UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Sistema de reportes Web para el área de ventas de empresa de ventas por catálogo

INFORME DE ACTIVIDADES PROFESIONALES

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A

Carlos Eduardo Salazar Cárdenas

ASESOR DE INFORME

Ing. Jorge Alberto Rodríguez Campos

Ciudad Universitaria, Cd. Mx., 2018

Agradecimientos

En honor y agradecimiento a mi padre que con su guía y fuerza hizo de mi hermano y de mí hombres de bien. Por ser mi más grande motivación en todo lo que hago.

. . .

A mi madre, por la constancia, devoción y cariño con la nos educó. Por ser un gran ejemplo a seguir como profesional, líder y ser humano.

A mi noble hermano Daniel, por ser el compañero que siempre está a mi lado de la mejor forma.

A mis abuelos Aurora y Antonio por demostrarme tanto amor y comprensión a lo largo mi vida.

A mi tío Caleb, por ser ese apoyo incondicional tan importante en los buenos y malos momentos.

Índice de figuras

Introducción.

El presente documento es un reporte de la actividad profesional enfocada al área de desarrollo de software, que realicé en una empresa líder en ventas por catálogo a nivel mundial (en adelante se hará referencia como “la empresa”), en la cual desarrollé actividades propias del rol de programador Java Sr. para el área de publicidad y ventas, dentro de las cuales, pude diseñar, programar e implementar soluciones de gran innovación y beneficio para la empresa, durante el periodo de junio de 2015 a febrero de 2017.

El informe abarca la descripción de las actividades de análisis, diseño e implementación de uno de los proyectos más grandes en los que estuve involucrado en la empresa. El sistema que desarrollé junto con un equipo de programadores Java, tuvo un gran alcance e impacto en los usuarios finales, así como el beneficio que representó para la organización y la relación que se tuvo con diferentes áreas en cuanto a dependencias y entregables.

Uno de los puntos importantes a resaltar de este documento es la forma en la que se abordó el desarrollo de la solución de software para las necesidades de negocio en el proyecto. En particular el uso de marcos de referencia, bibliotecas para Java y capacidades de software poco exploradas dentro de la empresa, pero de gran uso y credibilidad en la industria del software. En ese sentido, plasmaré la experiencia adquirida al enfrentar las negativas por parte de gerencia con respecto al uso de nuevas tecnologías dentro de un ambiente corporativo estricto.

Para la construcción de la solución descrita en este documento, se utilizó una implementación de la metodología en cascada de desarrollo de software, sobre la cual se llevaron a cabo varias iteraciones a lo largo del proyecto para lograr 2 versiones estables de una aplicación web en producción, con todo lo que ello implica a nivel de servicios, base de datos y despliegue de información; logrando tal impacto en los usuarios, que se buscaría la implementación de este sistema en otros países, posteriormente. Cabe mencionar la elección de la metodología en cascada se decidió a partir del tipo de requerimientos y los periodos de tiempo con los que se contaba para el desarrollo, tema que se abordará más adelante a detalle.

La estructura de documento incluye cuatro capítulos y una sección de conclusiones y recomendaciones para desarrollos futuros, que se describen a continuación:

En el capítulo 1 se describe en términos generales la estructura organizacional de la empresa, así como la forma en la que se tiene establecida el área de tecnología y de desarrollo de software en cuanto a herramientas, soporte y requerimientos. Se resaltan los objetivos a nivel de negocio.

En el capítulo 2, se especifica el marco teórico utilizado en el desarrollo del sistema. Se abarca la definición de ITIL (Information Technology Infrastructure Library), de forma muy concreta y se puntualiza la forma en la que se implementa en la empresa, en orden de poner en contexto el caso de estudio para abordar los temas que en adelante se desarrollan.

Así mismo, se abarca la metodología en cascada para el desarrollo de software y en general, se puntualizan todos y cada uno de los conceptos requeridos tanto a nivel técnico como a nivel de negocio.

En el capítulo 3, se detalla el requerimiento realizado por las áreas de negocio hacia el área de desarrollo de software correspondiente a la construcción del sistema que se aborda en este documento. Se explica cómo surgió esta necesidad y se da un contexto más amplio de la vertical de negocio sobre la cual se trabajó.

Adicionalmente, se describen las ventajas que representaría el sistema para la organización, mismas con las que se impulsó este proyecto en términos de presupuesto. Adicionalmente, se revisa la definición del sistema que se realizó por el equipo de desarrollo junto con el equipo encargado de la definición del producto, en términos de reglas de negocio para definir los procesos requeridos de software. Se puntualizan los criterios de aceptación establecidos por los usuarios.

En el capítulo 4, se describen las particularidades del proyecto que fueron requeridas para la construcción del sistema, desde el análisis hasta la liberación, bajo las reglas de la especificación de ITIL. En este capítulo, también se entra en el detalle técnico de las tecnologías, las técnicas de desarrollo, la gestión de dependencias con otros equipos, los periodos de prueba requeridos, el uso de los recursos de programación, la distribución de las tareas dentro del equipo, la gestión gerencial, la documentación y las asignaciones de trabajo dentro y fuera del equipo.

Adicionalmente, se revisa a detalle el pase a producción del sistema, se abordan los puntos importantes y los problemas con los que nos enfrentamos a nivel

técnico. Se da una breve explicación de los procesos requeridos para la implementación de las aplicaciones bajo la especificación en ITIL y se puntualizan las especificaciones no funcionales que definimos para este sistema. Se presenta el proceso de soporte a la aplicación que se siguió para asegurar la calidad del servicio.

Finalmente, en la sección de conclusiones, se hace una recapitulación de todos los beneficios que la implementación del sistema trajo a la organización y de la proyección que ganó a nivel Latino América, gracias a su aceptación por parte de los usuarios finales. Se presentan los requerimientos y negociaciones que se sostuvieron para su implementación en otros países.

Capítulo 1. Estructura organizacional y objetivos de negocio de la organización

1.1. Generalidades de la empresa

La empresa a la que se hace referencia en este documento, se encuentra en el giro de ventas por catálogo con más de seis millones de representantes de ventas a nivel mundial y con presencia en más de 50 países alrededor del mundo, en cada uno de los cuales se ha consolidado como una de las marcas más relevantes de este giro. La empresa ofrece una amplia variedad de productos, que va desde contenedores herméticos hasta cosméticos y perfumes de alta calidad.

Históricamente la empresa, se ha caracterizado por apoyar el empoderamiento económico femenino y fomentar la belleza con propósito de la mujer independiente. Cuenta con 130 años en el mercado y con 60 años de haber llegado a nuestro país, donde ha presentado crecimiento mucho mayor del esperado. Actualmente, México es el 3° mercado más importante de la compañía, después de Rusia y Brasil, ya que cuenta con casi un millón de representantes de ventas activas.

La empresa se distingue por reflejar innovación y calidad en sus productos, así como apoyar activamente el optimismo femenino con la confianza y respeto que brinda a sus clientes en todos y cada uno de sus servicios. Cabe mencionar que la compañía está abierta a aspirantes a representante de ventas del género masculino, también.

Actualmente, los objetivos de la empresa en México, trascienden las fronteras del país, pues se desarrollan sistemas, modelos de negocio, campañas de marketing (publicidad) y diseño de folletos para toda Latino América.

1.2. Situación de la empresa en México

La empresa ha logrado establecer en México una operación de calidad mundial, sumamente organizada, funcional y que cuenta con una red de inventario, empaquetado, distribución y cobranza de primer nivel, implementada a lo largo de todo el país. Es importante mencionar que la red de ventas de la empresa abarca casi toda la República Mexicana, dividendo la distribución por secciones, regiones y zonas en un esquema multinivel en el que las mismas representantes de ventas tienen a su cargo a otras representantes, convirtiéndose en líder de ventas, y

obtienen beneficios por el rendimiento de su la red de ventas que ellas mismas construyen. A su vez las regiones y zonas de ventas están a cargo de personal altamente capacitado que da asesoramiento y continuidad a las representantes y líderes de venta para asegurar su éxito.

La empresa cuenta con una planta productora y empaquetadora de calidad mundial en Celaya, Guanajuato, la cual tiene un sistema robotizado para el empaquetamiento automático de los productos por cada orden de compra. Este gran centro, permite también a especialistas de diferentes tecnologías y líneas profesionales, desempeñarse dentro del negocio de la empresa.

Así mismo, la empresa cuenta con sus oficinas centrales de México en Polanco, Ciudad de México para las áreas de publicidad, manejo de producto, planeación, legal, contable y diseño. Adicionalmente la empresa, tiene una sede tecnológica enfocada a los proyectos de desarrollo y operación de software, ubicada en Xola, Ciudad de México. En la cual distintas áreas de desarrollo, soporte técnico, configuración, mantenimiento y administración dan atención a cada una de las áreas de negocio de la empresa.

Por último, es importante mencionar que la empresa cuenta con diversos ‘call center’, distribuidos en la Ciudad de México, para dar apoyo, soporte y capacitación a sus representantes de ventas de todo el país.

No obstante, al ser una empresa originaria de Estados Unidos de Norte América, toda la gestión y planeación a nivel administrativo se reporta directamente a la sede de la empresa en Nueva York, Estados Unidos. Por lo que todos los empleados de la empresa México deben contar con un inglés escrito y conversacional fluido.

La organización en México de la empresa es realmente muy destacada y ha dado pie a que las sedes de la empresa en otros países pongan atención a la innovación que se está realizando en nuestro país para poder implementarla en sus mercados y así poder alcanzar el crecimiento que ha tenido hasta el momento la empresa en México. Este hecho ha incluso propiciado a que proyectos de software creados en México, sean replicados en países de Latino América.

1.3. Estructura organizacional

La empresa cuenta con una estructura organizacional matricial tanto para México como para todos los demás países de su operación, lo cual le permite a los recursos de la empresa desenvolverse en diferentes tipos de proyectos, con alcances y actividades muy particulares en cada uno de ellos, adquiriendo una experiencia multidisciplinaria muy valiosa al involucrarse con distintas áreas de especialidad.

Puntualizado, las estructuras jerárquicas matriciales consisten en que todos y cada uno de los recursos humanos de la empresa reportan a un jefe directo inmediato pero a la vez pueden involucrarse directamente con otras áreas o incluso reportar a otro gerente/director durante un periodo de proyecto determinado si así lo requieren las necesidades de la empresa en ese momento, lo cual puede resultar muy interesante a nivel tanto para los jóvenes que comienzan su carrera como para los profesionales de mayor experiencia pues se puede llegar a tener bastante exposición con otras áreas de México, Latino América y Estados Unidos, lo cual podría representar un cambio en la asignación de actividades y paulatinamente un ascenso.

La organización jerárquica de la empresa a nivel México en el área de tecnología se divide en una dirección general a nivel Latino América a la que reportan 3 direcciones de especialidad a nivel México, como se muestra en la figura 1.

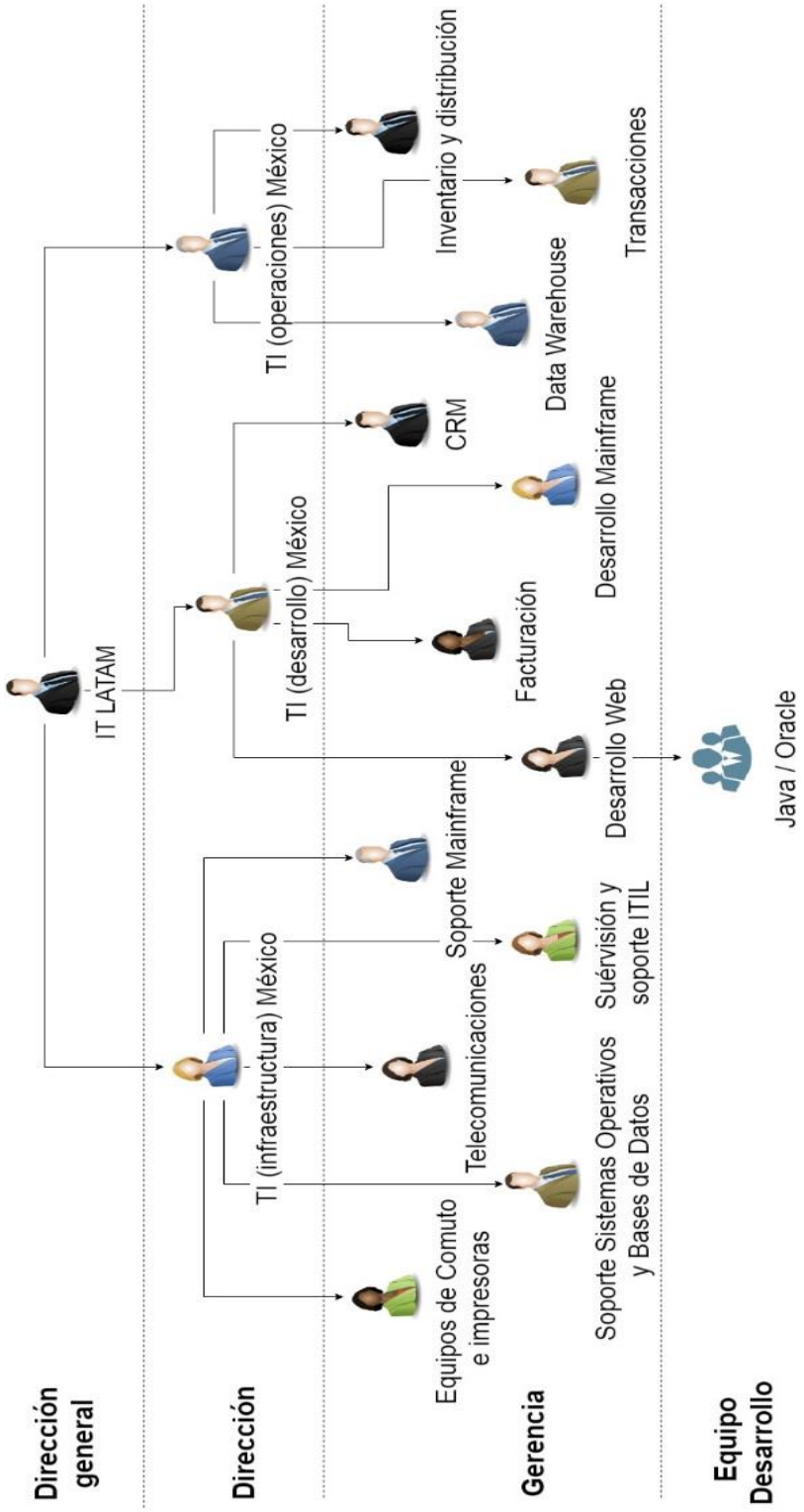


Figura 1. Estructura jerárquica de la empresa

Fuente: Elaboración propia

Al mismo tiempo, la dirección de la empresa Tecnológica en Latino América tiene a su cargo a todas las direcciones de especialidades tecnológicas en los países de Centro América y Sur América en los que tiene presencia la empresa a excepción de Brasil. Por otro lado, la dirección general de Tecnología en Latino América, reporta directamente al cargo de Vice Presidencia Tecnológica de la empresa a nivel NOLA (Norte América y Latino América), quien a su vez reporta directamente a la Presidenta General de TI de la empresa a nivel mundial. Por lo que realmente es una organización en la que no existe una cantidad considerablemente amplia de puestos gerenciales y de dirección entre el puesto más alto a nivel jerárquico con los puestos no gerenciales.

Adicionalmente, es importante destacar que el porcentaje de mujeres en cargos directivos y gerenciales en la empresa es mucho mayor que en otras organizaciones, pues de acuerdo a la investigación realizada por la Doctora Magali Cárdenas en sus investigaciones de equidad de género, menos del 30% de los cargos a este nivel son designados a recursos del género femenino. En el área de TI de la empresa en México, este porcentaje es de poco menos del 50%, lo cual está muy por arriba del promedio reportado y es muy favorable para la equidad de género. No obstante, demuestra una congruencia en cuanto a la cultura y las acciones con la que se lleva la empresa. Incluso, el cargo de la Dirección General a nivel mundial de toda la empresa está a asignado a una profesional de género femenino.

1.4. Objetivos de negocio de la empresa

Al igual que los corporativos de clase internacional, la empresa tiene trazadas métricas y objetivos a nivel financiero y económico a las que las líneas de negocio se alinean y a los que se dedican los mayores esfuerzos de la empresa en todos los niveles. Sin embargo, cada país tiene necesidades particulares que deben ser cubiertas en orden de alcanzar los objetivos globales trazados para la empresa. Para ello, cada país tiene la libertad de definir sus prioridades en orden de alcanzar los objetivos globales.

1.4.1. Objetivos a nivel global

De acuerdo con la página oficial en México, la empresa tiene la visión de ser la Compañía que mejor entienda y satisfaga las necesidades de productos, servicio y autoestima de la mujer en todo el mundo. Es decir, ser la primera elección de las mujeres para adquirir productos de belleza, producir cosméticos y productos de la más alta calidad y a su vez ser la compañía más admirada del giro. Objetivos que además de definir la cultura de la empresa, pretenden un alto crecimiento de ventas.

1.4.2. Objetivos a nivel Latino América

A nivel Latino América, se busca además de cumplir la visión global de la empresa, consolidar todos los países de esta división de la en cuanto a procesos, sistemas y operación. Lo cual, a nivel de TI, implica compartir diseños bases de datos, arquitecturas de aplicaciones e incluso administración y configuración. Por ello, se busca que todos los desarrollos contemplen la opción multinacional, es decir, ser capaces de soportar la operación de otros países además del inicial, lo cual se aborda más adelante.

Uno de los objetivos más importantes en particular para el área de desarrollo de software es la refactorización, optimización y reutilización de código y aplicaciones en orden de alcanzar más y mejores resultados con el menor del uso de Hardware posible, lo que representa un gran ahorro para la empresa en cuanto el consumo de recursos. Adicionalmente, se busca que el área de sistemas de Latino América funcione como una sola unidad, en cuanto a soporte, gestión de dependencias y proyectos multinacional, cabe mencionar que aún se está trabajando en este objetivo dentro de la empresa en Latino América.

1.4.3. Objetivos a nivel México

Los objetivos de la empresa para nuestro país en particular, se enfocan en el crecimiento de la plantilla de ventas, así como en la retención de la misma, mediante un alto nivel de atención y servicio al cliente. Por lo que, a nivel de TI, muchos de los esfuerzos de desarrollo, mantenimiento, soporte y administración van enfocados a estas verticales. Derivado de los esfuerzos destinados a este objetivo se ha implementado un sistema de métricas de atención para las áreas de TI enfocadas en dar soporte a las representantes de ventas muy estricto y que permite a los niveles gerenciales identificar las áreas de oportunidades de estos equipos.

1.4.4. Mi rol en la empresa

El equipo de desarrollo conformado para la construcción de la solución de software descrita en los siguientes capítulos, constó de 4 personas dedicadas de tiempo completo a este proyecto. El equipo que desarrolló la solución posteriormente descrita fue el equipo de Desarrollo Web para Ventas y Marketing.

Dada la importancia de este proyecto el equipo constaba de 3 desarrolladores Java, todos expertos en desarrollo Web con Java y cada uno con habilidades

particulares. En la figura 2, se muestra la organización interna del equipo de desarrollo.

