis predictivo, implica que los diferentes módulos interactúen entre sí. Esta comunicación interna se detalla cronológicamente con los números del 1 al 6, representando cada uno lo siguiente:

1. El usuario aplica las búsquedas, filtros y acotaciones pertinentes dentro de la aplicación, obteniendo así la información precisa sobre lo que necesite analizar.
2. La herramienta *Engine* recupera entonces desde el motor de búsqueda los datos pertinentes y de valor para el usuario.
3. Se manda el *streaming* de información hacia el programa en R para realizar la predicción a través del modelo ARIMA. Para recibirlos, R opera como un servicio.
4. R recibe la cadena de información y la procesa, aplicando el modelo auto-regresivo ARIMA a los datos recibidos y obteniendo como resultado un conjunto de valores que representan los datos que se esperan para un periodo de tiempo a futuro.
5. El servicio levantado en R envía de regreso al programa principal la nueva información, que consiste en los datos que fueron predichos.
6. Finalmente, a través de la aplicación web, se presentan los datos adicionales al usuario.

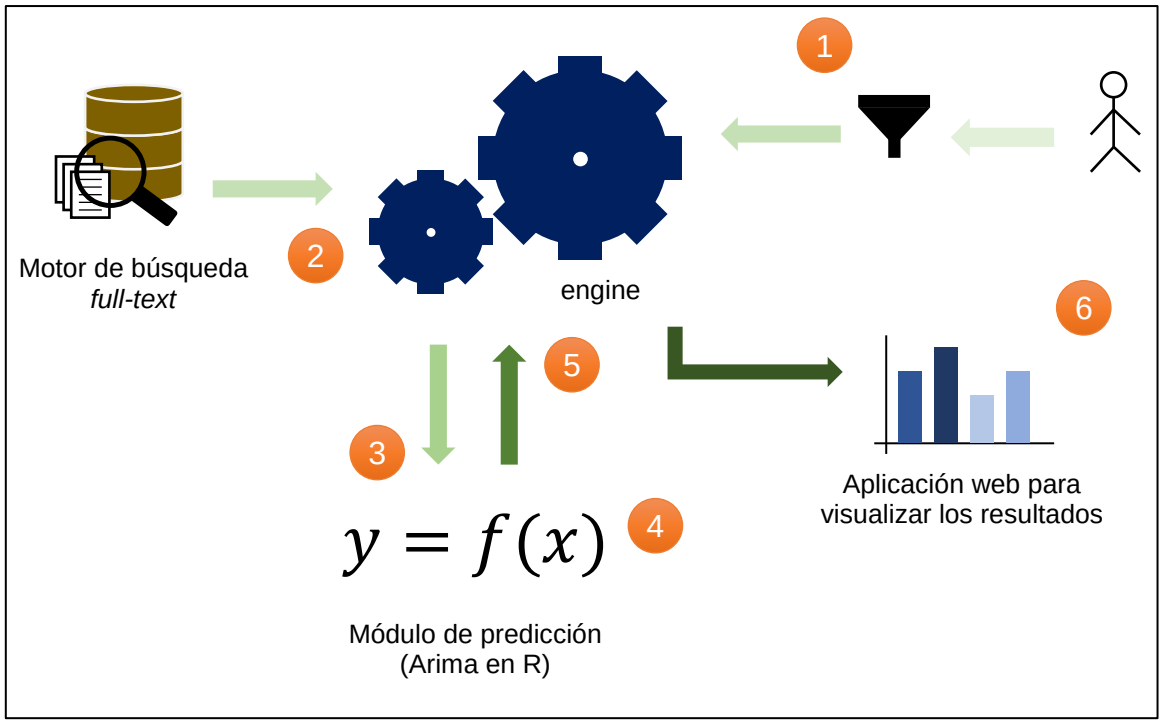

Figura 20. Arquitectura particular del módulo de predicción desarrollado. Fuente: elaboración propia

Un análisis minucioso sobre los diferentes niveles en los que se trabaja dentro del software se detalla a continuación, tomando en cuenta los frameworks utilizados para su construcción.

La integración de la arquitectura con el modelo-vista-controlador, quien hace una distinción entre la capa de datos, la lógica de la aplicación y las diferentes vistas con la que interactúa el usuario, se ilustra en la Figura 21.

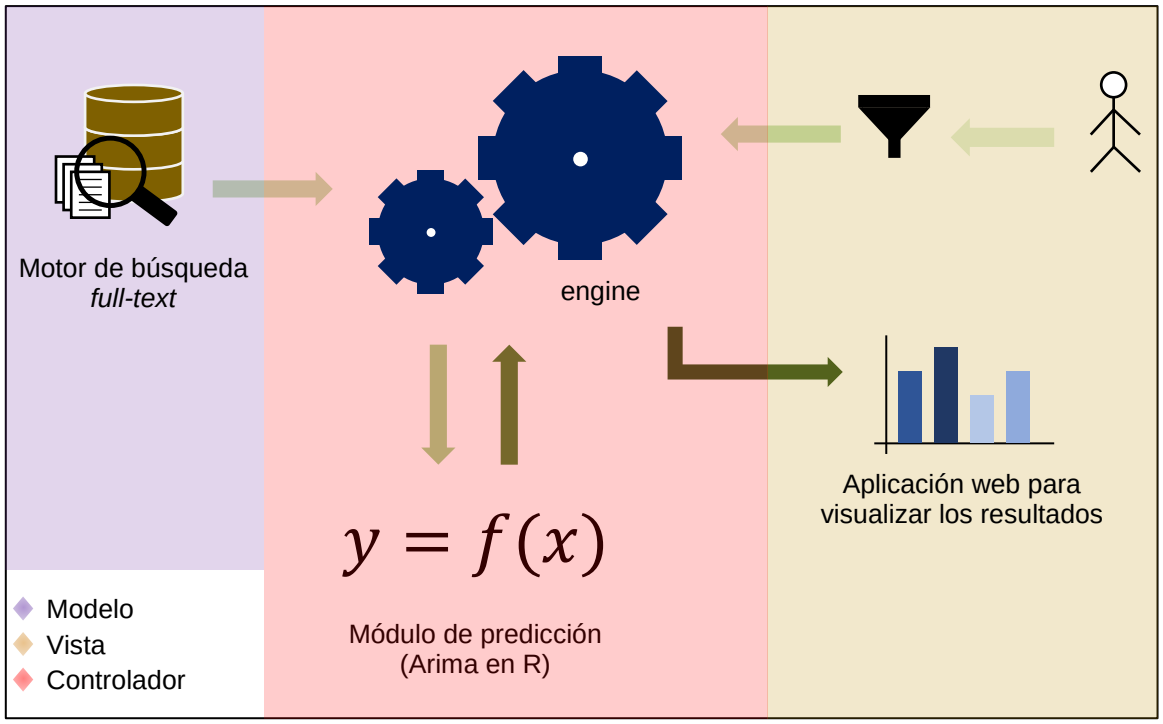

Figura 21. MVC dentro de la aplicación. Fuente: elaboración propia

Esta separación es optimizada a través de las herramientas de Java Server Faces y Spring, apoyándose también con otras como las utilizadas para el *front-end* (siendo estas Bootstrap y Primefaces), el API y SDK para la comunicación con el motor de búsqueda y la implementación y conexión con el servidor en R utilizando la extensión Rserve.

Para la parte del modelo, se hace uso del motor de búsqueda Solr, quien cuenta con un API y algunas librerías que conforman a su SDK para la interacción directa entre Java y la base de datos. Con estas, es posible realizar el guardado y consumo de datos desde la aplicación.

La integración de la librería *Solrj*, quien permite este tipo de comunicación, se realiza a través de Maven, quien posibilita su inclusión prácticamente transparente e inmediata dentro del proyecto.

En cuestiones de la vista, el software se apoya plenamente en la construcción de páginas web utilizando etiquetas de JSF, gracias a las cuales es posible interactuar directamente con la lógica de la aplicación. Además, para una visualización que permita una interacción intuitiva, se implementa la misma con el *framework* Bootstrap, quien está pensado para brindar al programador una forma fácil y rápida de generar páginas web responsivas[38] y de fácil utilización para el usuario final. Asimismo, para la representación de las gráficas que se utilizan para dar una mejor visibilidad de los datos al analista, se le presentan utilizando herramientas de JavaScript[39], en particular, una librería denominada Highcharts.