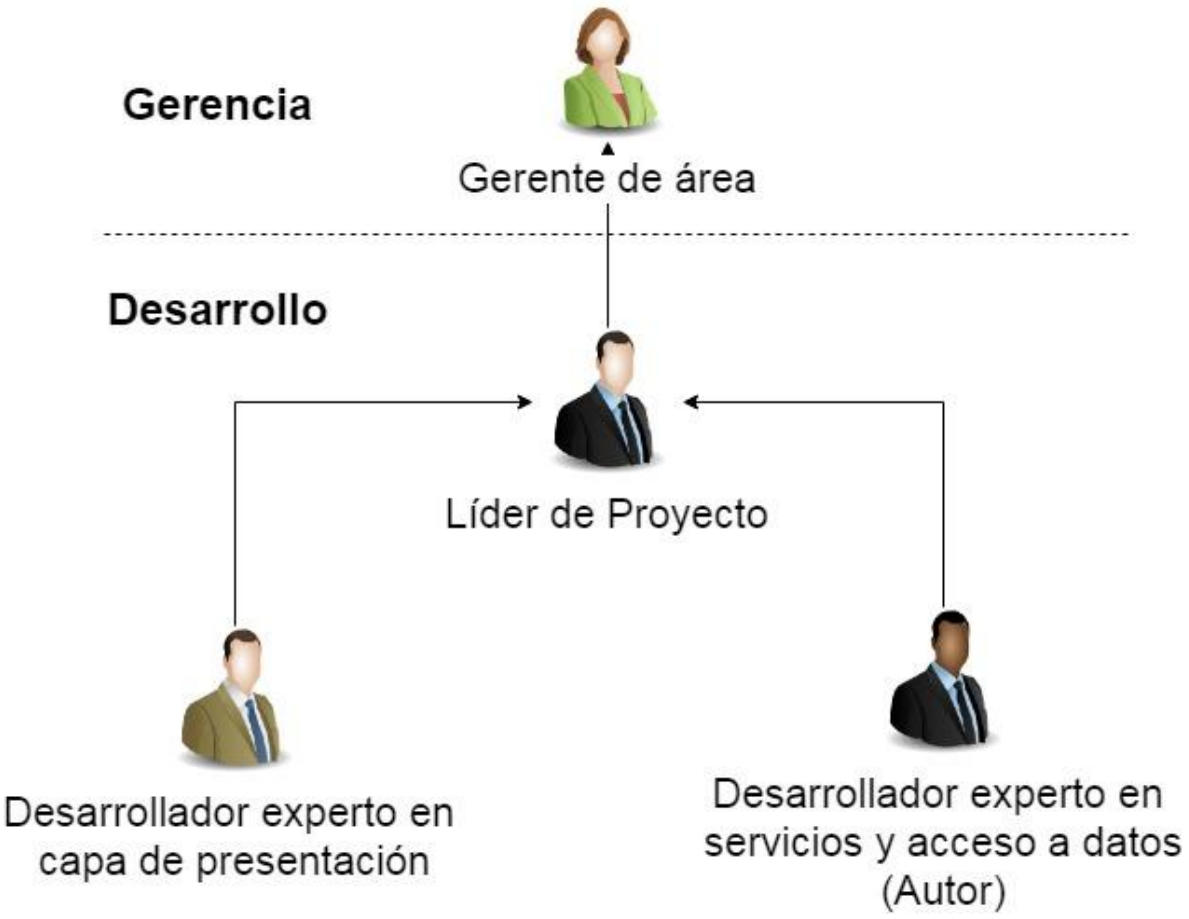


Figura 2. Equipo de desarrollo del área de TI en México para ventas y publicidad

Fuente: Elaboración propia

El rol de cada uno de los desarrolladores del equipo fue el siguiente:

- **Gerente de área.-** definición de aplicación en términos de negocio, aseguramiento de cumplimiento de requerimiento, manejo de tiempos y costos, definición de compromisos de tiempo y alcance.
- **Líder de proyecto.-** gestión de dependencias con otros equipos, definición de tiempos y asignación de tareas, definición de arquitectura general de la aplicación, apoyo con impedimentos técnicos.
- **Experto en capa de presentación.-** definición de arquitectura y desarrollo de la capa de presentación, configuración de segmentos de lectura de

actividad, para alimentación de métricas, gestión de despliegue de etiquetas y recursos web.

- **Experto en capa de servicios.-** definición de arquitectura, seguridad y desarrollo de la capa de consumo de servicios y acceso a datos. Apoyo con el desarrollo de programas para base de datos y capa de presentación, gestión de repositorio y control de versiones de código.

  Definición de arquitectura, seguridad y desarrollo de base de datos para la aplicación, así como programas embebidos en la base de datos. Carga y limpieza de información a base de datos.

En cuanto a las tareas compartidas por los desarrolladores y el líder de proyecto se encontraban las tareas de pruebas unitarias, pruebas de integración, aseguramiento de calidad, seguridad y revisión del código.

Capítulo 2. Marco teórico

A continuación, se abarca la descripción en términos generales de los tópicos técnicos a los cuales se hace referencia en este documento, en orden de poner en contexto el vocabulario y la terminología del proyecto.

2.1 ITSM (Information Technology Service Management)

ITSM es el acrónimo de 'Information Technology Service Management', en español 'Administración de los Servicios de Tecnologías de la Información'. Es una especificación que define un acercamiento estratégico para el diseño, la entrega, la administración y la mejora de la forma en la que se consumen los servicios de TI dentro de una organización, con el objetivo de asegurar los procesos y alcanzar los objetivos de negocio (Verma, 2017).

A partir de esta especificación han nacido muchos marcos de referencia a los cuales las organizaciones se pueden apegar para la implementación correcta de ITSM, tal es el caso de los marcos de referencia descritos a nivel general a continuación:

- **COBIT (Control Objectives for Information and related Technology).-** En español: Control de Objetivos para Tecnologías de la Información. Es un marco de referencia para desarrollar, implementar, monitorear y mejorar los servicios de TI mediante prácticas basadas en gerencia y dirección.
- **ISO 20000.**- Es un estándar global para la implementación de servicios de TI, basado en ITSM y dictada por la IEEE. Da fundamentos a muchos otros marcos de referencia como ITIL y MOF.
- **MOF (Microsoft Operations Framework).-** En español: Marco de Referencia para Operaciones de Microsoft. Es una serie de 23 documentos que definen los procesos de crear, implementar y administrar los servicios de TI de forma efectiva mientras se busca la optimización de costos.
- **ITIL (Information Technology Infrastructure Library).-** En español: Biblioteca de Infraestructura de Servicios Tecnológicos. Es un marco de referencia que dicta una serie de buenas prácticas para la implementación de los servicios de IT, alineados a las necesidades de negocio. Es considerado el marco de referencia más utilizado en el mundo y cuenta con 5 etapas dentro de su especificación, mismas que se revisan adelante.

A continuación, se hace una revisión más a detalle el marco de referencia ITIL ya que es el marco de referencia que utiliza la empresa para la implementación de ITSM (Margaret, Techtarget, 2017)

2.2 ITIL (Information Technology Infrastructure Library)

Originalmente publicado en 1980 con el objetivo de proveer de una referencia estandarizada sobre las mejores prácticas en cuanto a la implementación de ITSM en Reino Unido, ITIL es hoy en día uno de los marcos de referencia más utilizados en la industria de las Tecnologías de la Información.

ITIL está diseñado para estandarizar las prácticas de análisis, planeación, entrega y soporte de los servicios de TI en una organización, su objetivo es precisamente, mejorar la eficiencia de la entrega de los servicios de TI a las áreas de negocio. Actualmente se encuentra en su versión 3, la cual se divide en 5 secciones, como se muestra en la figura 3.

Figura 3. Áreas de servicio de ITIL

Fuente: http://www.dinangreek.net/wp-content/uploads/2016/02/itil.png (9 de Abril de 2017)

- **Estrategia de servicio. -** En esta sección se especifica que todas y cada una de las partes de los servicios que conforman la infraestructura de TI deben estar orientadas y enfocadas, precisamente a casos de negocio con objetivos claros de negocio.

- **Diseño de servicios. -** En esta sección se provee de una guía para la generación y mantenimiento de las políticas y documentos generados para la organización en materia de servicios de TI.
- **Transición de servicios.-** Esta sección se enfoca en el proceso de administración de cambios y liberación. Provee de guías para los procesos de liberación e implementación de servicios de TI en ambientes productivos.

- **Operación de servicios.-** Esta sección se enfoca en los procesos de entrega de servicios de TI. Provee de guías para la ejecución de soporte, liberación y puntos de control.

- **Mejora continua de servicios.-** Esta sección se enfoca en identificar los procesos involucrados en la mejora de los servicios, así como en atender las fallas alrededor de los servicios productivos para su constante mejora.

- **Service Desk (Mesa de servicio).-** Adicionalmente, como parte de la implementación de ITIL, el Service Desk o Mesa de Servicio, permite comunicar los clientes o usuarios con los grupos de servicio, mediante un solo punto de contacto y a su vez, centralizar toda la coordinación de los grupos de trabajo para asegurar la entrega de los servicios de TI. (Margaret, Techtarget, 2017).

El objetivo del Service Desk es alinear la operación diaria de las áreas de TI a ITIL y tener un control documental estricto de la misma. Debido a la gran cantidad de información, lógica de negocio, permisos, roles, etc. el proceso de Service Desk se gestiona mediante una herramienta de software especializada. Dentro de esta herramienta se permiten gestionar, de acuerdo a ITIL, todo el seguimiento a los diferentes tipos de tickets:

- **Incidente.-** Se levanta un ticket de tipo incidente cuando uno o más usuarios reportan una falla en el servicio de TI que consumen. Se asigna un impacto y una urgencia de acuerdo a los usuarios afectados y el proceso de negocio al que atienda.
- **Cambio.-** Se levanta un ticket de tipo cambio cuando un miembro de un grupo responsable de un servicio de TI requiere hacer una modificación a un servicio de TI productivo o requiere agregar un nuevo elemento de configuración a un ambiente productivo.
- **Requerimiento.-** Se levanta un ticket de tipo requerimiento cuando un usuario de negocio o un miembro de un equipo responsable de un servicio

de TI, requiere el apoyo de un grupo responsable de un servicio de TI en particular, para realizar una modificación a un servicio de TI productivo o requiere agregar un nuevo elemento de configuración a un ambiente productivo.

- **Problema.-** Se levanta un ticket de tipo problema cuando se han presentado incidentes similares (que hacen referencia a los mismos elementos de configuración) repetidamente. El problema conlleva a una investigación, para determinar la causa raíz de los incidentes que se han presentado y poder dar una solución definitiva.
- **Investigación.-** Como se mencionar en el punto anterior, un problema conlleva a una investigación, en orden de determinar a detalle la causa raíz de los frecuentes incidentes sobre el mismo elemento de configuración o servicio.

Adicionalmente, un Service Desk es capaz de administrar los siguientes elementos que no son tickets:

- **Elemento de Configuración.-** Es cualquier sistema, servidor, base de datos, paquete o fichero que se involucre con un servicio de TI.
- **Interacción.-** Es el evento en el que un usuario o responsable de los servicios de TI, contacta a un operador de Service Desk, vía telefónica, chat o correo, ya sea para reportar una falla o para dar seguimiento a una existente. Este tipo de ticket no requiere dar seguimiento y solo es para tener un registro.

Cabe mencionar que el objetivo de Service Desk y la administración de incidentes, no es solo gestionar y documentar los procesos, cambios y configuraciones realizadas dentro de una organización a sus servicios tecnológicos. De hecho su objetivo principal es restablecer la disponibilidad de los servicios de la forma más pronta posible, a pesar de que sea con work arounds (soluciones no optimas o temporales) y a partir de ello, solucionar las incidencias de raíz en todos los casos (Ucisa, 2017).

2.2.7 Gestor de service desk en la empresa

Dentro de la empresa, se ha implementado una Administración de la Infraestructura de Tecnologías de la Información basada en ITIL por todos los beneficios que otorga en cuanto a orientar todos los servicios de TI hacia el negocio.

En cuanto el Administrador de Service Desk la empresa cuenta con un sistema especializado de HP para el manejo de este proceso, en el que se lleva el control de todos los tipos de Tickets anteriormente mencionados.

2.3 Ciclo de vida del desarrollo de software

El ciclo de vida del desarrollo de software, se refiere a los pasos a seguir durante el proceso de la construcción de soluciones de este tipo. Existen diferentes implementaciones del este ciclo de vida, como el modelo en cascada, el modelo en espiral, las metodologías ágiles o los proceso iterativos e incrementales. Sin embargo, todas y cada una de estas se comprometen a abarcar la mayoría de las siguientes fases, en orden de asegurar la calidad del software a entregar:

- Planeación del proyecto
- Levantamiento/recepción de requerimientos
- Estimación de tiempo y esfuerzo
- Compromiso de tiempos de prueba y entrega
- Diseño y definición de arquitectura y modelos
- Desarrollo y codificación
- Fase de Pruebas
    - Unitarias y de funcionalidad
    - De integración
    - De rendimiento
    - De seguridad
    - Validación de usuario
- Documentación
- Liberación / Implementación
- Mantenimiento y mejoras

En el caso de la empresa, las metodologías que más se ocupan son el modelo en cascada y el marco de referencia Scrum (Softwaretestingfundamentals, 2017) de acuerdo a las necesidades de cada proyecto. Particularmente, para la solución descrita en este documento, se utilizó el modelo en cascada, por lo que se desarrolla más ampliamente a continuación.

2.3.1 Metodología en cascada para el desarrollo de software

La metodología en cascada es un acercamiento tradicional al ciclo de vida del desarrollo de software que describe el proceso de forma lineal y secuencial durante sus fases, lo cual se observa de forma clara en la figura 4.

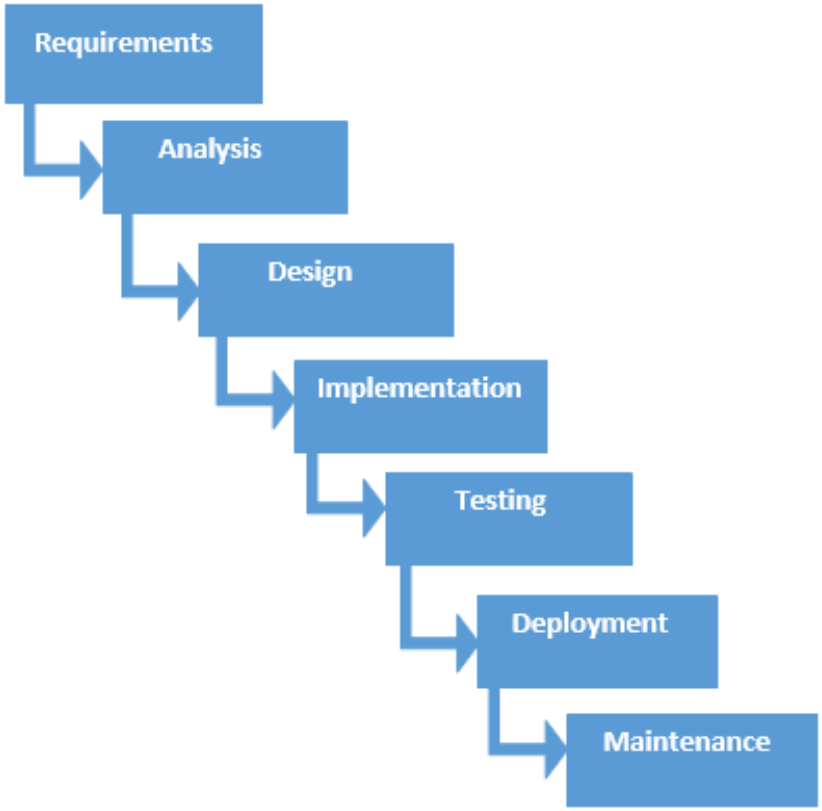


Figura 4. Metodología en cascada para el desarrollo de software

Fuente: elaboración propia con base en (Naveen, 2017).

La metodología permite definir un periodo de tiempo para cada una de sus fases durante las cuales, el equipo de desarrollo se enfocará únicamente en las actividades propias de la fase en curso. Estrictamente, una vez terminado el tiempo designado para cada fase, no se podrá regresar ni tomar tiempo de las fases siguientes para realizar actividades pendientes, por lo que es de suma importancia que los recursos estén dedicados a sus actividades durante 100% del tiempo.

Este acercamiento es muy conveniente cuando se tiene toda la información necesaria para determinar el tiempo y esfuerzo destinado para cada fase, sin embargo puede resultar poco flexible cuando el equipo se enfrenta a dificultades poco visibles durante la planeación del proyecto. Precisamente, dadas estas situaciones, la metodología ha evolucionado a otra metodología llamado ‘cascada Modificado’, el cual permite a los equipos regresar una o más fases para poder

atender cualquier tipo de necesidad o imprevisto y continuar posteriormente con las actividades agendadas. (Margaret, Techtarget, 2017)

Para la solución descrita se implementó la metodología de cascada Modificado, gracias a sus ventajas de planeación y estricta forma documentar cada una de las capas del proceso de desarrollo. Adicionalmente, se adapta a los procesos de pruebas, validación y liberación tan estrictos dentro de la empresa al pasar por cada capa dentro de un solo periodo específico durante el proyecto. No obstante, el levantamiento de requerimientos dentro de la empresa, se realiza una única vez y se espera una sola entrega final por parte del usuario.

La razón por la que, en general, no se utilizó una metodología ágil para el proyecto fue debido a la estricta forma de llevar los proyectos dentro de la empresa, pues los usuarios se comprometen a hacer pruebas y validación de la aplicación una única vez dentro del proyecto, la liberación de aplicaciones tanto en un ambiente de pruebas como en uno productivo es tardada y requiere trámites administrativos costosos en tiempo y los desarrolladores brindan apoyo y soporte a otros proyectos simultáneamente, por lo que no pueden estar dedicados únicamente a las actividades de un solo proyecto.

2.4 Arquitectura de software

El concepto de Arquitectura de software, se refiere a la estructura de un sistema, misma que se compone de elementos con propiedades particulares que se relacionan entre sí. Es decir, la correcta estructuración de los sistemas en un alto nivel con el propósito de que cumpla con requerimientos no funcionales como alto desempeño, seguridad, escalabilidad, y mantenimiento, de acuerdo a lo establecido en cada proyecto. (Microsoft, 2017).

En esta sección, se abarcan los conceptos programación y arquitectura de sistemas que fueron utilizados para el desarrollo e implementación de la solución de software descrita en este documento, así como las aportaciones en cuanto a la toma de decisiones del uso de tecnologías y diseño que realicé.

2.4.1 Programación orientada a objetos

La programación orientada a objetos es un paradigma de programación que propone modelar las soluciones de software, simulando objetos de la vida real. Este paradigma permite definir las estructuras de los datos y las operaciones que pueden ser aplicadas a dichas estructuras pensando en el comportamiento y estructura de las entidades representadas en un ambiente real.

Las principales ventajas de este paradigma, son la herencia, que promueve la reutilización de código, el encapsulamiento, que permite proteger los atributos de cada objeto de ser modificadas directamente por otros objetos. Además permite hacer uso de los métodos de un objeto sin conocer su funcionamiento interno.

Este paradigma también tiene muchos beneficios en cuanto a diseño ya que permite modelar soluciones de software de la misma forma en la que otros sistemas (no computacionales) resuelven un problema. Finalmente, el mantenimiento del software basado en este paradigma se vuelve mucho más sencillo, ya que se tienen sistemas bien estructurados y modularizados (Popyack, Zoski, & Salvage, 2017).

En general, existen ciertos conceptos fundamentales a nivel de definición, en la programación orientada a objetos que permiten un entendimiento mucho más amplio del paradigma.

- **Objeto.-** Son unidades utilizadas para modelar los objetos de la vida real a nivel de software.
- **Clase.-** Es un prototipo que permite definir atributos y métodos que a su vez definirán el estado y comportamiento de los objetos.
- **Interfaz.**- Es un contrato entre una clase y el modelado estructural, que permite una buena estructuración del sistema. Cuando una clase implementa una interfaz, está obligada a proveer el comportamiento publicado por la misma.
- **Paquete.**- Es un espacio nombrado en la estructura del proyecto, para organizar clases relacionadas entre sí de forma semántica y lógica. Sirve para hacer el manejo del mismo mucho más fácil.

A nivel de comportamiento, existen ciertas definiciones en general, que al igual que los conceptos de definición, nos ayudarán a entender el paradigma de la programación orientada a objetos.

- **Encapsulamiento.-** protege los atributos de una clase de intervenciones del exterior. Así mismo remueve la responsabilidad de manejar el funcionamiento interno de los objetos al código exterior.
- **Herencia.-** La herencia provee un poderoso mecanismo para organizar y estructurar el software mediante relaciones. Permite a una clase hacer uso de los atributos y métodos de otra clase de forma segura.
- **Polimorfismo.**- Es la habilidad de un objeto de ser referenciado por distintos tipos de variables que derivan de una misma clase o interfaz.

Una de las implementaciones más representativas de este paradigma es el lenguaje de programación Java, mismo que se utilizó para el desarrollo de la solución descrita (Oracle, 2017).

2.4.2 Patrones de diseño de software

Los patrones de diseño son soluciones de programación repetibles para un problema de diseño de software común. Son una descripción de cómo resolver los dichos problemas y pueden ser usados en diferentes situaciones. Regularmente hacen de la ejecución del software más rápida y efectiva. Además, de favorecer el fácil entendimiento y mantenimiento del código, mediante el uso de estándares en la programación. (Sourcemaking, 2017)

En términos generales, existen tres tipos de patrones de diseño de software:

- **Patrones de creación.-** Estos patrones de diseño se enfocan en la definición de clases y la instanciación objetos, en orden de asegurar la efectividad del proceso de la creación de objetos.
- **Patrones de estructura.-** Estos patrones de diseño se enfocan en la composición de las clases y los objetos, definen formas de componer objetos para heredar funcionalidad de forma eficiente.
- **Patrones de comportamiento.-** Estos patrones de diseño se enfocan en la comunicación entre objetos de clases.

2.4.3 Arquitectura orientada a servicios

La arquitectura orientada a servicios, define una colección de módulos que se comunican entre sí, cada uno de ellos se encarga de una sola función, en orden de favorecer el bajo acoplamiento y la alta cohesión del código, además de facilitar la escalabilidad y el mantenimiento al tener módulos dedicados a una sola tarea en particular. Esta arquitectura promueve la independencia de los módulos, así como la integración de nuevos módulos de forma natural.