Finalmente, para el controlador que funge como el núcleo de la aplicación, se utilizan servlets, pues dicho software se encuentra plenamente basado en el lenguaje Java. Los servlets implicados para la realización del módulo son dos; el primero, gestionado por Java Server Faces, es empleado para realizar la comunicación entre la vista y el programa, con la bondad de que cuenta con abreviaturas de código HTML para agilizan su implementación. Por su parte, el segundo servlet es manejado por Spring MVC, quien posee una serie de características adicionales, útiles para mantener aplicaciones robustas al permitir por ejemplo la inyección de dependencias, la reutilización de código y diferentes mecanismos de seguridad adicional.

El *framework* de Spring es utilizado también debido a que este puede interactuar de una manera transparente con la base de datos relacional, utilizada para la configuración de la aplicación. Esto a través del “API de persistencia de Java” (JPA, por sus siglas en inglés), que es una especificación descrita dentro de Java EE para

---

[38] Adaptables al tamaño de la ventana del navegador.
[39] Lenguaje de programación utilizado principalmente del lado del cliente, para la construcción de páginas web dinámicas.

la comunicación entre el lenguaje Java y una base de datos transaccional, sin importar cuál fuese. Gracias a Spring, la configuración de la conexión y la interacción futura entre estos módulos se simplifica sustancialmente.

La inyección de dependencias, que es una característica importante de la tecnología, representa otro punto a favor para utilizarla. Con ella, se posibilita la comunicación interna entre los diferentes módulos del software *Engine*, pudiendo así “inyectar dependencias” entre las clases programadas. Es así como el módulo de predicción desarrollado puede recibir los filtros y acotaciones que el usuario final introdujo.

Después, haciendo uso de la librería Solrj que fue agregada a través de Maven, el módulo predictivo (quien ya cuenta con todos los parámetros necesarios para consumir información) realiza una consulta al motor de búsqueda. Este, recupera todos los registros que le fueron solicitados y los regresa en un formato adecuado para que, una vez más, el módulo predictivo pueda recibirlos y enviarlos al servicio en R.

Las consultas que pueden realizarse dentro del motor de búsqueda tienen cierto contraste con las que se ejecutan dentro de una base de datos relacional. Con ellas, es posible realizar desde búsquedas simples, como que un campo cumpla ciertos parámetros (igual a falso, mayor a 0), hasta algunas más especializadas, por ejemplo, la búsqueda de una cadena en particular dentro de todos los campos en cualquier documento guardado, las cadenas más parecidas a alguna introducida e inclusive una búsqueda espacial para ubicar puntos dentro de algún polígono.

Si bien, este tipo de consultas pueden ejecutarse también dentro de una base de datos relacional, es mucho más fácil, práctico y eficiente realizarlas en una herramienta *full-text*, como es el caso de Solr.

En cuanto al servicio (denominado así en un sistema operativo Windows) o demonio (nomenclatura utilizada para los sistemas UNIX) de R, pensado para la comunicación desde y hacia el módulo de análisis, este fue implementado con

ayuda de la extensión Rserve, quien cuenta con un SDK para Java, permitiéndole identificarse como cliente y así poder acceder a toda la suite de herramientas con las que cuenta el lenguaje.

Para su instalación, es necesario agregar el paquete Rserve mediante la instrucción:

```
install.packages("Rserve")
```

para posteriormente realizar la configuración de red correspondiente y finalmente levantar el servicio. En dicha configuración es preciso señalar el número de puerto por el que se comunicará, el espacio de trabajo a utilizar y el tipo de codificación utilizada para los datos transmitidos.

Después de haberse levantado, el demonio estará listo para recibir peticiones, ejecutarlas y devolver respuestas. Lo siguiente consiste en posibilitar la comunicación entre Java y dicho servicio, utilizando la dependencia correspondiente e incluyéndola, igualmente, mediante Maven.

Con las herramientas que provee esta dependencia, es posible configurar una conexión dedicada, mediante la instrucción:

```
RConnection c = new RConnection ([IP_DEL_SERVICIO]);
```

para posteriormente trabajar directamente con instrucciones del lenguaje R. Por ejemplo, ejecutando la siguiente línea de código:

```
Double d [] = c.eval ( "rnorm (10)"). AsDoubles ();
```

Java mandará al servicio de R la instrucción:

```
rnorm (10)
```

y así obtener dentro de la variable `d` 10 muestras aleatorias de la distribución $N(0,1)$.

Predicción

Para la puesta en marcha del modelo auto-regresivo ARIMA, Java (utilizando el servicio en R) llama a una función previamente programada, pasándole también el streaming de datos a analizar. Con ello, R obtiene información que representa un comportamiento estimado a futuro, basándose en los datos históricos recopilados. La sección de código en R que realiza este análisis hace uso de otro paquete denominado "*forecast*".