De esta forma, los sistemas cliente podrán consumir cada uno de los servicios de acuerdo a los procesos de negocio de forma agnóstica y eficaz. Esta arquitectura es conveniente, tanto en ambientes corporativos sino también en medianas empresas y startups tecnológicas. (Linthicum & Giza, 2017). En la figura 5 se puede observar en términos generales la definición de una arquitectura cliente servidor de acuerdo a Oracle.

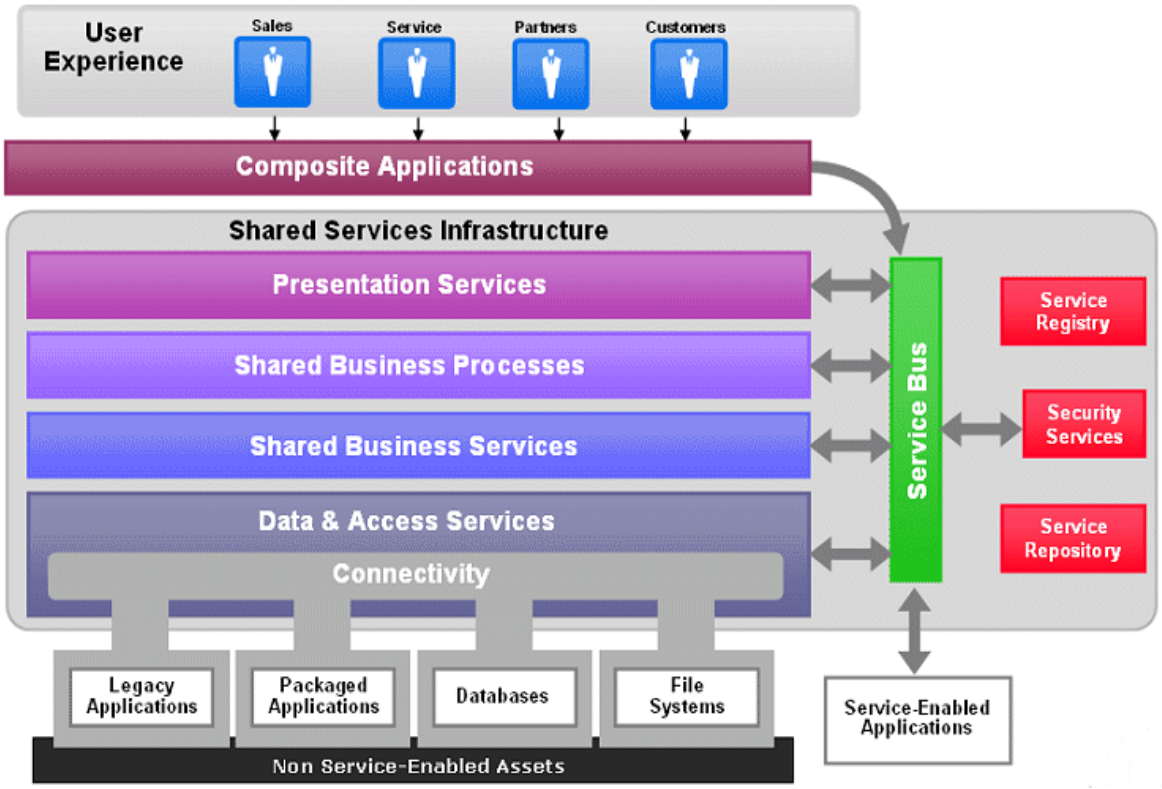


Figura 5. Arquitectura orientada a servicios

Fuente: http://docs.oracle.com/cd/E13171_01/alsb/docs30/concepts/wwimages/SOA_SuccessArch.gif (9 de Abril de 2017)

Cabe mencionar, que la implementación de esta arquitectura no va ligada con ningún lenguaje de programación, base de datos, ni herramienta. Basta con la construcción de servicios eficientes y confiables, que expongan métodos de comunicación seguros.

2.4.4 Arquitectura de software por capas

Esta arquitectura consiste en una serie de componentes independientes, con un rol y una responsabilidad específica dentro de la aplicación. El objetivo de esta arquitectura es facilitar la construcción de modelos efectivos y estructurar los proyectos de una forma ordenada y escalable. Adicionalmente, agiliza el desarrollo, las pruebas, el mantenimiento y la administración de las aplicaciones, gracias al alcance bien definido de cada uno de los módulos. (Wayner, 2017)

En general, para la implementación de esta arquitectura en ambientes corporativos, se definen 4 capas, las cuales se describen a continuación (Richards, Safaribooksonline, 2017).

- **Capa de presentación.-** en esta capa se tiene toda la interacción con el cliente o usuario del sistema o servicio, tanto de entrada como de salida.
- **Capa de negocio.-** en esta capa se realizan todas las validaciones y procesos de negocio pertinentes a la aplicación y las funciones de la misma.
- **Capa de persistencia.-** en esta capa se invocan todas las operaciones relacionadas de consulta y modificación de información en las bases de datos de forma programática, así como las veces de mapeo de la información recibida.
- **Capa de Base de Datos.-** esta capa es en la que se encuentran precisamente las bases de datos y se almacena toda la información relacionada con el negocio y las aplicaciones.

Cabe mencionar que la arquitectura de software por capas, permite añadir o quitar las capas que sean necesarias de acuerdo a la solución que se busque implementar. La estructura en general de la arquitectura de software por capas, se ve representada en la figura 6.

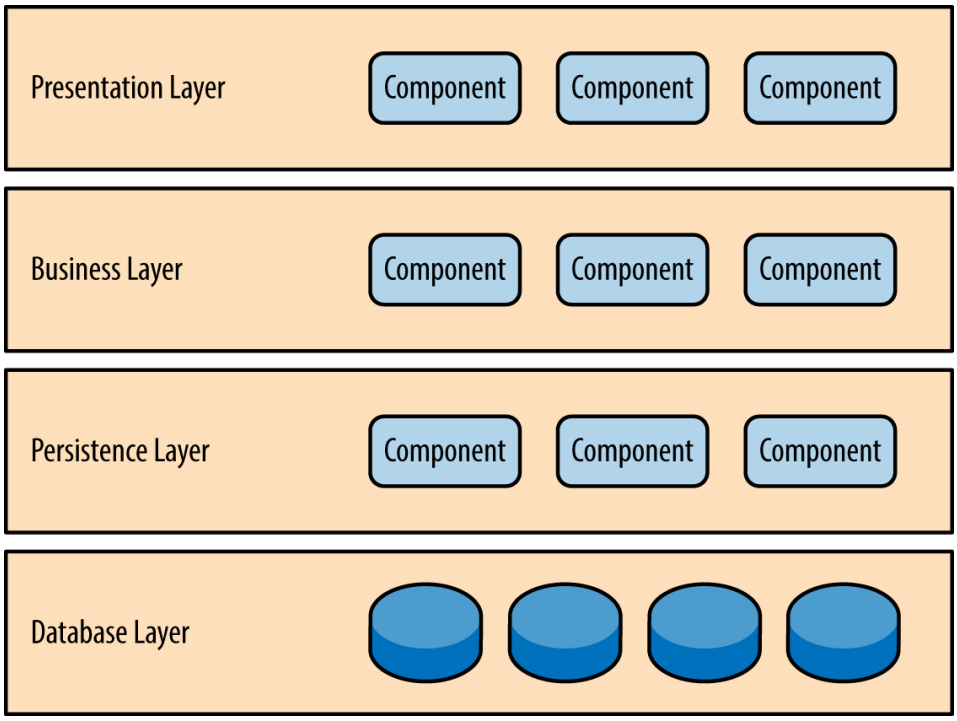


Figura 6. Arquitectura de software por capas

Fuente: https://d3ansictanv2wj.cloudfront.net/sapr_0101-df339e516c077c33fae16623c8ec80c1.png (9 de Abril de 2017)

2.4.5 Patrón modelo, vista, controlador en sistemas cliente

El patrón de diseño 'modelo, vista, controlador' (Richards, Apple, 2017), define 3 tipos de objetos y la forma en la que se comunican entre sí, dentro de una aplicación que tiene interacción con los usuarios, a cada uno de estos objetos se le puede referir también como capa o nivel. Es mayormente usado en sistemas que requieren interacción con usuarios mediante una interfaz gráfica. Este patrón, permitirá a los programadores definir una mejor estructura del código y lograr una mayor reutilización del mismo. Además de modelar efectivamente las vistas a desplegar a los usuarios y la información de las mismas. A continuación, se explica con mayor detalle cada una de las capas de este patrón de diseño:

- **Modelo.-** encapsula la información específica de una aplicación y define la lógica para manipular dicha información. No tienen una conexión explicita con las vistas, sino que lo hacen mediante los controladores.
- **Controlador.-** funciona como un intermediario entre las vistas de y los modelos de la aplicación, realizan las tareas de coordinación de la aplicación y el manejo del ciclo de vida de los objetos. Interpreta las acciones del usuario y actualiza las vistas cuando los modelos cambian.

- **Vista.-** Es un objeto que los usuarios pueden ver y que puede responder a las acciones de los usuarios. Su principal objetivo es desplegar la información de los modelos. La comunicación con los modelos la hace a través de los controladores.

La estructura en general de la arquitectura Modelo Vista Controlador, se representa en la figura 7.

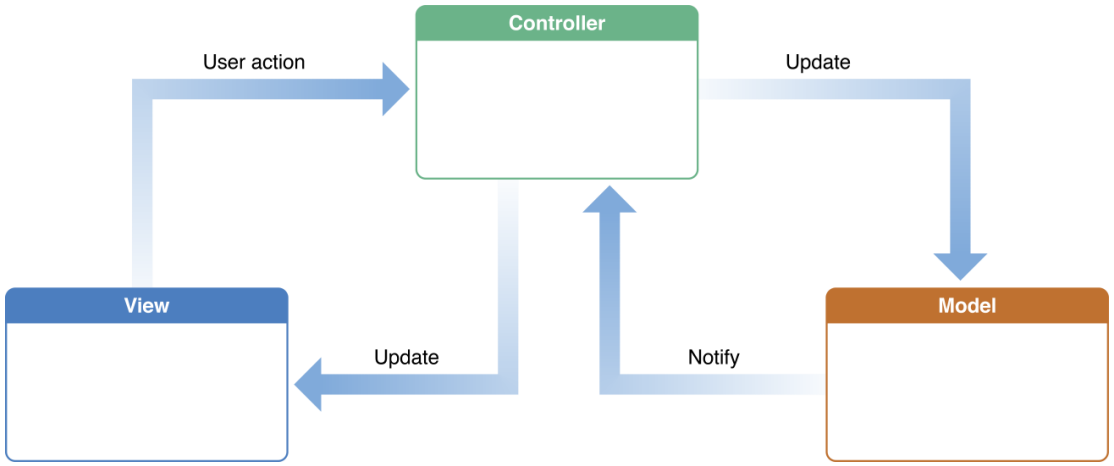


Figura 7. Arquitectura modelo vista controlador

Fuente:
https://developer.apple.com/library/content/documentation/general/conceptual/devpedia-cocoacore/art/model_view_controller_2x.png (9 de Abril de 2017)

2.4.6 DMZ (zona desmilitarizada)

El concepto de redes: DMZ o Zona desmilitarizada (Demilitarized Zone, en inglés) se refiere a una zona de la red de una organización que se sitúa entre la red interna y la red externa o internet y que tiene la función de, precisamente, realizar la comunicación entre estas dos redes. Este concepto es muy importante en la arquitectura de aplicaciones de las organizaciones, ya que la mayoría de las aplicaciones están expuestas a Internet, debido a la interacción con sus usuarios.

El efectivo despliegue de aplicaciones en esta zona de la red, requiere de configuración a nivel de puertos, firewalls y proxies, conceptos específicos de Ingeniería en Redes. (Muycomputer, 2017)

La interacción del área de red DMZ con Internet y la red interna de una organización, se ve representada en la figura 8.

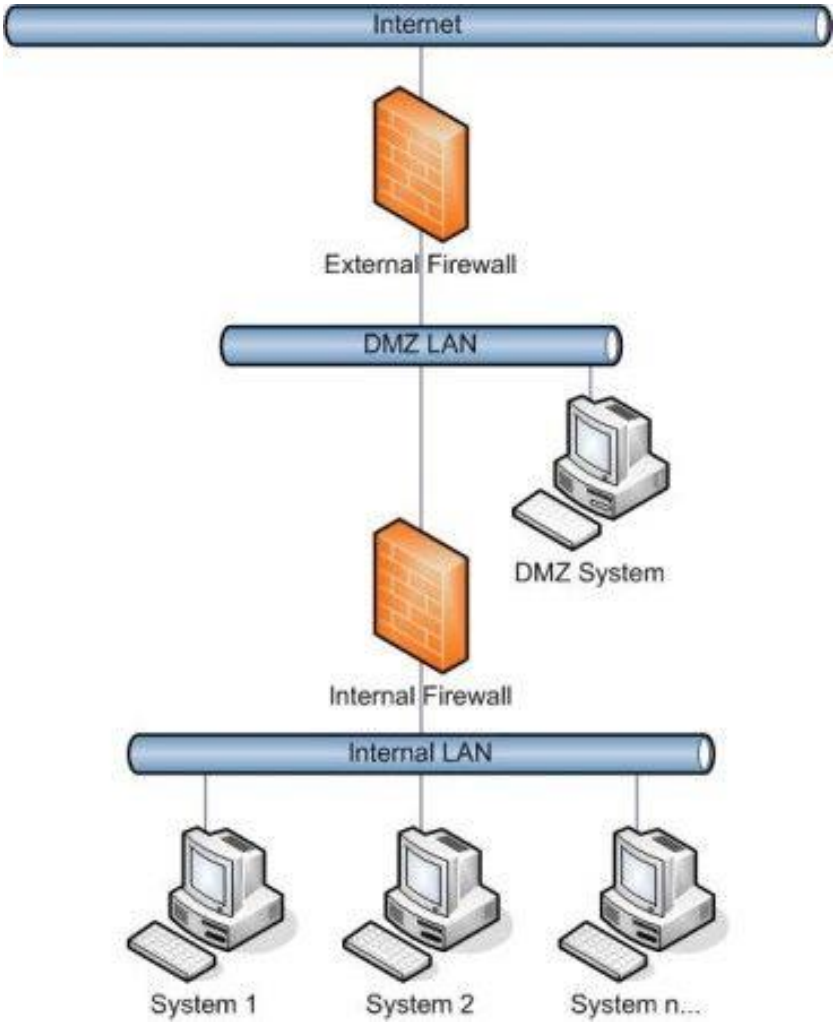


Figura 8. Diagrama de la DMZ en la red interna de una organización

Fuente: http://www.muycomputer.com/wp-content/uploads/2014/01/dmz1-367x450.jpg (9 de Abril de 2017)

2.4.7 REST API (servicios RESTful)

Un REST API, es un servicio basado en la arquitectura REST (Representational State Transfer), que permite la comunicación de servicios para realizar operaciones, mediante el protocolo HTTP. Y en la definición de API (Application Programming Interface) para exponer en sus URIs el acceso a los métodos de consulta y modificación de información para los sistemas cliente de una forma conveniente para su consumo programático. (Hannan & Wilson, 2017)

Un REST API, permite dividir cada una de las transacciones requeridas por el negocio en una serie de pequeños módulos, lo cual hace de este tipo de servicios, sistemas muy eficientes y flexibles tanto para su construcción como para su consumo. (Kearn, 2017)

Algunas de las ventajas que tiene la exposición de métodos REST frente otras formas de comunicación de servicios son:

- Utiliza el protocolo de comunicación estándar de Internet: HTTP (y HTTPS)
- Permite el intercambio de información en formatos: xml, json, txt, html, etc
- El servidor no requiere conocer el estado del cliente para darle la información requerida.
- Las URIs son agnósticas y explorables.
- Utiliza los verbos HTTP para la ejecución de sus operaciones.
- Eficiente el uso de ancho de banda, al evitar la transmisión excesiva de metadatos.
- Son de fácil acceso por cualquier tipo de cliente (móvil, desktop, web, etc).
- El consumo de operaciones individuales mediante URIs, los hace muy flexibles y eficientes.
- Al ser modulares, son flexibles para el mantenimiento y la modificación.

La arquitectura en su forma más simple de un REST API, se ve representada en la figura 9.

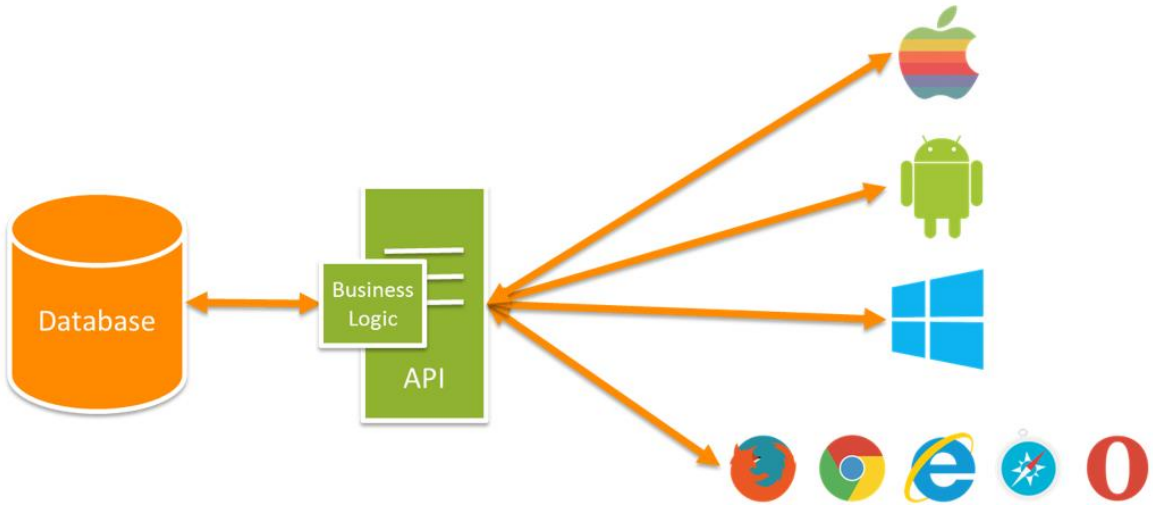


Figura 9. Arquitectura de un API REST

Fuente: https://blogs.msdn.microsoft.com/martinkearn/2015/01/05/introduction-to-rest-and-net-webapi/ (9 de Abril de 2017)

En Java particularmente, la implementación de REST APIs se vuelve es relativamente sencillo con el marco de referencia para la construcción de aplicaciones Web: Spring Web MVC (Spring REST), (Orenstein, 2017) , ya que permite implementar seguridad con autenticación, control y validación de peticiones, optimización de transacciones y envío de respuestas en formato Json, uno de los estándares en transferencia de información más usados a nivel mundial.

2.5 Bases de datos

Para cualquier sistema o aplicación, las bases de datos son un elemento fundamental para la persistencia de información. En la actualidad, las bases de datos mayormente utilizadas en ambientes corporativos son las bases de datos relacionales o bases de datos basadas en SQL (tecnología que se explica en el siguiente punto). Las cuales permiten, a partir de sus capacidades relacionales, modelar y gestionar toda la información de uno o varios procesos de negocio.

2.5.1 Bases de datos relacionales

Una base de datos relacional es una colección de datos categorizados, organizados y relacionados entre sí. Cada tabla tiene registros, los cuales almacenan instancias únicas de información. Adicionalmente, cuenta con metadatos, que describen los renglones y las tablas. (Rouse, Techtarget, 2017)

El acceso, modificación y consulta de la información en las bases de datos relacionales, se hace a través del lenguaje SQL que son las siglas de (Structured Query Language). SQL tiene diferentes funciones dentro de la base de datos y para ello cuenta con 3 sublenguajes, cada uno con una función particular y con sentencias particulares (Orafaq, 2017).

- **DDL (Data Definition Language).-** se utiliza para definir la estructura y el esquema de las bases de datos relacionales.
- **DML (Data Manipulation Language).-** se utiliza para manipular los datos dentro de una base de datos relacional.
- **DCL (Data Control Language).-** se utiliza para controlar los accesos y, roles y administración de las base de datos relacionales.
- **TCL (Transaction Control Language).-** se utiliza para controlar los cambios realizados por las sentencias DML, es decir confirmar, guardar o revertir.