El paquete *forecast*, como su nombre lo indica, hace uso de herramientas matemáticas para realizar diferentes tipos de análisis predictivo, tomando como entrada series de tiempo y modelos lineales. Cuenta con métodos y herramientas que permiten mostrar y analizar el comportamiento de los datos en un intervalo de tiempo, incluyendo alisado exponencial[40] y ARIMA automático.

Haciendo uso de las funciones que se incluyen dentro de dicho paquete, particularmente las implicadas para el análisis ARIMA, se realiza la predicción de información, y utilizando herramientas propias de R, se aplica el formato adecuado previo a la devolución de resultados hacia Java.

Finalmente, el módulo programado en Java recupera los datos devueltos desde el servicio y los representa en un formato agradable y de fácil comprensión para el usuario (a través de gráficas), dentro de la aplicación web del software *Engine*.

[40] Método matemático para pronosticar la demanda de un producto en un periodo de tiempo.

Capítulo 4. Resultados y conclusiones

Resultados

Una vez desarrollado el módulo de predicción y análisis e incorporado a la herramienta *Engine*, se evidenció que, tanto para su elaboración como para la de cualquier aplicación empresarial, son requeridos diversos conocimientos, todos ellos adquiridos dentro de la carrera de Ingeniería en Computación. De entre los que se aplicaron a este proyecto, se encuentran: redes de datos, ingeniería de software, bases de datos e inteligencia artificial.

Además, por lo extenso y robusto del proyecto fue necesario hacer uso de diferentes herramientas que trabajan de manera conjunta, buscando la mejor manera de aprovechar las cualidades de cada una y así entregar al usuario los resultados deseados.

La utilidad que se le puede dar a la aplicación desarrollada es bastante extensa, pues depende únicamente de los datos históricos recopilados, definidos por el interés particular del usuario final. De esta forma se pretende que esta herramienta no esté enfocada hacia un sector específico, sino que se extienda a cualquiera, buscando ser una solución útil para la mayor cantidad de usuarios y maximizando la valía de la misma.

Es por ello que el principal resultado, que se encuentra implícito en el proyecto, es precisamente el valor que le aporta a los analistas, proveyéndoles de una interpretación rápida y comprensible sobre comportamiento futuro de los datos de interés, a partir de la exploración de la enorme cantidad de información generada a través de Internet.

No obstante, a través del desarrollo se encontraron también ciertas complicaciones que, si bien no impiden la ejecución del módulo desarrollado, sería recomendable resolverlas a futuro para optimizar su funcionamiento. Dentro de estas se encuentran las siguientes: una mejor limpieza de datos, útil para discriminar

información imprecisa; la recolección de una mayor cantidad de información, pues entre más contenido analizado, los resultados entregados suelen ser más precisos y la representación y utilización de umbrales de error, para ser incluidos en la visualización de la predicción, valiosos para que el usuario entienda qué tan certero es el resultado representado, entendiendo también que, entre más datos analizados y recopilados, dichos umbrales se reducirían considerablemente.

Ahora bien, durante la búsqueda y posterior puesta en marcha de un modelo matemático para la obtención de datos futuros, se fue descubriendo que una de las mejores alternativas era la utilización de algún modelo auto-regresivo, derivando finalmente el empleo del modelo ARIMA. Este, aplica a su vez un proceso estocástico para la generación de ruido blanco, que en términos numéricos representa la naturaleza aleatoria de la información.

La representación visual realizada de los datos estimados y la cantidad de puntos dibujados en la gráfica, utilizan el mismo formato de los que se encuentran representados previamente, es decir, dado que en las diferentes gráficas se presenta a la información como un recuento diario, los resultados obtenidos por parte del modelo se ilustran de igual manera. Sin embargo, el usuario tiene la posibilidad de fijar un rango especifico de puntos a mostrar a futuro, siendo entre más grande menos preciso. Esto último se debe a que, debido a la naturaleza del modelo, que se sustenta en la información histórica recopilada, entre más alejados estén los resultados, el análisis implícito tendrá que utilizar datos anteriores no verídicos (sino estimados) para aproximar un valor, aumentando en consecuencia el umbral de error presente.

Entre estas escalas de representación para los datos predichos se encuentran: tres días, una semana, quince días y un mes; segmentación que se hizo pensando tanto en la flexibilidad brindada para el usuario, como en los umbrales de error citados anteriormente. Así, la escala de una semana, por ejemplo, que cuenta con siete puntos estimados, tiene un error menor que la de un mes, teniendo esta última 30 puntos, es decir, 23 puntos más y posteriores a la anterior.

Conclusiones

Analizando la información que arroja la herramienta desarrollada, incluida en la aplicación empresarial del software *Engine*, se concluye que los objetivos se cumplieron, pues la utilización de dicha herramienta, en conjunto con los otros módulos del sistema, hacen posible la recuperación de información sumamente útil para los usuarios finales, quiénes contarán con elementos tangibles y bien fundamentados, provenientes del análisis de grandes cantidades de información, que ayudarán a dichos analistas para la toma de decisiones futuras.

Con base en los resultados se puede decir que la hipótesis propuesta se comprueba, dado que al utilizar herramientas de *big data* para analizar información de diferentes fuentes abiertas, empleando también la teoría matemática inherente que hace uso de los procesos estocásticos, es posible entregar al usuario información sumamente útil, brindándole la posibilidad de implementar estrategias para modificar esos índices a su favor.

Es pertinente mencionar que los conocimientos matemáticos necesarios, inherentes al desarrollo fueron más elevados de los enseñados dentro de la Facultad de Ingeniería, por lo que fue preciso solicitar el apoyo de algunos compañeros de licenciatura pertenecientes a la Facultad de Ciencias, quienes incentivaron también a tomar algunos cursos intersemestrales relacionados con dicha materia. Gracias a todos ellos, fue posible comprender, con cierto grado de detalle, los temas relacionados con procesos estocásticos y los modelos autoregresivos, y que, una vez entendidos, resultó evidente que no eran complicados.

Aquí, existe un área de oportunidad para el mejoramiento de la calidad de la enseñanza dentro de la Facultad, dado que en ciencias básicas no se tiene un nivel como el que se debería alcanzar. Un recién egresado podría imaginar que al salir de la universidad ya cuenta con los conocimientos técnicos y matemáticos suficientes para emplearse dentro del mundo profesional, sin embargo, conforme lo experimenta se dará cuenta de que no es así.

Por ello, se le solicita a la Facultad de Ingeniería considere aumentar este nivel de enseñanza, así como una actualización al temario de la asignatura de estadística y probabilidad, dado que el conocimiento actualmente aportado es apenas suficiente para un profesionista, sobre todo dentro del mercado actual del manejo de datos, en donde materias tales como la anterior son esenciales para dicho desenvolvimiento profesional.

El desarrollo de la presente tesis hizo notar que, una vez terminados los estudios dentro de la Facultad de Ingeniería, todavía no se cuenta con un perfil profesional suficiente para llevar a cabo proyectos empresariales de tal magnitud. Además, hizo posible apreciar que en la industria es preciso colaborar con otros perfiles de profesionistas, tales como informáticos, matemáticos, actuarios, ingenieros de otras especialidades, abogados, contadores, entre otros, por lo que se vuelve una necesidad entender cómo comunicarse con todos ellos adecuada y eficientemente. Sin embargo, en los años de estudio dentro la Facultad, los estudiantes se encuentran aislados del mundo real, específicamente en cuanto al trabajo en conjunto se refiere, por lo que el tener que interactuar con estas diferentes personas durante el desarrollo de este proyecto, constituyó un reto.

En la práctica, se utilizó la metodología ágil Scrum, la cual permitió alcanzar los objetivos planteados, debido a la interacción continua y colaborativa con el *product owner* o el usuario de negocio principal. Esto también ayudó a identificar las cualidades propias de un ingeniero, mismas que lo llevan a participar dentro de cualquier industria, buscando resolver problemas y aportar soluciones innovadoras ante estos, siempre sacando provecho de las bases teóricas con las que cuenta. No obstante, es preciso en todo momento, e inclusive durante el desarrollo de este trabajo, estudiar temas adicionales, útiles para entender nuevos conceptos, tales como financieros y de administración de proyectos.

De entre las experiencias más importantes que dejó esta tesis se encuentra la del acercamiento hacia otros perfiles profesionales. Particularmente con los matemáticos, en quienes se identificó un nivel considerablemente alto en cuanto a conocimientos, pudiéndolos llevar hacia su utilización, y aplicándolos inclusive de

una mejor manera que como lo haría un ingeniero en computación; esto debido a los fundamentos teóricos que poseen.