2.5.2 DBMS y RDBMS (Manejadores de base de datos)

Un DBMS (Data Base Management System) es un sistema que permite crear y administrar bases de datos, haciendo posibles las operaciones de inserción, actualización, lectura y borrado de la información (Mullins & Christiansen, 2017). Este sistema es responsable de asegurar los cuatro atributos primarios de las transacciones en cada operación, a los cuales se les conoce como ACID que es el acrónimo en inglés de: Atomicity, Consistency, Isolation, and Durability o en español: atomicidad, consistencia, aislamiento y durabilidad, las cuales se describen a continuación:

- **Atomicidad.-** En una transacción que involucra dos o más piezas de información, se deben de confirmar ambas, de lo contrario, ninguna de ellas lo hará.
- **Consistencia.-** En una transacción que modifica el estado de la información y ocurre un error, la información regresará a su estado anterior.
- **Aislamiento.-** Una transacción en proceso que aún no es confirmada, debe permanecer aislada de todas las demás transacciones.
- **Durabilidad.-** La información confirmada es guardada por el sistema. Aún después de una falla o un reinicio, la información debe seguir disponible y en su estado correcto.

Cabe mencionar que estos atributos, fueron especificados bajo las normas ISO 10026-1. Finalmente, del concepto de manejador de base de datos, hereda el concepto de: manejador de base de datos relacionales. Que en esencia hereda los conceptos del DBMS pero aplicados a un ambiente de base de datos relacional (Elias, 2017).

2.5.2 Data warehouse (almacén de datos)

Una data warehouse o almacén de datos, es un repositorio de toda la información de una empresa, abarcando diferentes sistemas de colecciones de negocio. Se enfoca en la captura de datos de diferentes fuentes y regularmente se construye incrementalmente. El data warehouse no es solo una base de datos corporativa que integra y depura la información obtenida de una o más fuentes distintas. De hecho, el verdadero objetivo es procesar dicha información, permitiendo su análisis desde infinidad de perspectivas y con grandes velocidades de respuesta (Rouse, Techtarget, 2017).

Las principales características de un almacén de datos de acuerdo con (Fernández, 2017) son:

- **Integrado.-** los datos almacenados en el data warehouse deben integrarse en una estructura consistente, por lo que las inconsistencias existentes entre los diversos sistemas operacionales deben ser eliminadas. La información suele estructurarse también en distintos niveles de detalle para adecuarse a las distintas necesidades de los usuarios.
- **Temático.-** sólo los datos necesarios para el proceso de generación del conocimiento del negocio se integran desde el entorno operacional. Los datos se organizan por temas para facilitar su acceso y entendimiento por parte de los usuarios finales. Por ejemplo, todos los datos sobre clientes pueden ser consolidados en una única tabla del data warehouse. De esta forma, las peticiones de información sobre clientes serán más fáciles de responder dado que toda la información reside en el mismo lugar.
- **Histórico.-** el tiempo es parte implícita de la información contenida en un data warehouse. En los sistemas operacionales, los datos siempre reflejan el estado de la actividad del negocio en el momento presente. Por el contrario, la información almacenada en el data warehouse sirve, entre otras cosas, para realizar análisis de tendencias. Por lo tanto, el data warehouse se carga con los distintos valores que toma una variable en el tiempo para permitir comparaciones.
- **No volátil:** el almacén de información de un data warehouse existe para ser leído, pero no modificado. La información es por tanto permanente, significando la actualización del data warehouse la incorporación de los últimos valores que tomaron las distintas variables contenidas en él sin ningún tipo de acción sobre lo que ya existía.

2.5.3 Procesos ETL (Extract, Transform, Load)

Los procesos ETL o en español: extracción, transformación y carga, apoyan a la centralización y categorización de la información tanto en bases de datos tradicionales como en almacenes de datos, permiten la transferencia de información entre bases o repositorios de datos (Sinnexus, 2017).

- **Extracción.-** obtención de información de las distintas fuentes tanto internas como externas.
- **Transformación.-** filtrado, limpieza, depuración, homogeneización y agrupación de la información.
- **Carga.-** organización y actualización de los datos y los metadatos en la base de datos.

La figura 10, representa de forma general la arquitectura más común para un Data Warehouse o Almacen de Datos.

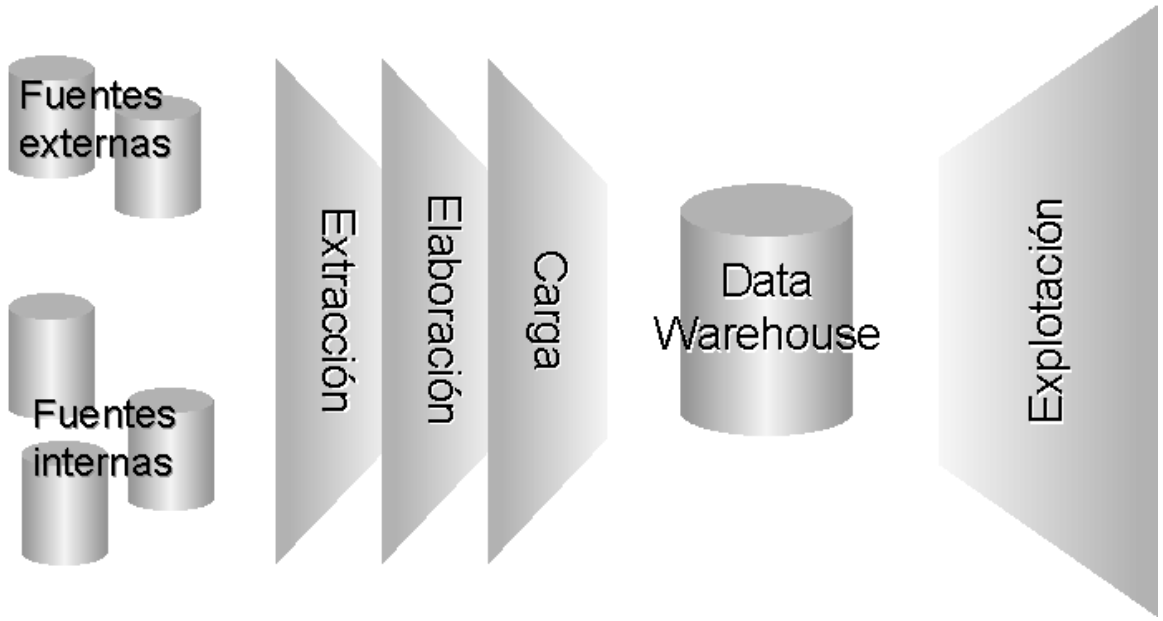


Figura 10. Arquitectura de un data warehouse

Fuente: http://www.dataprix.com/files/manualdwh/que_es_un_data_warehouse_archivos/image87.gif.pagespeed.ce.il0h7_6xye.gif (9 de Abril de 2017)

Cabe mencionar que estos procesos son independientes de cualquier lenguaje o base de datos, de hecho podrían poblar bases de datos de sql a no sql

y viceversa, y pueden ser construidos en cualquier lenguaje que resulte conveniente para el proceso en sí.

2.5.4 ORM (Object Relational Mapping)

El mapeo de objetos relacionales, es un modelo de programación que consiste en la transformación de las tablas de base de datos, en una serie de entidades de programación que facilitan la lectura y manipulación de la información obtenida, así como la ejecución de operaciones transaccionales dentro de la base de datos. (Tuprogramacion, 2017).

Adicionalmente, este modelo de programación, nos permitirá manejar de una forma mucho más eficiente las relaciones entre tablas. Dentro del mundo de Java, Hibernate es uno de los proveedores de ORM, más utilizado. Hibernate, está basado en la especificación de Java Empresarial para acceso a datos: JPA o Java Persistence API (Hbernate, 2017).

Capítulo 3. Necesidades del negocio y requerimientos del proyecto

3.1 Contexto de negocio

Como se menciona en el capítulo 1, la empresa es cuenta con áreas tecnológicas enfocadas en atender a las áreas administrativas y de negocio en cuanto a requerimientos, en orden de, precisamente, favorecer los objetivos de negocio de la empresa. En este capítulo se detalla el contexto de las necesidades que dieron origen a los requerimientos para el desarrollo de la solución de software.

3.2 Necesidades de negocio

De acuerdo a la estructura que se tiene en la empresa, la administración del personal de ventas es responsabilidad de las líderes y gerentes asignadas a cada región y área de distribución de la empresa, por lo que requieren de todo el apoyo posible de la empresa en cuanto a métricas e indicadores en tiempo real para la toma de decisiones y la correcta gestión de su personal, de acuerdo al progreso en cada campaña y los objetivos trazados para sus equipos.

Antes de la solución descrita, la información que se entregaba a las líderes y gerentes, carecía de comparativas con los objetivos de venta trazados para ellas, tenía un retraso de entrega muy significativo. Adicionalmente, estaba consolidado ni centralizado. Por otro lado, en ocasiones, el personal de ventas asignado a una sola líder era tan numeroso que la gestión resultaba altamente costosa en cuanto a tiempo y esfuerzo. No obstante, dada esta mala organización del reporteo, para las líderes y gerentes era muy difícil comparar sus avances campales contra sus objetivos.

Derivado de esta problemática, surgió la iniciativa de negocio de mejorar estos procesos en orden de ayudar a sus líderes y gerentes en este sentido, para favorecer la gestión de sus equipos y con ello propiciar un aumento en las ventas.

3.3 Objetivos de negocio

Dada la problemática que enfrentaban los procesos de negocio de ventas en cuanto a reporteo de información a nivel liderazgo y gerencial, se decidió comenzar un proyecto tecnológico que diera solución a la centralización de la información, la entrega y disponibilidad de información en tiempo real, la facilitación de los datos de contacto de las representantes asignadas a una líder o gerente y la presentación de la información de forma comparativa a los objetivos de negocio.

Adicionalmente, este proyecto se pensó desde su inicio con un enfoque multinacional, ya que los mercados de centro América y sud América presentaban problemas muy similares a México. Si el proyecto daba resultado en el país podría ayudar a los procesos de reporteo en otros países, también. Por lo que durante el diseño se tomaron en cuenta estas consideraciones.

3.4 Retroalimentación del equipo de desarrollo

El equipo de negocio presentó la iniciativa al equipo de desarrollo en una primera instancia, para obtener una retroalimentación sobre la posible solución a las problemáticas actuales dentro de la línea de negocio de reportes a gerentes de ventas.

Cabe mencionar que por gracias a las observaciones realizadas por mi parte, se logró tener una visibilidad mucho más amplia de las consideraciones a realizar, ya que contaba con una experiencia técnica amplia en el tema, además de un fuerte entendimiento de los procesos de negocio.

Después de una evaluación detallada, el equipo de desarrollo presentó los puntos destacados para considerar dentro los requerimientos, de acuerdo a las capacidades y alcance tecnológico de las herramientas y el equipo, mismos que se presentan a continuación:

- La centralización de los reportes requeridos por líderes y gerentes de ventas, requiere de la recepción diaria y automatizada de información desde el data warehouse a una base de datos de aplicación enfocada a esta solución.
- La comparación visual de los avances contra los objetivos de ventas, requiere de una interfaz gráfica dinámica que favorezca visualmente estos indicadores.
- La gestión de los miembros del equipo de ventas, debe centralizarse también y permitir el envío de mensajes de forma inmediata al identificar áreas de oportunidad con base a las métricas presentadas.
- Las líderes y gerentes deben ser capases de consultar el detalle de ventas del personal asignado a ellas.
- El sistema debe estar expuesto a Internet para permitir la consulta de las líderes y gerentes, fuera de la red interna de la empresa.
- La medición de la frecuencia y los reportes que se consultan es de suma importancia para la toma de decisiones, posteriores a la liberación del sistema.

- Existe información que se genera de los procesos de factura, la cual es más conveniente presentar en archivos que en una interfaz gráfica.

A partir de estas recomendaciones el equipo de negocio, obtuvo visibilidad de lo que era posible construir y de lo requerido para ello, mediante lo cual formulo los requerimientos para el equipo de desarrollo.

3.5 Requerimientos

Después de un proceso de revisión y aprobación, el área de negocio formuló sus requerimientos para el equipo de desarrollo:

3.5.1 Requerimientos generales (funcionales)

- Construcción de un sistema web para la centralización de la información de reportes y métricas de venta a nivel liderazgo y gerencial.
- Exponer el sistema construido a Internet.
- Administrar la seguridad y autenticación para el acceso al sistema (Acceso solo permitido a líderes y gerentes de ventas activas).
- Implementar sistema de bitácora, métricas y analíticas de acuerdo a las consultas recibidas en el sistema.
- Personalización de reportes, para permitir a líderes y gerentes consultar el detalle de ventas del personal a su cargo.
- Envío de mensajes a personal de ventas por líderes y gerentes al personal a su cargo, dentro de la misma aplicación.
- Presentar la información en archivos, cuando sea más conveniente de acuerdo a la naturaleza del reporte.
- Presentar la información de reportes y métricas de forma óptima (visualmente).
- La aplicación web, deberá mostrar la información con periodicidad de actualización de día vencido.

Es importante destacar que, dentro de mis labores como desarrollador también se incluyó la traducción de requerimientos a nivel de negocio a requerimientos a nivel técnico, facilitando el entendimiento de las necesidades el área de negocio hacia la programación.

3.5.2 Requerimientos específicos

- Construcción de un tablero que presente los indicadores generales de ventas en comparación con los objetivos trazados a alcanzar.
- Construcción de reportes en forma de tabla específicos por cada línea del proceso de ventas (ventas, cambios, devoluciones, nombramientos, bajas, etc).
- Los reportes presentados deberán de ser capaces de mostrar información de la campaña actual y de 4 campañas anteriores.
- El sistema mostrará la última fecha de actualización de la información que presenta.
- La interfaz gráfica deberá de ser responsiva a cualquier tamaño de pantalla.
- Los reportes en forma de tabla presentados, tendrán la capacidad de filtrar la y ordenar la información por columna.
- Los reportes en forma de tabla presentados, se podrán exportar a archivos de Excel.
- El sistema no mostrará información de representantes que no correspondan a la líder o gerente que ingrese al sistema.
- El tiempo de actualización de la información será de 24 horas.

3.5.3 Requerimientos no funcionales

- La solución, deberá de mantenerse disponible para las líderes y gerentes de ventas, las 24 horas del día, los 7 días del año, en Internet.
- La solución deberá soportar el total de usuarios de la aplicación (total de líderes y gerentes de ventas) simultáneamente.
- La aplicación deberá de desplegar la información de forma rápida al usuario final.
- La solución deberá contar con respaldos de base de datos para reaccionar ante emergencias.
- La solución deberá de contar con métodos de alerta hacia las áreas de soporte, cuando sucedan errores.
- La solución deberá de manejar la seguridad de la información (mediante autenticación de usuarios, manejo de sesiones, protección de urls y bloqueo de potenciales ataques web).

Capítulo 4. Arquitectura y detalle técnico del desarrollo

4.1 Generalidades técnicas, dentro de la organización

La empresa cuenta con la implementación de tecnologías de desarrollo de muy alto nivel, para dar solución a los requerimientos del negocio de forma rápida y eficiente. Teniendo licencias de las mejores bases de datos, sistemas operativos, herramientas de comunicación y servidores aplicativos, en cuanto a rendimiento y desempeño.

Además, cuenta con una fuerte estructura técnica y organizacional que permite dar soporte a problemas, incidentes, cambios y liberaciones, en orden de enfocar los esfuerzos de los recursos de desarrollo, plenamente a la construcción de soluciones.

Es de suma importancia resaltar que debido a que participé desde el inicio del proyecto, tuve la oportunidad de colaborar con el diseño de la arquitectura de esta solución, en la parte de servicios Rest y Base de Datos, el uso de Javascript en el navegador y la administración de la herramienta de gestión de versiones: Git.

4.1.1 Herramientas para el desarrollo de software en la empresa

Con el objetivo de favorecer la compatibilidad, comunicación y reutilización de aplicaciones, se cuenta con un estándar a nivel mundial de las herramientas utilizadas para el desarrollo de software en la empresa. Como parte de las tareas del análisis previo al desarrollo, participé en la elección de los sistemas operativos más convenientes para la solución.

Primeramente, se cuenta con servidores Linux (Red Hat) y Windows Server virtualizados de servidores físicos que se encuentran principalmente en la sede de la empresa en Nueva York, Estados Unidos, los cuales abarcan 3 ambientes de programación en orden de favorecer la agilidad de entrega de las aplicaciones:

- **Desarrollo.-** se utiliza para probar cambios de la forma más conveniente que al equipo de programación le parezca en orden de asegurar la funcionalidad del software de forma unitaria.
- **Pruebas.-** este ambiente se destina a realizar todo tipo de ejecución de casos necesarios para el aseguramiento del funcionamiento integral, de rendimiento y de seguridad. Además, en este ambiente se llevan a cabo la

ejecución de los casos de aseguramiento de funcionalidad que el dueño de la aplicación, del lado de negocio, solicitó.

- **Producción.-** este ambiente es exclusivo para los usuarios finales, es decir, clientes de la empresa o usuarios internos de la empresa con transacciones y operaciones reales de negocio, la seguridad en este ambiente es mucho más estricta, precisamente para asegurar el buen funcionamiento, la integridad de la información y la disponibilidad de las aplicaciones. Es el ambiente con más capacidades de procesamiento, memoria y rendimiento, en ocasiones está expuesto a Internet.

La secuencia con la que se distribuyen las aplicaciones a estos diferentes ambientes se muestra a continuación, nótese que los servidores físicos productivos, son independientes de los servidores de desarrollo y pruebas por motivos de seguridad. La disposición de los ambientes de ejecución de aplicaciones a nivel de servidores, se muestra en la figura 11.

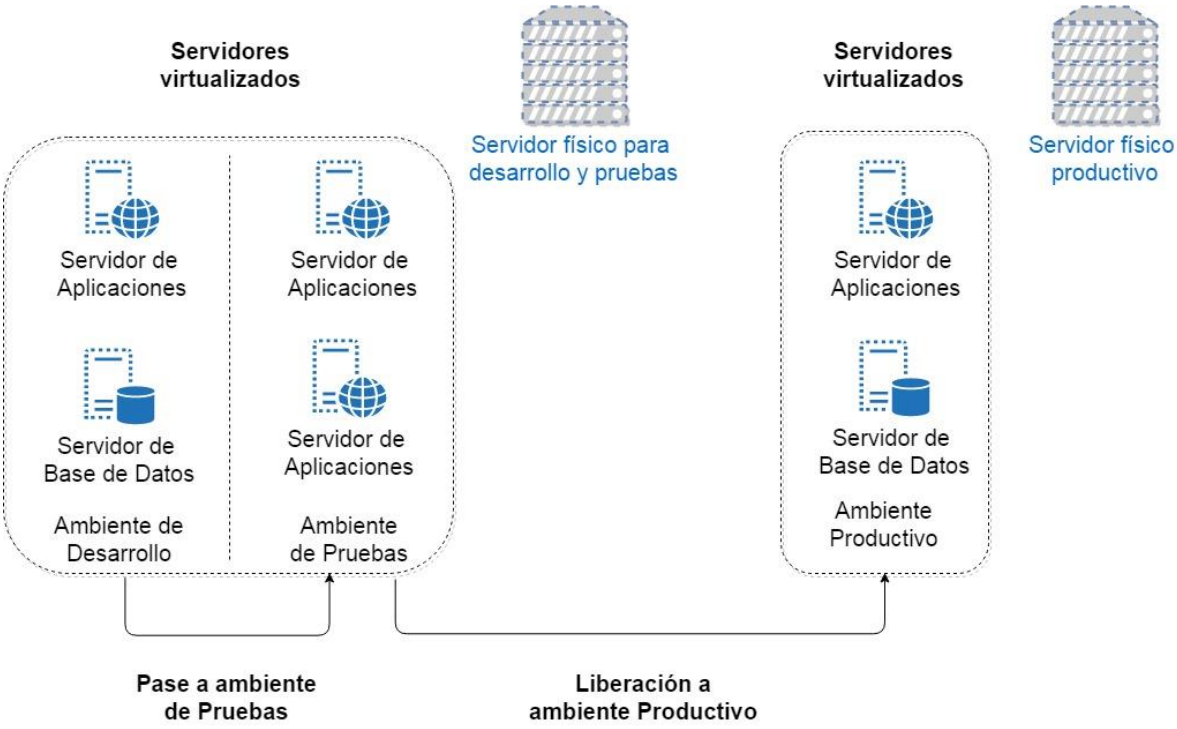


Figura 11. Ambiente de aplicaciones, del área de TI en México

Fuente: Elaboración propia

Como se puede apreciar en la imagen anterior, el pase a cada uno de los ambientes, depende del resultado de las pruebas realizadas en el ambiente anterior,

para asegurar la calidad de los servicios que se exponen, en los capítulos siguientes se detallará el proceso que puntualmente exige ITIL para el pase a cada uno de estos ambientes.