El tener la oportunidad de colaborar directamente con una empresa fue un suceso sumamente valioso, dado que brindó la posibilidad de dialogar, exponer dudas y trabajar directamente con doctores, quienes tenían un nivel de conocimientos amplio y consolidado en cuanto a lo aplicado; con maestros en ciencias y en computación, quienes establecían mecanismos de control eficiente para los algoritmos y procesos implicados; con informáticos, quienes tenían un control auditable y continuo de los mismos procesos; y con los usuarios finales, que aunque estaban fuera del país, estuvieron dispuestos a trabajar en conjunto para sacar el proyecto en tiempo y forma.

En cuanto a las actividades administrativas, se tuvo un hueco de conocimientos importante, pues, aunque en algunas materias como en Ingeniería de software se adquirieron los fundamentos teóricos y prácticos base, hubo otras, como Administración de proyectos de software, en donde se obtuvieron los necesarios, derivado del contenido deficiente aportado, comprobado tanto el proyecto presentado como en mi trabajo anterior. Se entiende que la Universidad nos da conocimientos base para partir de ellos y responsabilidad de cada uno obtener más, sin embargo, se considera urgente que la Facultad actualice los contenidos y profesores, pues los conocimientos de ciertas materias prácticas de la ingeniería real no corresponden al nivel de enseñanza obtenido, sobre todo del séptimo semestre en adelante.

Un beneficio de la tesis fue repasar prácticamente todas las materias aplicadas de la carrera, en tiempos cortos. Empero, esto permitió identificar la deficiencia de muchos profesores en su calidad de enseñanza, particularmente en materias tales como: bases de datos, administración de proyectos de software, diseño de interfaces para computadoras, lenguajes de programación o programación avanzada de métodos numéricos. Todas ellas necesarias para la presente tesis y en donde se pudo notar un déficit en la provisión de los conocimientos mínimos necesarios, teniendo como base el temario del plan de estudios de 2010.

Aunado a lo anterior, existió también la falta de como becario en algún programa de la UNAM, en el trabajo actual hay varios ex becarios de DGTIC y tienen un buen nivel, pese a que ya tengo un par de años de experiencia profesional y dos empresas para las cuales trabajé, la industria no se preocupa por enseñarle a la gente, son pocas las empresas que le invierten a sus empleados, de hecho en mi caso me entrené por mi cuenta en cursos que noté necesitaba para poder desempeñarme en otras empresas, siendo que esos conceptos debieron enseñárnoslos en la Facultad, sin embargo el conformismo y apatía que como alumno tuve evitó que exigiera a mis profesores fueran cabales en su enseñanza.

Asimismo tuve algunos, pocos, excelentes profesores que, en efecto, lo que nos dijeron en clase me ha ocurrido en la vida profesional, sin embargo ninguno de ellos hoy día es profesor de la Facultad, lo cual es preocupante, dado que es necesario tengamos más profesores con un nivel y experiencia en el mundo real que nos ayude a mejorar, nos han acostumbrado durante la carrera a los nombres de ciertos profesores, penosamente al salir a trabajar, nos topamos con la triste realidad de que es obsoleto o ya no se ocupa lo que nos enseñaron esos famosos profesores.

Recomendaría también tomar clases en otras Facultades, a destiempo supe que podía cursar materias en otras, pero al trabajar con matemáticos y actuarios me abrieron la visión ampliamente en las aplicaciones de la ciencia, y se supone que es lo que hacemos los ingenieros, pero veo que tenemos un nivel inferior. También mis compañeras de informática están muy bien preparadas en temas de dirección de proyectos, métricas, costeos y auditorias de procesos, de lo cual afortunadamente me han enseñado, es notorio que las gerentes en la empresa son de informática, los investigadores principales vienen de ciencias y algunos buenos gerente son de ingeniería, distinto a lo que nos enseñan en la Facultad, debemos tener apertura a otras carreras y perfiles.

Finalmente, ante lo expuesto, recomiendo y valido lo importante de hacer una tesis escrita, permite consolidar conocimientos, repasar materias y sobre todo valorar lo que no enseñaron (o no) en la Facultad. Si bien es cierto, me costó trabajo y tiempo el desarrollo de la misma, pero considero valió la pena el esfuerzo, no solo por el

repaso y aprendizaje, pero también por la valorización de esos temas que uno como alumno no valora en la vida académica: conocer otros perfiles, colaborar con otros proyectos, trabajar en equipo, saber comunicar y sobre todo mantenerse actualizado.