En cuanto al control de versiones del código, se cuenta con servidores SVN y Git, que son mantenidos, directamente por el área de integración e implementaciones, de los cuales cada equipo de desarrollo es responsable de la correcta administración para el aseguramiento de la persistencia del código fuente de sus aplicaciones. En cuanto a Git se cuenta con la licencia y el soporte de Gitlab Empresarial, por lo que los proyectos más grandes e importantes se llevan en esta herramienta.

En cuanto a las aplicaciones web que se despliegan a servidores Web, se utiliza el Servidor de aplicaciones WebSphere de IBM, el cual permite un manejo de sesiones mucho más eficiente y controlado, así como una conexión a bases de datos más segura y confiable.

Por último, las bases de datos que se utilizan tanto para ambientes de desarrollo y pruebas como para ambientes productivos son bases relacionales SQL de Oracle, que es uno de los motores de bases de datos relacionales más poderosos de la industria y permite guardar gran cantidad de tipos datos, además de una arquitectura que favorece la programación dentro de la base de datos para ejecutar ciertos segmentos de la lógica de negocio de la aplicación o de los procesos de carga y extracción.

4.1.2 Sistemas multinacionales

Como se mencionó anteriormente, la empresa tiene presencia en diferentes países a nivel mundial, por ello, los sistemas que se construyen en la empresa buscan tener las capacidades a nivel base de datos, servicios y capa de presentación de atender a diferentes países simultáneamente. Estas capacidades requieren de ciertas configuraciones que deben son consideradas y definidas durante el periodo de definición, antes de comenzar cualquier tipo de desarrollo.

Como se mencionó anteriormente, este tipo de sistemas se les llaman: sistemas multinacionales y la empresa ha llevado tan lejos este concepto que la aplicación móvil para todos los países en los que la empresa tiene presencia se consolidó como un solo desarrollo para todo el mundo, el cual permite al usuario

final elegir su país, con lo que accederá a una capa de servicios en particular, propia de su país, desplegando vistas y funciones particulares.

Otro ejemplo de este tipo de arquitecturas, es el sitio Web para las representantes de ventas, la cual se conforma por un desarrollo en común en cuanto a las vistas y los controladores para América Latina y América del Norte, direccionando desde la capa de controladores a los servicios y bases de datos del país desde el que se esté accediendo.

Cabe mencionar que el diseño de un sistema multinacional requiere de identificación y manejo de roles y sesiones en la aplicación MVC, los servicios Rest y la base de datos, diseño en el cual tuve la oportunidad de participar.

4.1.3 Soporte técnico global centralizado

Algo muy importante a destacar del soporte hacia los usuarios tecnológicos en la empresa, es la centralización de la información y la velocidad de respuestas que se brinda. La empresa cuenta con el soporte de toda su infraestructura tecnológica en 2 niveles, que se ofrece a los usuarios de base de datos, sistemas operativos, servidores web o repositorios de código y que consiste en brindar accesos, administrar la configuración, gestionar las capacidades, y gestionar la seguridad e integridad de los ambientes:

- **Nivel local.-** se refiere al soporte que se brinda en cada uno de los países a los usuarios de recursos para desplegar e implementar aplicaciones locales, es decir aplicaciones que no serán multinacional o que se alinean a una vertical de negocio muy particular del mercado de ese país.
- **Nivel global.-** se refiere al soporte que se brinda desde Estados Unidos de América o Reino Unido a los usuarios de recursos para desplegar aplicaciones multinacional o con un objetivo y un alcance internacional.

En la figura 12 se muestra la dependencia que se tiene de los equipos de desarrollo de software hacia los equipos de soporte para ilustrar mejor la organización en esta área de la empresa:

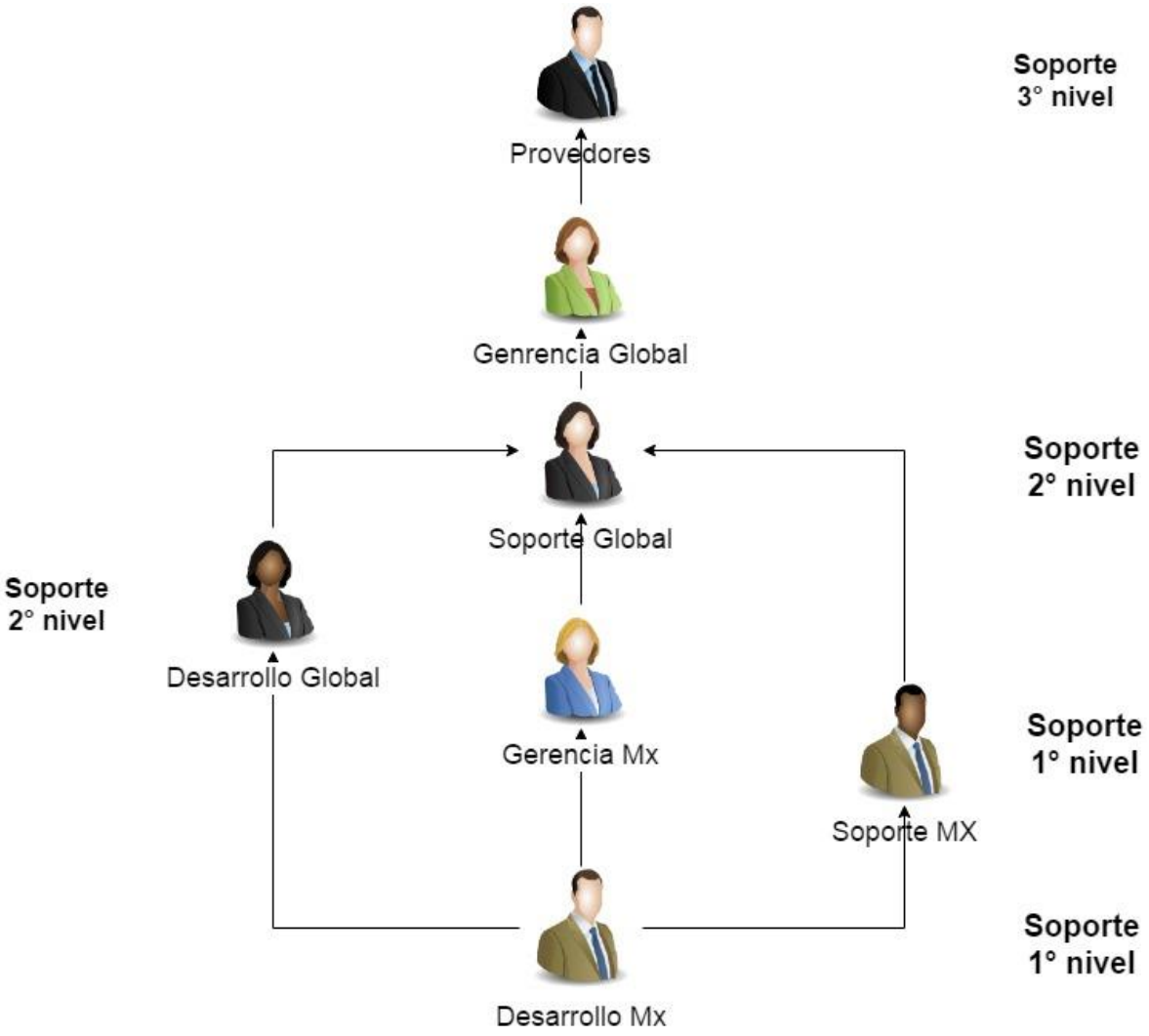


Figura 12. Soporte centralizado dentro de la empresa

Fuente: Elaboración propia

Como se puede apreciar en el diagrama, el soporte a nivel global también atiende directamente a los equipos de desarrollo con alcance multinacional. También, brinda apoyo a los equipos de soporte locales con temas de alta complejidad de ser necesario. Finalmente, los equipos de soporte global deberán recurrir directamente a los equipos de soporte de los proveedores del software licenciado que se usa en la empresa en caso de emergencia o imposibilidad para solucionar temas particulares con alguna herramienta. En ocasiones, los equipos de desarrollo a nivel global, dan soporte a los equipos de desarrollo locales, debido a la herencia de aplicaciones 'multinacional'.

Dentro de estos 2 niveles, también existe la parte de desarrollo, pruebas y producción, mismas que deberán documentarse de manera ligeramente distinta para la liberación en cada uno de estos ambientes.

4.2 Requerimientos de desarrollo de áreas de negocio

Como se mencionó en el capítulo 3, los requerimientos que surgen de las áreas de negocio hacia las áreas tecnológicas y de desarrollo de software deben de contar con un fuerte sustento en cuanto al costo beneficio del proyecto, documentando estos detalles a fondo para presentarlos a nivel dirección. Dichos documentos pasarán por la evaluación y una vez aprobadas, se comprometerá una fecha de entrega por parte del equipo de desarrollo designado, considerando un periodo de pruebas suficiente tanto para las áreas de aseguramiento de calidad como las áreas de usuarios de la aplicación. Este proceso en la empresa está alineado a la especificación de ITIL y se abarca ampliamente en el capítulo 2.

4.2.1 Equipos de desarrollo enfocados a áreas particulares de negocio

Las divisiones de los recursos de desarrollo dentro de la empresa se orientan por líneas de especialidad de negocio, es decir, cada área de negocio tiene un equipo especialista en desarrollo de software, para atender a sus requerimientos de construcción de aplicaciones. Adicionalmente, existen áreas tecnológicas como Desarrollo Cobol en Mainframe o Data warehouse las cuales no responden directamente a ningún área de negocio en particular, sino que se especializan en dicha tecnología para atender a todo el negocio de la empresa en general por temas de procesamiento masivo o almacenamiento masivo de datos.

Al igual que el esquema organizacional jerárquico, a continuación se presentan los equipos de desarrollo correspondientes a cada línea de negocio.

4.2.2 Dependencias con otros equipos de desarrollo, dentro de la organización

Cabe mencionar que el hecho de que a cada vertical de negocio corresponda un equipo de desarrollo, no significa que dicho equipo de desarrolla responda directamente a su áreas correspondientes de negocio, sino que atiende sus requerimientos directamente, mediante la aprobación de su líder o gerente.

La estructura organizacional de la empresa para los equipos de TI en México, involucrados en el proyecto, se puede apreciar en la figura 13.

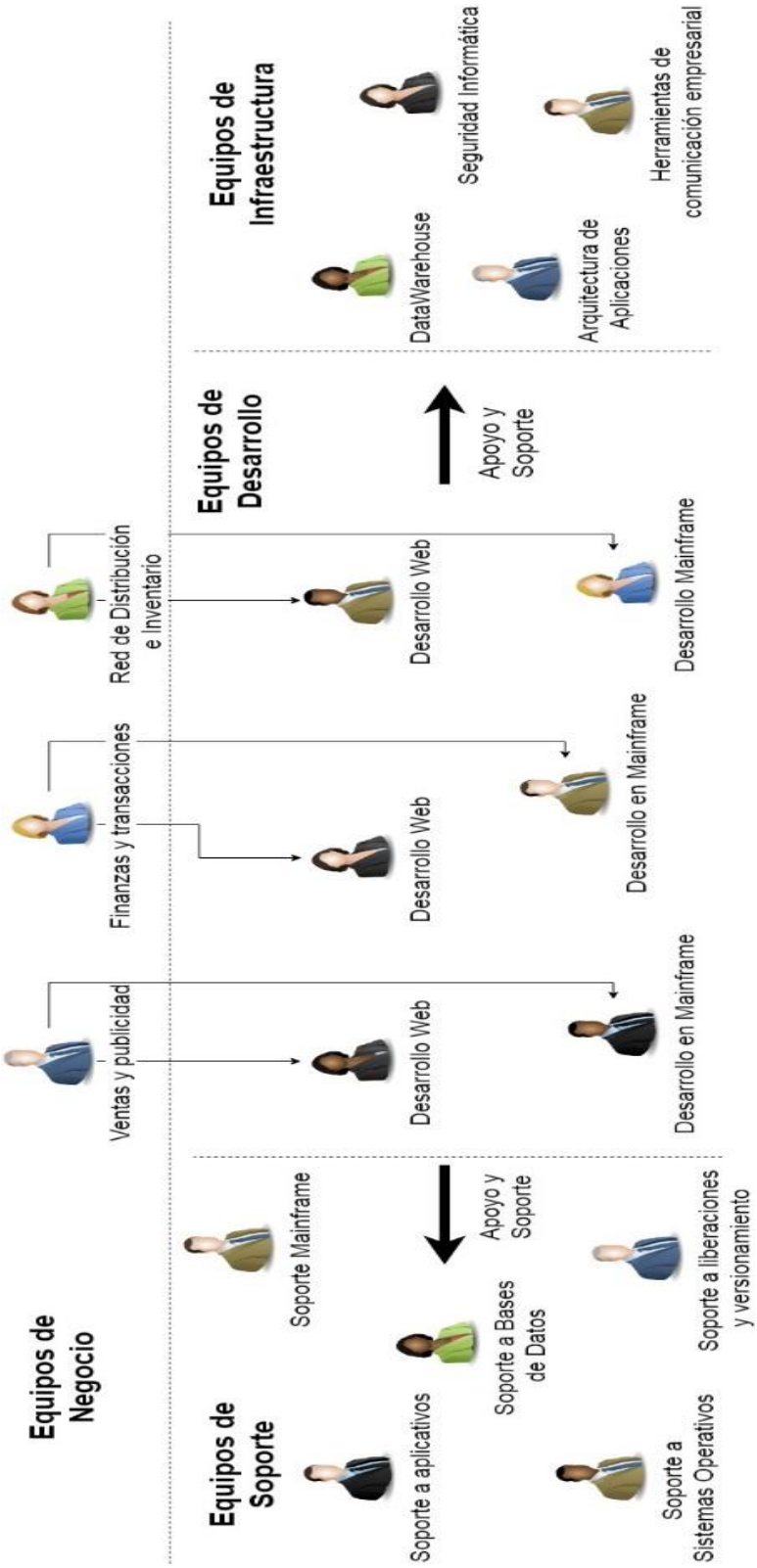


Figura 13. Relación entre equipos de negocio y desarrollo en el área de ti en México

Fuente: Elaboración propia

Como se puede apreciar en la imagen, se cuenta con equipos de desarrollo en general, que apoyarán a todos los equipos de desarrollo específico. A continuación, se detalla el rol de cada uno de estos equipos:

- **Data Warehouse.-** El equipo de almacenamiento masivo de datos, se encarga de extraer toda la información de cada uno de los sistemas y bases de datos de la empresa y consolidarlo en un solo lugar, centralizando la información para su posterior explotación.
- **Seguridad Informática.-** El equipo de seguridad informática, se encarga tanto de la seguridad a nivel de computadoras personales, red interna y externa, así como la seguridad a nivel base de datos y capa de presentación y servicios en cada una de las aplicaciones construidas.
- **Arquitectura de Aplicaciones.-** Este equipo tiene como objetivo el asegurar la calidad del código, realiza pruebas de rendimiento y so de recursos por aplicación, define la arquitectura de las aplicaciones multinacional.
- **Comunicación empresarial.-** Este equipo provee de aplicaciones de comunicación entre sistemas distribuidos, web y sistemas Mainframe, para asegurar las conversaciones seguras y eficientes entre aplicaciones.

4.2.3 Implementaciones globales, pendientes en México

Dentro del ecosistema tecnológico de la empresa, se tienen grandes proyectos de adquisición e implementación de sistemas, herramientas y tecnologías de software para atender a los equipos de desarrollo a nivel global en términos de compilación, manejo de dependencias, construcción, distribución y control de versiones con el objetivo de fomentar su eficacia mediante este tipo de herramientas.

En este sentido, la empresa en México ya goza del uso de ciertas herramientas provistas por la empresa en Estados Unidos, tal es el caso de los repositorios Gitlab para Git o de repositorios Maven con Nexus.

Actualmente se cuenta con herramientas de la empresa en Estados Unidos de Norte América con las que la empresa en México aún no cuenta, tal es el caso de las herramientas de integración continua y automatizada, es decir, distribución de compilados a servidores de pruebas o de producción después de la compilación. En este sentido, falta mucho por seguir impulsando y mejorar dentro de la empresa, ya que estas decisiones dependen de un nivel dirección local y no de los líderes de desarrollo.

Cabe destacar que mis tareas dentro de la empresa, incluían ser parte de las liberaciones a producción, en el sentido de soporte y validación.

4.3 Arquitectura de base de datos

El diseño y modelado de la base de datos de la solución, fue una tarea sumamente importante ya que al ser una aplicación de reporteo, depende completamente de la información que almacene la base de datos, en esta sección se hace una revisión de cómo se realizó el diseño y la construcción de la misma.

Una de las tareas en las que más me vi involucrado, fue precisamente el diseño de la arquitectura y el modelo de la base de datos, tanto en elección de tecnologías como en modelado de datos y métodos de acceso y entrega de información.

4.3.1 Tecnologías utilizadas

De acuerdo con lo establecido en la empresa, las bases de datos de aplicaciones Java deben de ser Oracle. En orden de asegurar una mejor compatibilidad entre el software y la base de datos al ser tecnologías soportadas por Oracle, empresa de la cual se tiene soporte para la configuración, administración y atención a incidentes en su versión 11g.

Para la construcción de la base de datos, además del lenguaje sql para la definición del modelado y programación de la base de datos se usaron tecnologías alrededor de ella, como es el caso de:

- **Java con Hibernate (ORM).-** tecnología que usamos para la extracción de datos desde la programación hacia las bases de datos relacionales.
- **Shell script.-** tecnología que usamos para la invocación de los procedimientos almacenados de forma automatizada y captura de errores en dichos procesos, fuera de la base de datos para un mejor manejo de los mismos.
- **PL.-** (Procedural Language o Lenguaje de Procedimientos), tecnología que usamos para la programación de procedimientos almacenados para cargar información y para realizar limpieza de datos (depuración). Cabe mencionar que es conveniente utilizar PL cuando se pretende manipular gran cantidad de registros en una base de datos, dado que es un lenguaje nativo de cada manejador de Base de Datos.
- **SQL Developer.**- Este ambiente de desarrollo Integrado, lo usamos para la consulta a bases de datos Oracle, ya que contiene edición de código,

navegación de objetos de base de datos y administración de múltiples conexiones, entre otras funcionalidades.

PL, es un lenguaje para las bases de datos relacionales que brinda la ejecución de funcionalidad directamente en la base de datos con elementos como variables, ciclos, estructuras de control y manejo de errores, para un procesamiento de la información por lotes flexible y rápido. Es recomendable usarlo sobre lenguajes externos a la base de datos en casos de procesamiento masivo de información, debido al eficiente manejo de los datos. (Oracle, 2017).

4.3.2 Diseño de base de datos

En cuanto la arquitectura que se diseñó para la base de datos, en orden de cumplir con el flujo de la empresa para el manejo de la información, se diseñaron tres diferentes esquemas por los que la información pasaría para finalmente almacenarse en tablas para consulta productiva:

1. **Tablas Externas.-** mediante la tecnología de Oracle de tablas externalizadas, se puede cargar mediante un archivo en Excel, csv o separado por tildes, información directamente a una tabla de Oracle sin ejecutar ningún comando sql. De este modo se cargaba la información a la base de datos de las interfaces (archivos) recibidas de Data warehouse. Cabe mencionar que en este tipo de tablas no se puede realizar modificaciones a la información, solo sirven para consulta.
2. **Tablas de ensamblado.**- en estas se copia toda la información de las tablas externas, tal cual se encuentra, en orden de tener la información guardada directamente en la base de datos y no en archivos, para poderla manipular.
3. **Tablas productivas.**- estas tablas son las tablas para consulta productiva, mismas que se alimentan de las tablas stagging pasando por un proceso de limpieza en el cual se asegura que los valores a ingresar estén dentro de los rangos esperados, de lo contrario son descartados y reportados. Al mismo tiempo se valida que la información a ingresar de las tablas stagging a las tablas dimention, no exista, de lo contrario se hace una actualización de estos registros, para evitar reingresar la información y perder la consistencia histórica.

La arquitectura de base de datos mostrada, fue diseñada por mí en conjunto con el líder de proyecto del equipo. El diseño de base de datos, implementado para la solución, se representa en la figura 14.