Bibliografía

Aggarwal, C. C. (2011). An introduction to social network data analytics. In C. C. Aggarwal, *Social network data analytics* (pp. 1-15). Estados Unidos: Springer.

Apache Software Foundation. (2016). *Apache Maven Project*. Retrieved marzo 2, 2016, from https://maven.apache.org

Apache Software Foundation. (2016). *Apache Tomcat®.* Retrieved from http://tomcat.apache.org/index.html

Arenas Delgado, G., de la Peña Leal, I., García Guerrero, S., Llamas Garcinuño, E. M., Bruno Paniagua, M., & López Viñas, J. (2016). *APACHE SOFTWARE FOUNDATION*. Retrieved febrero 24, 2016, from http://apachefoundation.wikispaces.com/

Bernal, C. A., Sierra, H. D., Cortés, A., & García, M. (2013). *Proceso administrativo para las organizaciones del sigo XXI.* Bogotá, Colombia: Pearson.

Bressolf, P. C. (2014). *Stochastic Processes in Cell Biology.* Berlín, Alemania: Springer.

Chung, W. (2014). BizPro: Extracting and categorizing business intelligence factors from textual news articles. *International Journal of Information Management, 34*, 272-284.

Crawley, M. J. (2007). *The R book.* New York: John Wiley & Sons.

Darnall, R., & Preston, J. M. (2010). *Project Management From Simple to Complex.* Flat World Knowledge, Inc.

de Arce, R. (2006). Modelos ARIMA. *Programa Citius. Técnicas de Previsión de variables financieras*.

Dudney, B., Lehr, J., Willis, B., & Mattingly, L. (2004). *Mastering JavaServer Faces.* Nueva York, Estados Unidos: Chichester: Wiley.

Feldman, R. (2013). Techniques and applications for sentiment analysis. *Communications of the ACM, 56 (4)*, 82-89.

Fernández, Y., & Díaz, Y. (2012, enero-abril). Patrón Modelo. *Telem@tica. Revista digital de las tecnologías de la información y las comunicaciones*, 47-57.

Gandomi, A., & Haider, M. (2014). Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management*, 137-144.

García, J., Ignacio, J., Mingo, I., Imaz, A., Brazález, A., Larzabal, A., . . . García, J. (2000). *Aprenda Java como si estuviera en primero.* San Sebastián, España: Escuela Superior de Ingenieros de San Sebastián.

Hahn, U., & Mani, I. (2000). The challenges of automatic summarization. In IEEE, *Computer, 33(11)* (pp. 29-36).

He, W., Zha, S., & Li, L. (2013). Social media competitive analysis and text mining: A case study in the pizza industry. *International Journal of Information Management, Volume 33, Issue 3*, 464-472.

Heidemann, J., Klier, M., & Probst, F. (2012). Online social networks: A survey of a global phenomenon. *Computer Networks, Volume 56, Issue 18*, 3866-3878.

Hirschberg, J., Hjalmarsson, A., & Elhadad, N. (2010). "You're as sick as you sound": Using computational approaches for modeling speaker state to gauge illness and recovery. In A. Neustein, *Advances in speech recognition* (pp. 305-322). Estados Unidos: Springer.

IEEE Computer Society. (2004). *Guide to the software engineering body of knowledge 2004 version : SWEBOK.* Los Alamitos, California.

Ihaka, R. (1998). R : Past and Future History. *Computing Scince and Statistics*, 392-396.

International Business Machines Corp. (2012, junio 18). *IBM Developerworks.* Retrieved septiembre 10, 2016, from ¿Qué es Big Data?: https://www.ibm.com/developerworks/ssa/local/im/que-es-big-data/

Jiang, J. (2012). Information extraction from text. In C. Aggarwal, & C. Zhai, *Mining Text Data* (pp. 11-41). Estados Unidos: Springer.

Khare, T. (2012). *Apache Tomcat 7 Essentials.* Birmingham, Reino Unido: Packt Publishing.

Kurniawan, B. (2015). *Servlet and JSP: A Tutorial.* Quebec, Canadá: Briany Software Inc.

Labrinidis, A., & Jagadish, H. (2012). Challenges and opportunities with big data. *Proceedings of the VLDB Endowment*, 2032-2033.

Liben-Nowell, D., & Kleinberg, J. (2003). The link prediction problem for social networks. *Proceedings of the twelfth international conference on information and knowledge management* (pp. 556-559). Nueva York, Estados Unidos: Association for Computing Machinery.

Martínez, F., & Chávez, G. (2010). *Administración de proyectos guía para el aprendizaje.* México D. F.: Pearson, Prentice-Hall.

Mascareñas, J. (2013). Procesos estocásticos: introducción. *Monografías de Juan Mascareñas sobre Finanzas Corporativas*.

Münch, L., Flores, B. E., & Cacho, I. (2014). *Administración : gestión organizacional, enfoques y proceso administrativo.* Naucalpan de Juárez, Estado de México: Pearson.

Oracle Corporation. (2016). Retrieved from Java Servlet Technology Overview: http://www.oracle.com/technetwork/java/javaee/servlet/index.html

Ortega, J. (2014). Cadenas de Markov. In J. Ortega, *Modelos estocásticos I* (pp. 45-87). Guanajuato, México: Centro de Investigación en Matemáticas.

Pacheco, P. (2011). *An introduction to parallel programming.* Massachusetts, Estados Unidos: Morgan Kaufmann.

Parthasarathy, S., Ruan, Y., & Satuluri, V. (2011). Community discovery in social networks: Applications, methods and emerging trends. In C. C. Aggarwal, *Social network data analytics* (pp. 79-113). Estados Unidos: Springer.

Patil, H. A. (2010). “Cry baby”: Using spectrographic analysis to assess neonatal health status from an infant's cry. In A. Neustein, *Advances in speech recognition* (pp. 323-248). Estados Unidos: Springer.

Project Management Institute. (2013). *Guía de los fundamentos para la dirección de proyectos (Guía del PMBOK®).*

Rincón, L. (2012). Introducción a los procesos estocásticos. *Prensas de Ciencias*.

Ruíz, M. (2006). *Procesos estocásticos.* Cartagena, España: Universidad Politécnica de Cartagena.

SCRUMstudy. (2013). *Una gía para el conocimiento de Scrum (Guía SBOK™).*

Sommerville, I. (2011). *Ingenieria de software.* México: Pearson - Addison wesley.

SpringSource. (2016). *Introduction to the Spring Framework*. Retrieved from Part I. Overview of Spring Framework: http://docs.spring.io/spring/docs/current/spring-framework-reference/html/overview.html

Toro López, F. J. (2013). *Administración de proyectos en informática.* Bogotá, Colombia: Ecoe Ediciones.

Ward, J. S., & Barker, A. (2013, septiembre 20). Undefined By Data: A Survey of Big Data Definitions. *eprint arXiv:1309.5821*, 2. Saint Andrews, Fife, Escocia: Universidad de Saint Andrews.

White, T. (2015). *Hadoop : the definitive guide.* Sebastopol, Crimea: O'Reilly Media.

Wirth, N. (2008). A Brief History of Software Engineering. *IEEE Annals of the History of Computing*, 32-39.

Anexo A. Glosario de términos

AOP - Programación orientada a aspectos (*Aspect Oriented Programming*)

API - Interfaz de programación de aplicaciones (*Application Programming Interface*)

ARIMA - Modelo autorregresivo integrado de media móvil (*autoregressive integrated moving average*)

ASF - *Apache Software Foundation*

EE - Edición empresarial (*Enterprise Edition*)

IoC - Inversión de control (*Inversion of Control*)

JEE - Java edición empresarial (*Java Enterprise Edition*)

JPA - API de persistencia de Java (*Java Persitance API*)

JSE - Java edición estándar (*Java Standard Edition*)

JSF - *JavaSever Faces*

JVM - Máquina virtual de java (*Java Virtual Machine*)

MVC - modelo-vista-controlador (*model-view-controller*)

PMBOK - Guía de los fundamentos de gestión de proyectos (*Project Management Body of Knowledge*)

POJO - *Plain Old Java Object*

SBOK - *Scrum Body of Knowledge*

SDK - Kit de desarrollo integrado (*software development kit*)

SE - Edición estándar (*Standard Edition*)

SWEBOK - *Software Engineering Body of Knowledge*

Este glosario contiene únicamente las principales abreviaturas usadas dentro de la presente tesis.