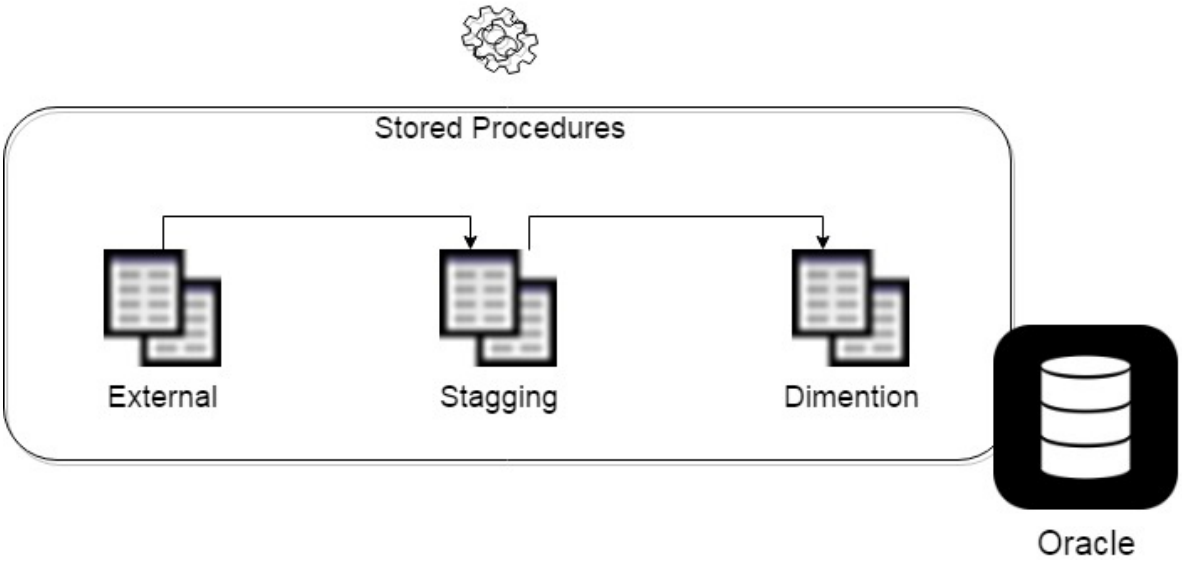


Figura 14. Definición de base de datos

Fuente: Elaboración propia

4.3.3 Modelado de Base de Datos

La base de datos de la solución, cuenta con diferentes tablas operativas y catálogos que guardan una relación entre ellas de acuerdo al negocio que se modeló.

Para el modelo de la base de datos de la solución, se tomó en cuenta el modelo existente para el proceso de negocio de nombramientos web y ordenes en línea del negocio de México (mismo que no es multinacional). Modelo de datos con normalización nivel 3 para evitar redundancia de datos, ya que es una base de datos que atiende a miles de usuarios al día. En la figura 15, se muestra un el diagrama entidad relación de dicho diseño a nivel lógico.

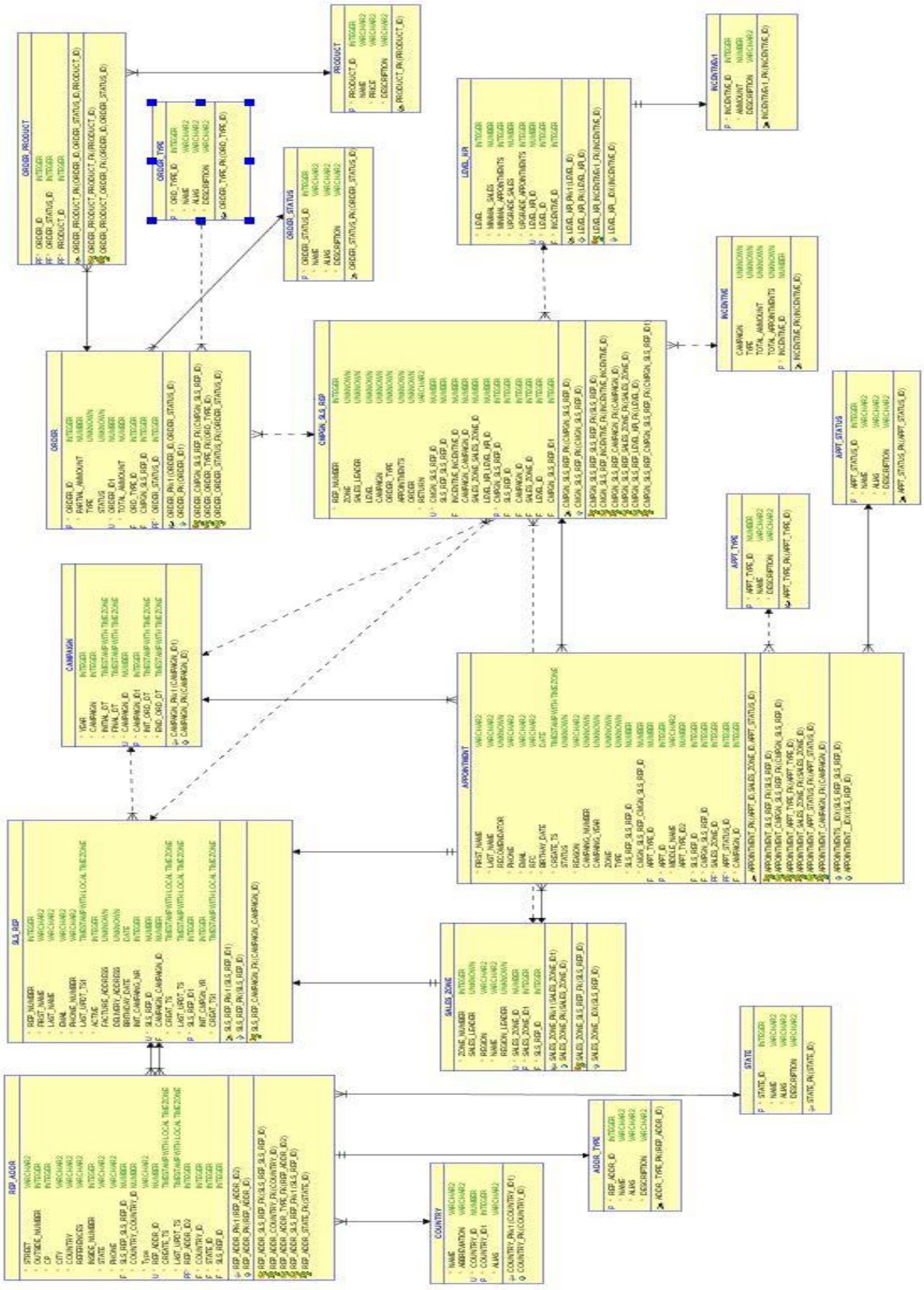


Figura 15. Modelo general de base de datos para la solución (nivel físico)

Este modelo fue diseñado para atender los procesos de negocio de órdenes en línea y nombramientos web en México, el cual es de suma importancia porque representa la entrega de productos y el acceso de nuevas vendedoras(os) a la empresa. Por lo cual el diseño tiene una alta normalización y un estricto modelado de datos.

Este tipo de modelos, resultan difíciles de implementar e ineficientes al replicar para una base de datos que pretende únicamente, desplegar información en una aplicación web de reportes, ya que se requiere migrar todo el modelo da datos para cumplir con las restricciones de llaves primarias y foráneas, además de hacer mucho más compleja la consulta a la información y necesitar mucho mayor espacio de almacenamiento en disco, por lo que se decidió tomar únicamente la información necesaria para cumplir con este requerimiento.

Para lograr importar toda la información necesaria sin caer en inconsistencias de datos, yo y el responsable del proyecto, trabajamos en reducir la normalización del modelo deliberadamente para guardar en una menor cantidad de tablas la misma información. Adicionalmente, trabajamos en preparar el diseño de la base de datos para otorgarle la capacidad de atender a otros países que en su momento requirieran de esta aplicación, mediante un campo de país, el cual se hizo obligatorio en todas las demás tablas.

Este diseño se puede observar en el diagrama entidad relación (a nivel físico) de la figura 16.

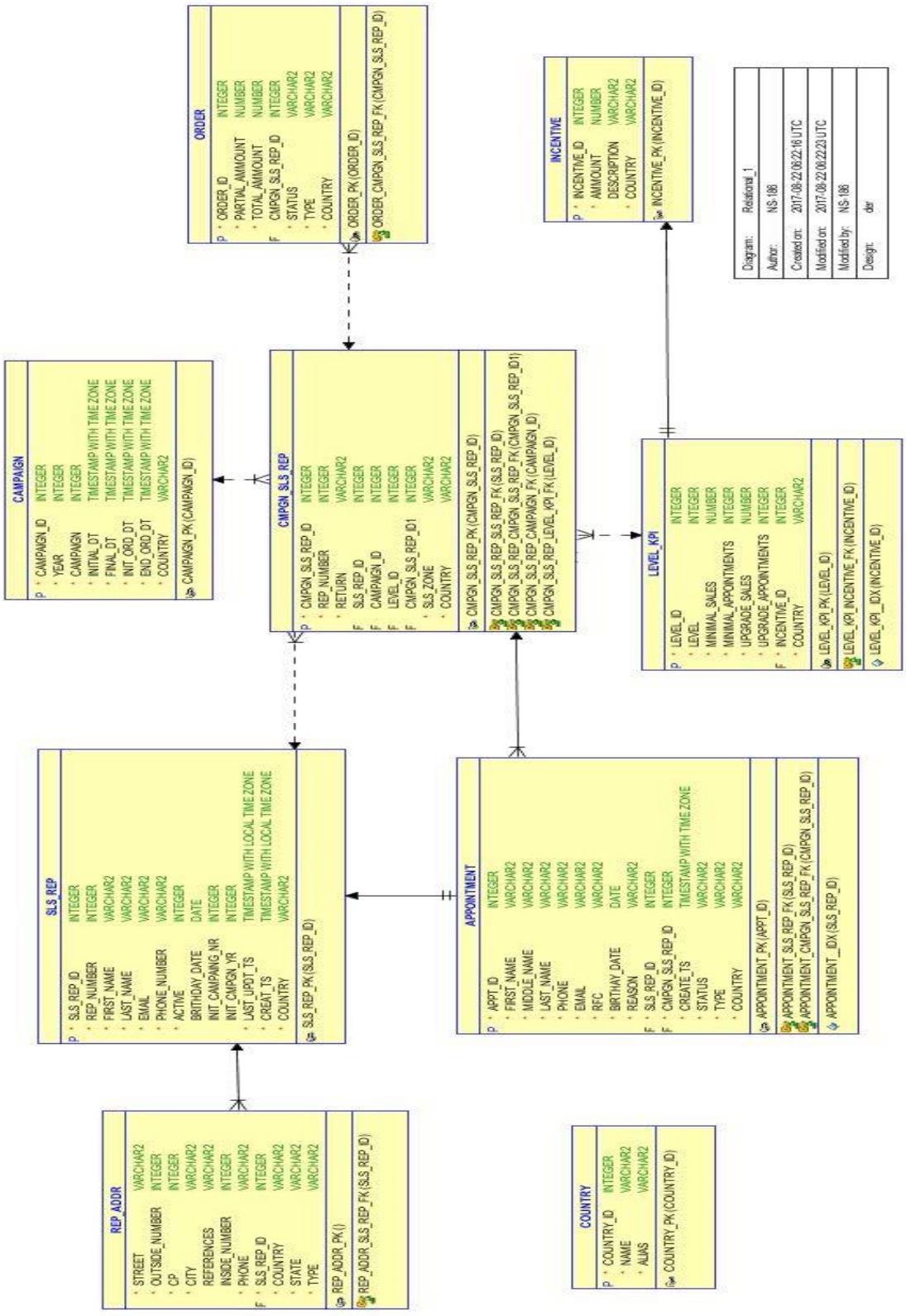


Figura 16. Diagrama entidad relación del modelo de la solución (nivel físico)

A continuación, se describe cada una de las tablas mostradas en el diagrama entidad relación de la solución:

Tablas operativas:

- **SLS_REP.-** En esta tabla se almacena la información de todo el personal de ventas de la organización, tanto a nivel de representantes como de líderes y de gerentes.
- **REP_ADDR.-** En esta tabla se almacenan las direcciones de cada de facturación y de reparto para cada uno de los registros de la tabla de representantes.
- **CMPGN_SLS_REP.-** Esta tabla se organiza por campaña, es decir por cada uno de los periodos de venta. En ella se almacena información sobre las ventas, nombramientos y devoluciones alcanzados por cada representante.
- **ORDER.-** En esta tabla se almacena información del pedido, cambio o devolución de cada representante por campaña.
- **APPOINTMENT.-** En esta tabla se almacena toda la información del proceso de nombramientos en la empresa, es decir la incorporación de eventos a la fuerza de ventas.

Tablas de catálogo:

- **COUNTRY.-** Catalogo en donde se almacena la referencia de cada uno de los países que abarca el mercado de la empresa, permite diferenciar a las representantes de cada país, le da la capacidad al sistema de funcionar para varios países con la misma base de datos.
- **CAMPAIGN.-** En esta tabla, se almacena el catálogo de las fechas de inicio, facturación y fin de cada campaña de la empresa.
- **INCENTIVE.-** En esta tabla se almacena el catálogo de beneficios e incentivos a los cuales las representantes se hacen acreedoras al alcanzar cierto nivel de ventas.
- **LEVEL_KPI.-** En esta tabla se almacenan los niveles de venta dentro de la empresa, de acuerdo a las metas en cuanto a órdenes, monto y nombramientos, por alcanzar.

4.3.4 Dependencias con DataWareHouse

Como se menciona en el capítulo 2, un data warehouse es una base de datos centralizada que consolida, limpia y explota la información de diferentes sistemas de la organización. En el caso de la empresa, la obtención de la mayor parte de la información se hace de los sistemas Mainframe. Sin embargo, también llega información de los sistemas de CRM (Customer

Relationship Manager), Call Center, sistemas facturación, sistemas de distribución, etc.

Para el caso de la empresa, el flujo en general que se tiene para la obtención de la información dentro de la solución descrita se representa en la figura 17.

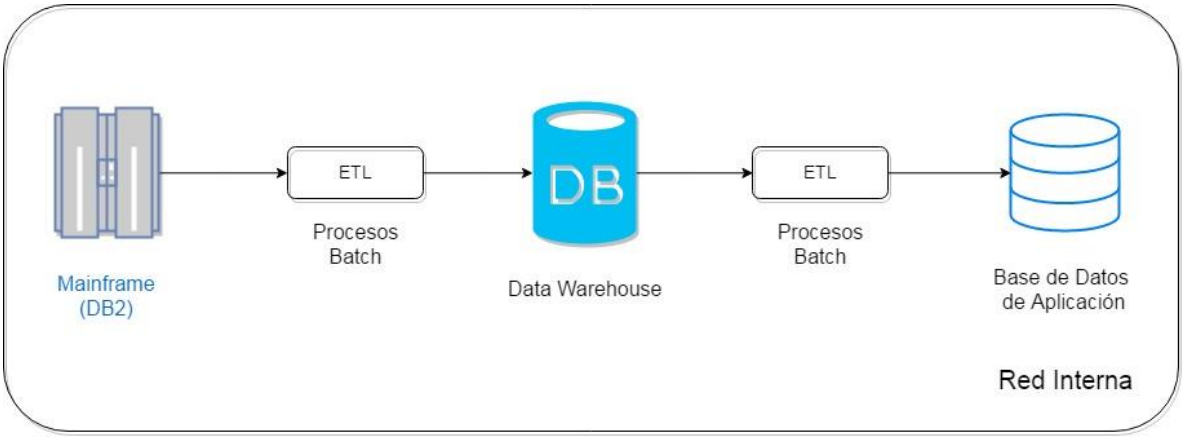


Figura 17. Arquitectura Data Warehouse (Almacén de Datos)

Fuente: Elaboración propia

El flujo de la información aquí presentado, se diseñó y realizó en conjunto con el equipo de Data Warehouse, atendiendo a las necesidades de la solución y los proceso que este equipo ya tiene definidos.

4.3.5 Proceso de carga automatizado

Toda la información de la empresa es consolidad y centralizada en la base de datos del equipo de Business Intelligence, dentro del data warehouse de la empresa (a nivel México). Para la extracción y carga de la información de data warehouse hacia la base de datos de la aplicación de la solución, se diseñó un proceso como el que se ilustra en la figura 18.

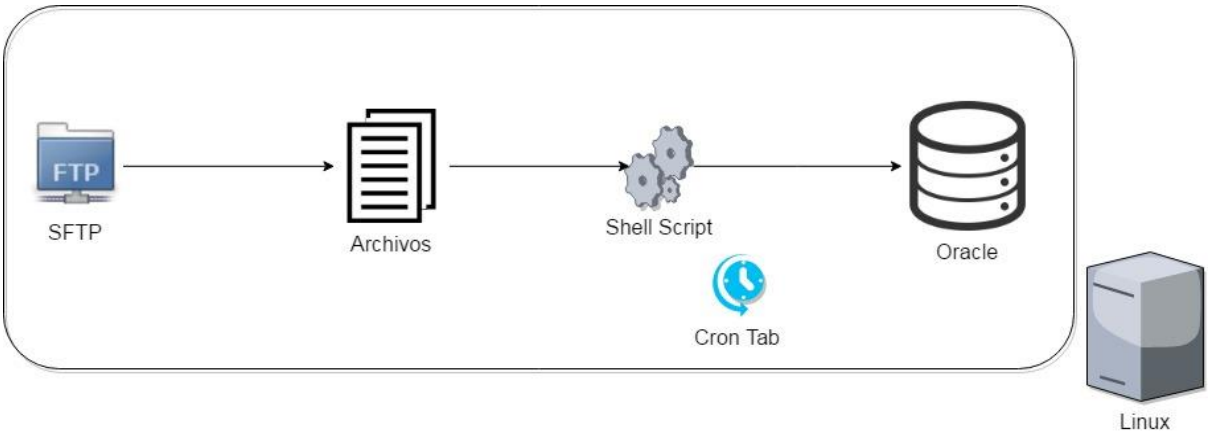


Figura 18. Proceso de carga de información

Fuente: Elaboración propia

Dicho proceso, consta de la extracción de almacén de datos a través de procedimientos almacenados en la base de datos hacia archivos de texto plano separados por tildes (~). Estos archivos serían enviados vía FTP al servidor de base de datos de la base de datos de la solución. Una vez en este servidor los archivos eran procesados por programas en Shell script para ser depositados en las rutas de carga de tablas externas de Oracle y posteriormente llamar a los procedimientos almacenados de la base de datos para la carga en las tablas de los esquemas posteriores. De esta forma se pueden controlar los errores ocurridos en los procedimientos almacenados afuera de la base de datos y enviar las notificaciones pertinentes tanto en cargas exitosas como en cargas fallidas.

4.4 Arquitectura de aplicación (orientada a servicios)

La definición de la arquitectura de la aplicación es de suma importancia ya que define la estructura e intercomunicación de todos y cada uno de los módulos de la solución de software, así como las particularidades de su construcción e implementación. En esta sección se revisa la arquitectura para la solución descrita en este documento.

4.4.1 Estructura general

La arquitectura necesaria para la solución, debía de intercomunicar 3 grandes módulos, cada uno con funciones específicas e independientes, que darían forma a la aplicación requerida. En general, se estructuró de la siguiente manera:

- **Módulo de Base de Datos.-** este módulo está conformado por el servidor a nivel de sistema operativo, la instalación e instancia de base de datos y la programación alrededor de ella. Este módulo está dentro de la red interna.
- **Módulo de Servicios Web.-** este módulo se conforma de un servidor aplicativo, la compilación de los servicios web de acceso a datos y la implementación de seguridad dentro de los servicios. Este módulo está dentro de la red interna.
- **Módulo de Aplicación Web.-** este módulo está conformado por un servidor de aplicaciones, el desarrollo de la aplicación web y una implementación de seguridad dentro de esta aplicación, a su vez. Este módulo se encuentra en DMZ.

Los módulos anteriormente descritos, cuentan con una integración entre sí, mediante diferentes protocolos de comunicación. Es decir, cuentan con una intercomunicación que les permite trabajar como una sola unidad capaz de realizar tareas particulares necesarias en el proceso de negocio. La integración descrita, se representa en términos generales, en la figura 19.

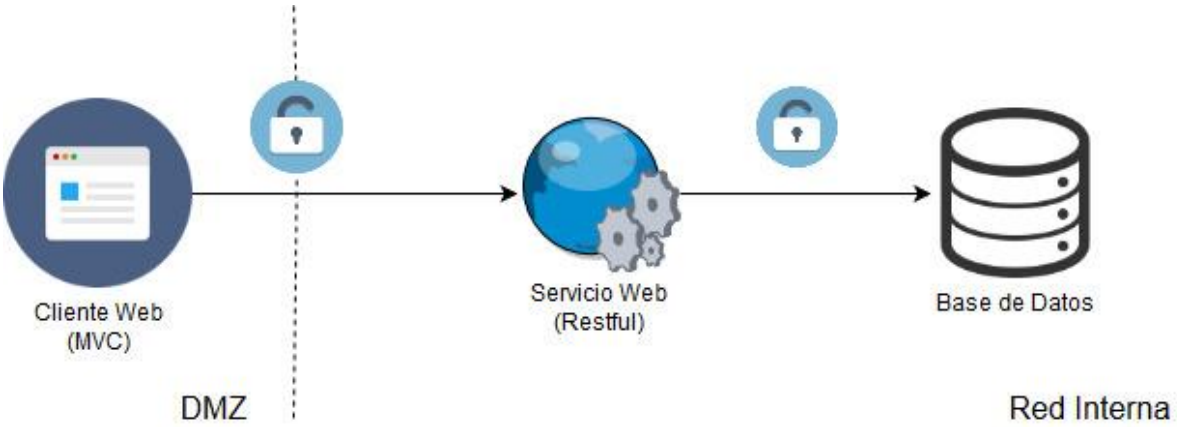


Figura 19. Arquitectura orientada a servicios

Fuente: Elaboración propia

Esta arquitectura de Servicios Rest, cumple con las necesidades de consulta y entrega de información hacia la aplicación MVC. Esta configuración segura mediante Spring Security Oauth, fue implementada por mí con el uso de mi experiencia en Spring de anteriores trabajos, de la comunicación, se muestra en términos generales, en las figuras 20 a 24.

```java
@Configuration
@EnableWebSecurity
public class OAuth2SecurityConfiguration extends WebSecurityConfigurerAdapter {

    @Autowired
    private ClientDetailsService clientDetailsService;
    @Autowired
    private DataSource dataSource;

    @Autowired
    public void globalUserDetails(AuthenticationManagerBuilder auth) throws Exception {
        auth.jdbcAuthentication()
            .dataSource(dataSource)
            .passwordEncoder(passwordEncoder())
            .usersByUsernameQuery("SELECT USERNAME, PASSWORD, ENABLED FROM USER WHERE USERNAME = ?")
            .authoritiesByUsernameQuery("SELECT USERNAME, ROLE FROM USER WHERE USERNAME = ?");
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.csrf().disable()
        .anonymous().disable()
        .authorizeRequests()
        .antMatchers("/oauth/token").permitAll();
    }
```

Figura 20. Configuración de autenticación con Spring Security Oauth

Fuente: Elaboración propia

```java
@Configuration
@EnableResourceServer
public class ResourceServerConfiguration extends ResourceServerConfigurerAdapter {

    private static final String RESOURCE_ID = "rest_api";

    @Override
    public void configure(ResourceServerSecurityConfigurer resources) {
        resources.resourceId(RESOURCE_ID).stateless(false);
    }

    @Override
    public void configure(HttpSecurity http) throws Exception {
        http.
        anonymous().disable()
        .requestMatchers().antMatchers("/test/**")
        .and().authorizeRequests()
        .antMatchers("/test/**").access("hasRole('USER')")
        .and().exceptionHandling().accessDeniedHandler(new OAuth2AccessDeniedHandler());
    }
}
```

Figura 21. Configuración de acceso a recursos mediante roles con Spring Security

Fuente: Elaboración propia

```java
package com.enlacetp.config.web;

import java.util.Properties;

@Configuration
@PropertySource("classpath:connection.properties")
public class JDBCConfiguration {

    @Autowired
    private Environment env;

    @Bean
    public DataSource dataSource() {
        DriverManagerDataSource manager = new DriverManagerDataSource();
        manager.setDriverClassName("oracle.jdbc.driver.OracleDriver");
        manager.setUrl(env.getProperty("oracle.url"));
        manager.setUsername(env.getProperty("oracle.user"));
        manager.setPassword(env.getProperty("oracle.pass"));
        manager.setConnectionProperties(jpaProperties());
        return manager;
    }

    @Bean
    public Properties jpaProperties() {
        Properties jpaProperties = new Properties();
        jpaProperties.put("hibernate.dialect", "com.oracle.jdbc.Oracle");
        jpaProperties.put("hibernate.hbm2ddl.auto", "validate");
        jpaProperties.put("hibernate.show_sql", "true");
        jpaProperties.put("hibernate.format_sql", "true");
        return jpaProperties;
    }
}
```

Figura 22. Configuración de acceso a base de datos para autenticación con Spring

Fuente: Elaboración propia

connection.properties

```
#Oracle DB Connection params
oracle.url=jdbc:oracle:thin:@localhost:1521:XE
oracle.user=user
oracle.pass=password
```

Figura 23. Propiedades de conexión en archivo properties

Fuente: Elaboración propia

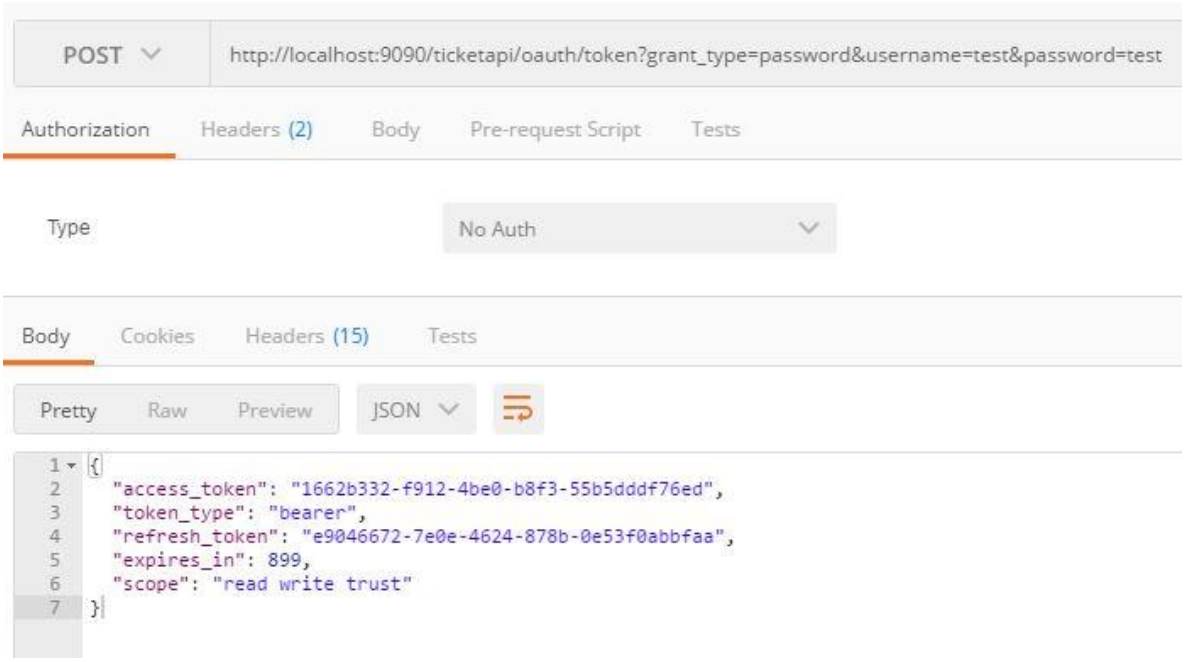


Figura 24. Post request para solicitud de token mediante credenciales

Fuente: Elaboración propia

4.4.2 Tecnologías utilizadas

Como se menciona en la subsección anterior, el uso de herramientas capaces de realizar las tareas requeridas, fue fundamental, por lo que la elección de las mismas resultó particularmente importante, a continuación, se definen las tecnologías utilizadas para el desarrollo de la solución en sus diferentes módulos de aplicación.

- **Java 7 (SE).-** El lenguaje de programación estándar para el desarrollo de aplicaciones en la empresa en Java en su versión 7. Esto debido a que se cuenta con el soporte de Oracle para esta versión. Por ende, los servidores Windows y Linux con los que se cuenta, soportan esta versión del lenguaje.
- **Java EE.-** Las aplicaciones construidas, se basan en la especificación empresarial del lenguaje, para construir soluciones de software con una alta seguridad y rendimiento.
- **Hibernate ORM**.- es un framework que implementa la especificación de Java EE para acceso a datos: JPA. Además, aumenta las capacidades de JPA para el manejo de datos, como un área de caching para las consultas frecuentes y un mapeo de registros de base de datos a objetos de programación.
- **C3P0.-** el manejador de las conexiones a la base de datos, que trabaja sobre Hibernate, para eficiente este proceso.

- **Spring framework.**- es un marco de referencia para la programación de aplicaciones basadas en la máquina virtual de Java, que implementa la inversión del control de forma eficiente y segura, además de una serie de módulos adicionales que abarcan desde acceso a datos, hasta aplicaciones web, algunos de los cuales fueron usados en el resto del proyecto.
- **Spring MVC.-** el módulo de Spring que permite construir aplicaciones web, basadas en la arquitectura MVC. Fomenta el uso de anotaciones dentro del código para el manejo de peticiones, además de proveer una gran cantidad de etiquetas HTML, personalizadas para la comunicación con la interfaz gráfica y contar con una configuración relativamente sencilla.
- **Spring REST.-** forma parte del módulo de Spring MVC, permite crear servicios REST con una configuración muy sencilla. Además de facilitar el manejo y validación de las peticiones y los parámetros. Permite enviar respuestas en formato XML o JSON.
- **Spring Security.-** Para implementar seguridad dentro de las aplicaciones MVC y REST, se hizo uso del módulo de seguridad web de Spring, el cual permite asegurar no solo las URLs del proyecto, sino también los métodos de la aplicación, mediante roles. Hace manejo automático de los request de autenticación mediante credenciales y de las sesiones mediante la configuración del proyecto.
- **Spring Oauth2.-** este módulo de Spring, implementa la especificación de Oauth2 dentro de Spring Security, en orden de realizar la autenticación por credenciales una única vez y a partir de esta, usar únicamente la validación de tokens (juego de caracteres alfanuméricos, ligados a una sesión activa).
- **Maven.-** con esta herramienta se gestionó, el manejo de dependencias y la construcción del proyecto, así como la ejecución de pruebas unitarias, en orden de hacer estos procesos más eficientes.
- **JUnit.-** esta librería de Java, permite automatizar las pruebas unitarias de la aplicación mediante anotaciones y métodos particulares de la misma, cuenta con integración con Maven y Spring para ejecutarse después de la compilación y antes de la construcción.
- **Git.-** dentro de esta herramienta, se llevó a cabo toda la administración de la colaboración dentro del código, ya que el proyecto fue codificado por 3 desarrolladores y esto aumenta la complejidad de la integración de las modificaciones y la distribución de código fuente.
- **Eclipse.**- uno de los IDEs (Integrated Development Environment) para el desarrollo en Java más completos del mercado, permite además de la edición del código, una gran cantidad de funciones adicionales, como el manejo interno de repositorios Git, ejecución de comandos de Maven y administración interna de servidores web, entre otros.

4.4.3 Servicio de autenticación

Como se menciona en la subsección de tecnologías utilizadas, en cuanto a la conexión de la aplicación web con los servicios REST, se utilizó la autenticación con Spring Security y Oauth, sistema que permite a los clientes de un API REST, acceder a los métodos del servicio con un solo login y a partir de este, otorgar al cliente un token ligado a su sesión, con el cual se le permitirá el acceso. De tal forma que no es necesario volver a intercambiar credenciales mientras el token este activo. Para el manejo de tokens por Spring, no es necesario el desarrollo de código en Java, basta con configurar este módulo de forma correcta.

Finalmente, para la autenticación de los usuarios del sistema en la aplicación web, se utilizó Spring Security con manejo de credenciales y sesiones. Es decir, los usuarios del sistema comparten sus credenciales de acceso en un formulario de login, mediante el cual se otorga el acceso al resto de la aplicación mediante roles. El manejo de la validación de credenciales y de sesiones es manejado automáticamente por Spring Security, basta con configurar esta herramienta dentro del desarrollo de la aplicación Web. Así mismo, la aplicación web es capaz de atender las peticiones de sus usuarios vía Internet, gracias a que está en la zona de DMZ.

4.4.4 Servicio REST de acceso a datos

Para la conectividad entre la base de datos y los servicios web, primeramente, se definió un esquema propio de la aplicación, dentro de la base de datos. Posteriormente, se creó un usuario de solo lectura únicamente para el uso de la aplicación en cuanto a la consulta de datos. Del lado del software, se utilizó el controlador de conexión a Oracle desde Java: JDBC en su versión 7, mismo que permite el acceso a Oracle programáticamente, teniendo como parámetros el servidor de la base de datos y el puerto en asignado a recibir conexiones, las credenciales de acceso y el nombre del servicio. Adicionalmente, para optimizar la conexión, se utilizó el manejador de pool de conexiones: C3P0 junto con Hibernate, librería de Java que maneja de forma automática las conexiones a la base de datos de forma mucho más eficiente que una implementación manual.

La implementación de Servicios Rest con Spring, en general, se muestra en las figuras 25 y 26.

```java
@RestController
@RequestMapping(value = "/test")
public class OperationController {

    final static Logger logger = Logger.getLogger(OperationController.class);

    //Add comment to Ticket
    @RequestMapping(value = "/comment", method = RequestMethod.POST)
    public ResponseEntity<String> test() {
        return new ResponseEntity<String>("Hello World", HttpStatus.OK);
    }
```

Figura 25. Controlador REST para manejo de request y envío de response

Fuente: Elaboración propia

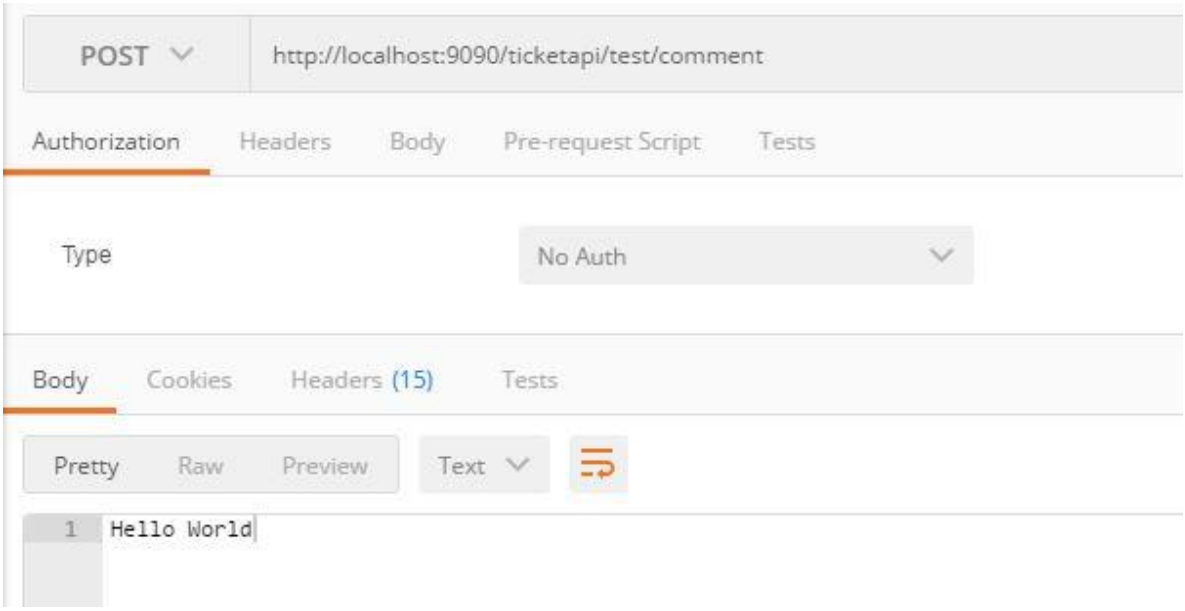


Figura 26. Consulta de método REST con post

Fuente: Elaboración propia

En cuanto a la estructura de dicho servicio, se definieron capas de aplicación en orden de contar con un proyecto sólido, mantenible, escalable y confiable, al estar alineados con las máximas de arquitectura de aplicaciones (por capas). Dicha estructura consta de una validación de request/response, ejecución de lógica de negocio y acceso/mapeo de datos, esta implementación se representa en el diagrama de la figura 27.

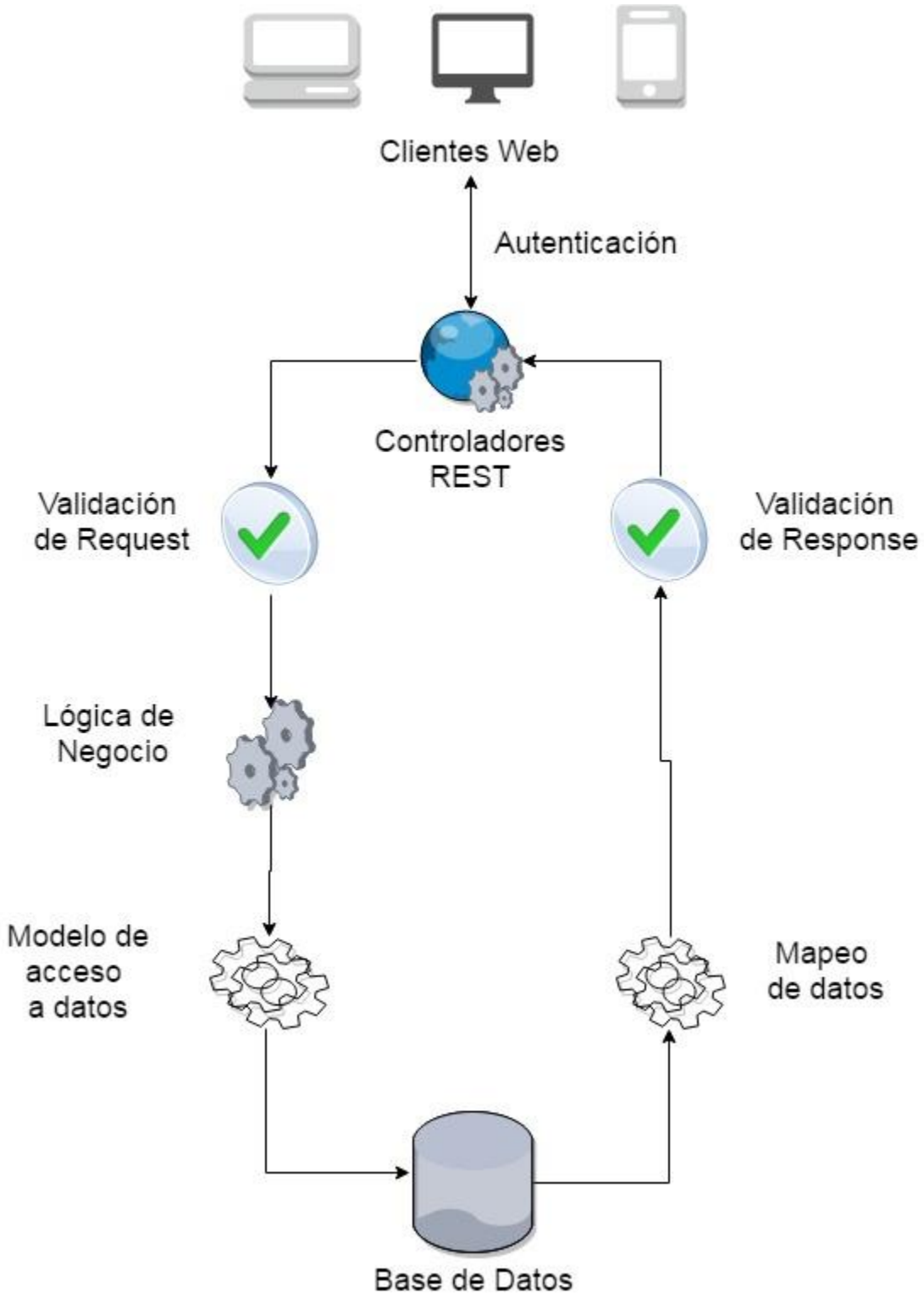


Figura 27. Arquitectura por capas en API REST

Fuente: Elaboración propia

Esta parte de la solución, así como la parte de base de datos, fue en la que mayormente estuve involucrado, tanto en el diseño, como en el desarrollo y el soporte.

4.4.5 Capa de presentación (MVC y responsividad)

Este modelo de programación se utilizó para la construcción de la aplicación web, el cual, como se menciona en el capítulo 2, permite manejar las peticiones HTTP, los controladores y los modelos. Particularmente, para el caso de la solución descrita en este documento se implementó el módulo de Spring, Spring MVC, el cual mediante configuraciones sencillas permite controlar el modelo MVC eficientemente, tal como se muestra en la figura 28.

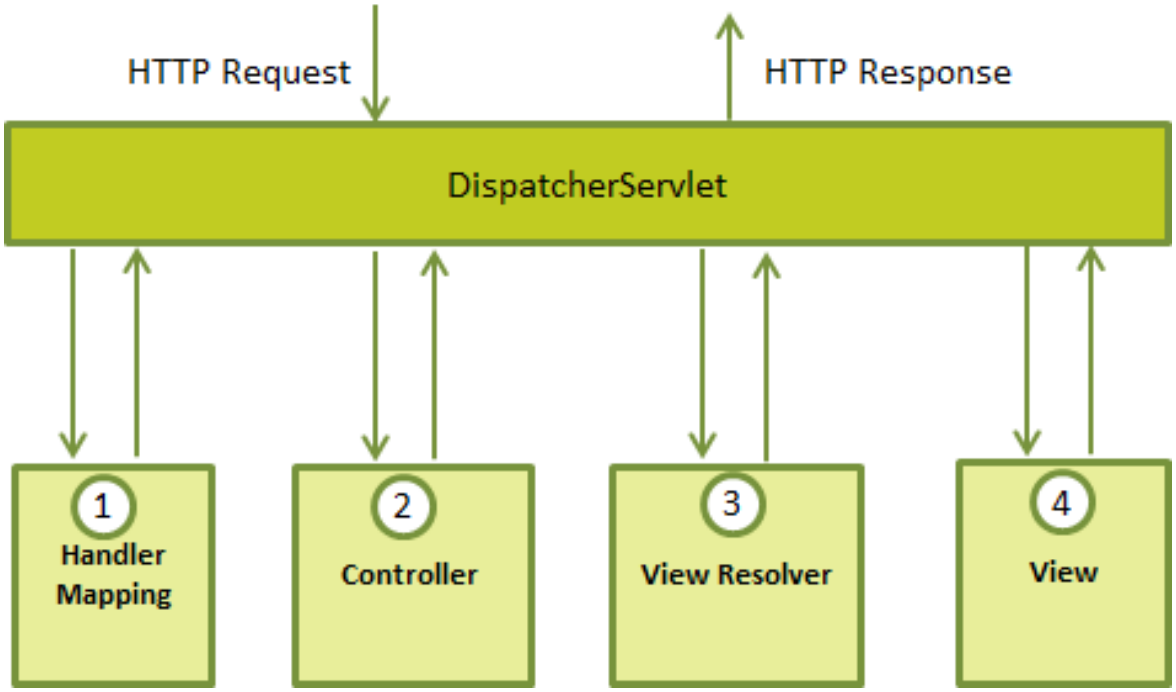


Figura 28. Implementación de Arquitectura MVC con Spring

Fuente: http://www.w3ii.com/es/spring/spring_web_mvc_framework.html (9 de Abril de 2017)

Como se puede apreciar en el diagrama anterior, al configurar Spring MVC dentro de un proyecto Web, manejará cada una de las etapas de un request de forma casi automática, añadiendo solo la lógica de negocio y una configuración adecuada para nuestro proyecto. Cabe mencionar que, dentro de esta

implementación, se integró también Spring Security, tal como se describe en la subsección anterior.

La implementación, en general, de Spring para crear sistemas basados en MVC, se puede apreciar en las figuras 29 a 31.

```
@Controller
@RequestMapping(value = "/")
public class MainController {

    //Welcome page
    @RequestMapping(value = "welcome", method = RequestMethod.GET)
    public ModelAndView root() {
        ModelAndView model = new ModelAndView();
        model.addObject("mssg", "Welcome!");
        model.setViewName("welcome");
        return model;
    }
```

Figura 29. Manejo de Request y envío de Vista poblada con modelo

Fuente: Elaboración propia

```
<%@page session="false"%>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>

<html>
    <body>
        <h1>"${mssg}"</h1>
    </body>
</html>
```

Figura 30. Definición de vista

Fuente: Elaboración propia

Figura 31. Despliegue de vista mediante request por método get

Fuente: Elaboración propia

Cabe mencionar que, para el diseño responsivo de las vistas de la aplicación, es decir, para la capacidad automática de las vistas de adaptarse a cualquier tamaño de pantalla, se utilizó el framework: Bootstrap, creado por la compañía Twitter para agilizar el diseño de vistas web responsivas. Así mismo el uso de Javascript, para refrescar únicamente los componentes de la vista que sufren actualizaciones con la respuesta de la petición a un servicio Rest, permitió un despliegue más ágil de la información. Ambos, en los que me vi muy involucrado a pesar de no estar encargado de esta capa de la solución.

4.5 Implementación de la metodología en cascada

De acuerdo a la especificación de la metodología en cascada del capítulo 2, nos ayuda a deliberar soluciones de software por etapas con requerimientos y validaciones estrictas de usuario.

Así mismo, esta metodología de desarrollo se adapta a las necesidades de la empresa, ya que antes de comenzar cada proyecto de desarrollo, se cuenta con todos los requerimientos del sistema. Adicionalmente, permite regresar una etapa en caso de que se requiera por el equipo. Esta metodología favorece la documentación y las pruebas de usuario, las cuales, son muy importantes para la gestión de cambios dentro de ITIL.

La secuencia de cada una de las fases de esta metodología se ilustra en la figura 32.

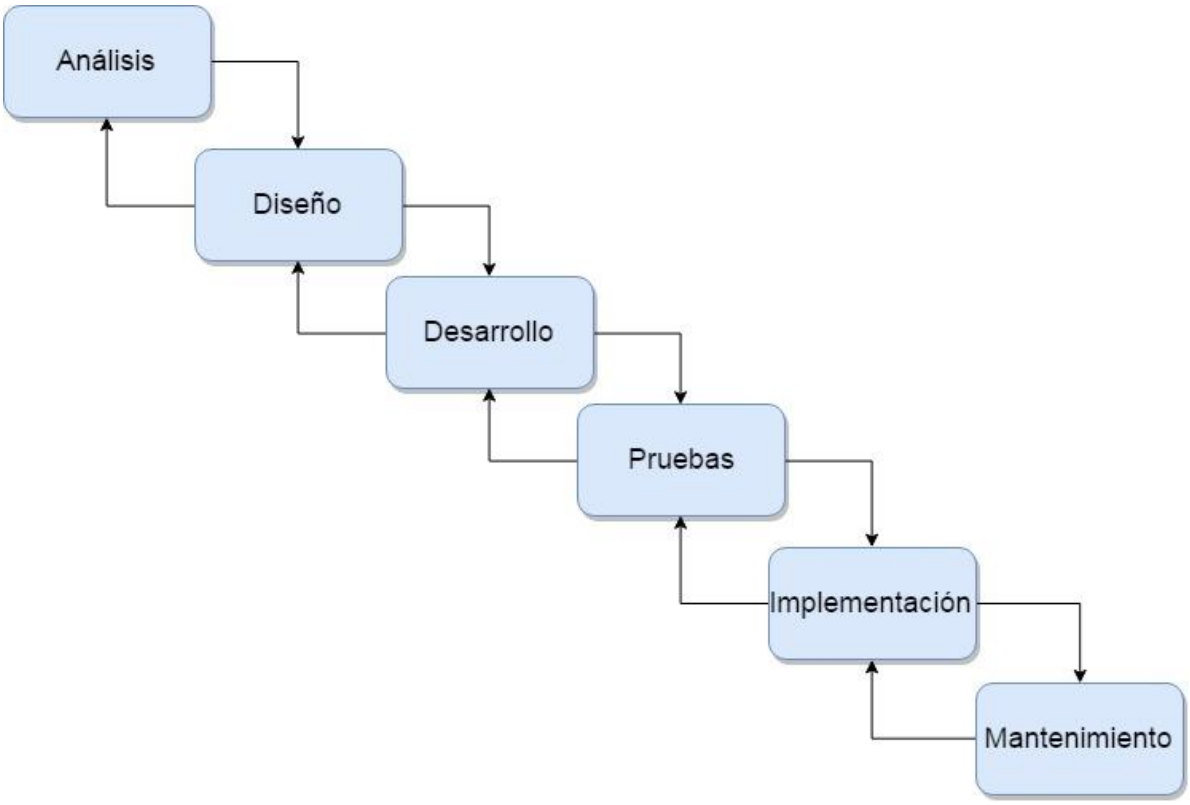


Figura 32. Modelo en cascada modificado

Fuente: Elaboración propia

4.6 Manejo de contenido estático y analíticas web

Es importante mencionar que el contenido estático mostrado en las aplicaciones web de la empresa no es manejado directamente por cada aplicación, sino que se cuenta con un sistema de gestión de contenido estático que permite acceder a él de forma segura mediante peticiones http. En este sentido, la aplicación web desarrollada como parte de la solución, se apegó a este proceso y delegó el manejo del contenido estático a este sistema.

Adicionalmente, se implementó un sistema de analíticas web de la empresa Adobe, dentro del sistema, en orden de obtener métricas y estadísticas del uso de la aplicación web. Este sistema consiste en agregar un segmento de código de Javascript a la aplicación web, que de manera asíncrona (con AJAX) envía mediciones a un servidor externo en Internet de Adobe, mismo que se encargaría de evaluar, filtrar, ordenar y acumular estas métricas para mostrar estadísticas del uso de la aplicación.

4.7 Implementación de ITIL

De acuerdo a las necesidades particulares de la empresa en cuanto a la atención a los usuarios de los servicios de TI, se decidió por parte de la dirección general global de TI en la empresa, implementar el sistema gestor de Mesa de Servicio de HP: Service 1.

Esta herramienta tiene una implementación de la especificación de ITIL dentro de la administración de su sistema. Adicionalmente, brinda soporte de 1° nivel, vía telefónica, por chat y por correo electrónico para el levantamiento, actualización, seguimiento y cierre de tickets, así como el soporte global de expertos en tecnologías de información a disposición de la empresa en cualquier momento, para atender incidencias de 3° nivel. En general, permite a las empresas no solo gestionar toda la mesa de servicio basada en ITIL, sino también centralizar y contratar mediante un tercero el soporte técnico de las tecnologías de información.

Cabe mencionar que no por el hecho de contar con el soporte de HP en este sentido, será conveniente carecer de personal especializado en cada una de las tecnologías dentro de la empresa, pues esto traerá demasiados inconvenientes, como el hecho de no contar con soporte local inmediato. De acuerdo a las recomendaciones de HP es conveniente contar con especialistas de manera local, para brindar soporte de 2° nivel, que, en muchas ocasiones, es brindado por equipos que no son directamente de soporte, como equipos de desarrollo de software, administración de Base de Datos o administración de Sistemas Operativos.

Gracias los procesos alineados a ITIL y la herramienta especializada de HP, el control de Incidentes, Requerimientos, Cambios y Problemas está altamente controlado dentro de la empresa. Por ende, la medición de la efectividad de los equipos, se puede medir y comparar históricamente, para mantener en continua mejora su rendimiento de acuerdo a estos indicadores. (Hp, 2017)

4.8 Pruebas y validación

De acuerdo a las políticas de la empresa, los sistemas construidos deben pasar por un amplio periodo de pruebas, en la que se validará el correcto funcionamiento del sistema de acuerdo a lo esperado en diferentes rubros. Estas pruebas se deberán documentar en una matriz y subir a la herramienta de Service Desk en el apartado de control de cambios. A continuación, se describe cada una de las secciones que se documentó en la matriz de pruebas para la liberación de la solución.

4.8.1 Pruebas Unitarias

Las pruebas unitarias consisten en probar individualmente todos y cada uno de los módulos desarrollados para asegurar el correcto funcionamiento de manera independiente.

Por lo general, este tipo de pruebas se automatizan con herramientas como JUnit en el caso de Java, para ser ejecutadas durante la compilación del proyecto y de este modo asegurar que no se afecten módulos estables al desarrollar módulos nuevos.

Automatización

Como se menciona en la subsección de herramientas, la automatización de pruebas unitarias y de funcionalidad puede construirse con JUnit, librería de Java que existe con este propósito.

La implementación de Junit, en general, se representa en las figuras 33 y 34, con ejemplos utilizados en la construcción de la solución.

```java
import com.enlacetp.caUtils.CADate;

public class ConvertDate {

    final static Logger logger = Logger.getLogger(ConvertDate.class);

    @Test
    public void test() {
        String unixDate = "123456789";
        assertTrue(unixDate instanceof String);
        String dateString = CADate.convertCADate(unixDate);
        DateFormat format = new SimpleDateFormat("yyyy-mm-dd hh:mm:ss");
        Date date;
        try {
            date = format.parse(dateString);
            logger.info("Date: " + date);
            assertTrue(date instanceof Date);
        } catch (ParseException e) {
            logger.error(e.toString());
            e.printStackTrace();
        }
    }
}

```

Figura 33. Codificación de pruebas unitarias con JUnit

Fuente: Elaboración propia

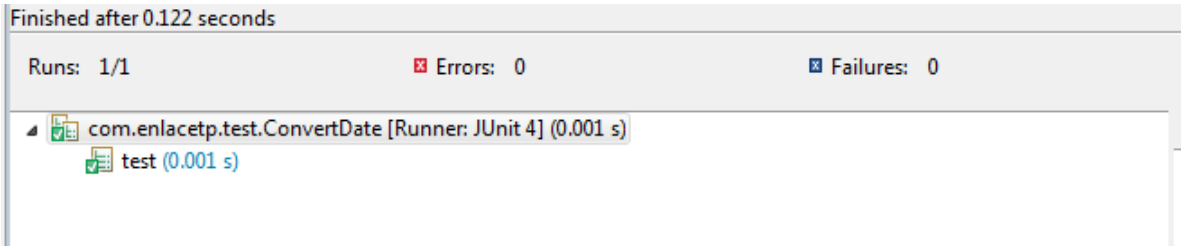


Figura 34. Resultado de la ejecución de pruebas unitarias de JUnit en Eclipse

Fuente: Elaboración propia

Para ello se realizó la construcción de casos de prueba para cada uno de los métodos expuestos por la aplicación web como los servicios REST. De tal modo que, al realizar cambios o correcciones en cualquiera de estas, se aseguraba el correcto funcionamiento de dicho método y el resto de los métodos. Este proceso

fue muy conveniente ya que ahorró mucho trabajo al equipo, en términos de validación. Además, la evidencia de estas pruebas se tomó en cuenta para la construcción de la matriz de pruebas.

4.8.2 Pruebas de integración

Las pruebas de integración consisten en asegurar el correcto funcionamiento de todo el flujo de la aplicación. Para la automatización para este tipo de pruebas se debe tomar en cuenta el control de las entradas y salidas de acuerdo al comportamiento esperado y simular los procesos que el sistema llevará a cabo.

Es importante tomar en cuenta que para estas pruebas se debe comenzar desde la capa de presentación, asegurar el correcto flujo y comunicación con la capa de servicios y con la capa de datos, asegurando la correcta integración entre todas las capas, cuando se generan peticiones desde la capa de presentación. Finalmente, revisar a detalle la respuesta que se obtiene en la capa de presentación y validar que esté correcta, de acuerdo a lo esperado.

Para la automatización de estas pruebas, el ingeniero de la capa de presentación, quien coordinó este proceso, utilizó la herramienta JMeter de Apache, con la cual se pueden lanzar peticiones consumiendo la capa de presentación. Mi rol en estas pruebas, fue monitorear los servicios web y la base de datos en tiempo real mientras se corrían estas pruebas. Adicionalmente, JMeter genera reportes, logs y gráficas de las ejecuciones, lo cual resultó muy útil para documentar estas pruebas. La interfaz de JMeter se puede apreciar en la figura 35.

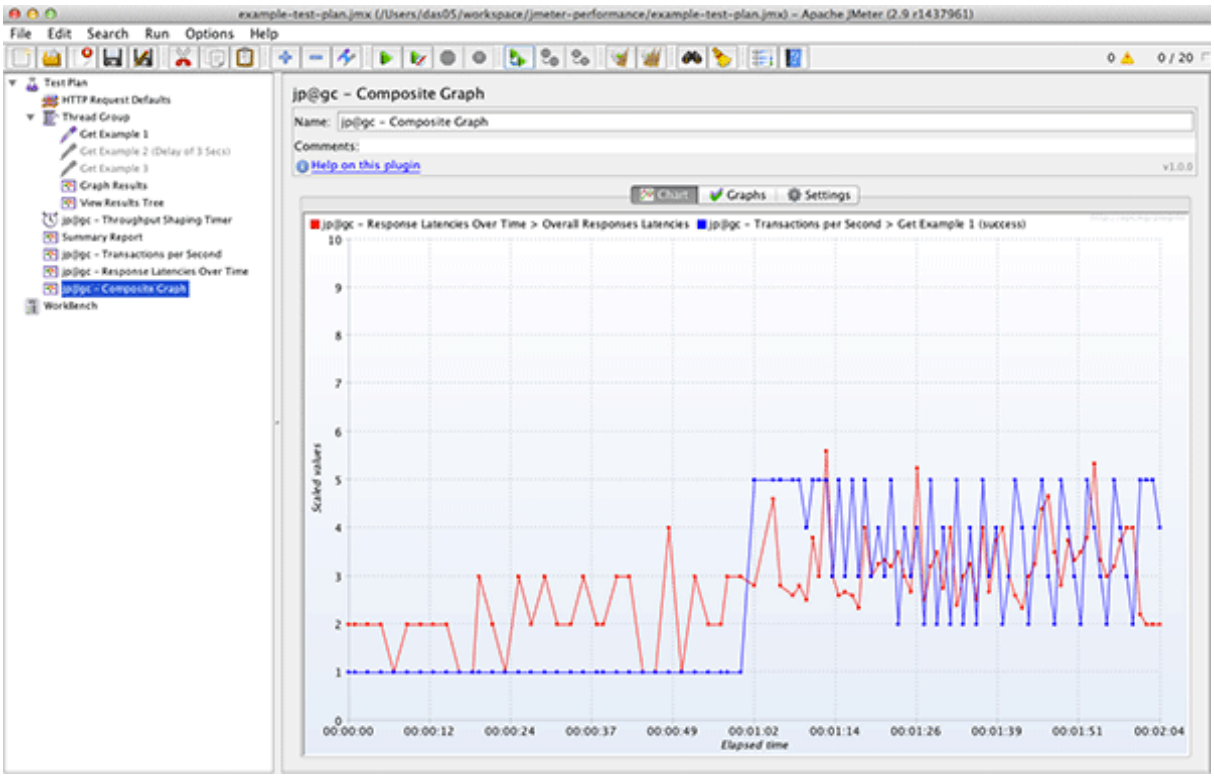


Figura 35. Interfaz de Apache JMeter

Fuente: https://cdn.tutsplus.com/net/uploads/2013/09/jmeter-screen-7.png (9 de Abril de 2017)

4.8.3 Validación de manejo de errores

Las pruebas de manejo de errores se refieren a asegurar que la aplicación responda de manera adecuada a los diferentes tipos de errores que se puedan llegar a presentar en la aplicación.

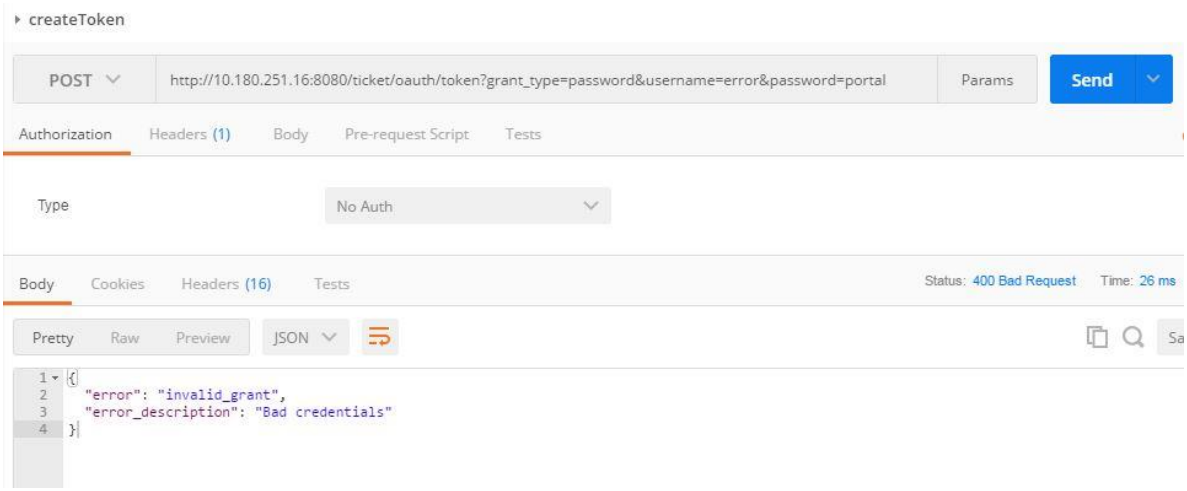
Este tipo de pruebas son fáciles de automatizar ya que se probarán módulos en concreto con entradas diferentes a las esperadas y se validará que la respuesta a ello sea la esperada.

En términos prácticos, estas pruebas consisten básicamente en enviar dentro de la petición (request), valores incorrectos y asegurar que la aplicación los maneje de forma correcta. Desde que estuve encargado de los servicios web, las pruebas de manejo de errores, estuvieron a mi cargo.

En la figura 36, se muestra como se validó el correcto manejo de errores en los servicios web, para un método en particular del servicio desarrollado, mediante Postman, el cliente para servicios REST.

Método: Crear Token

Credenciales inválidas



URL incorrecta

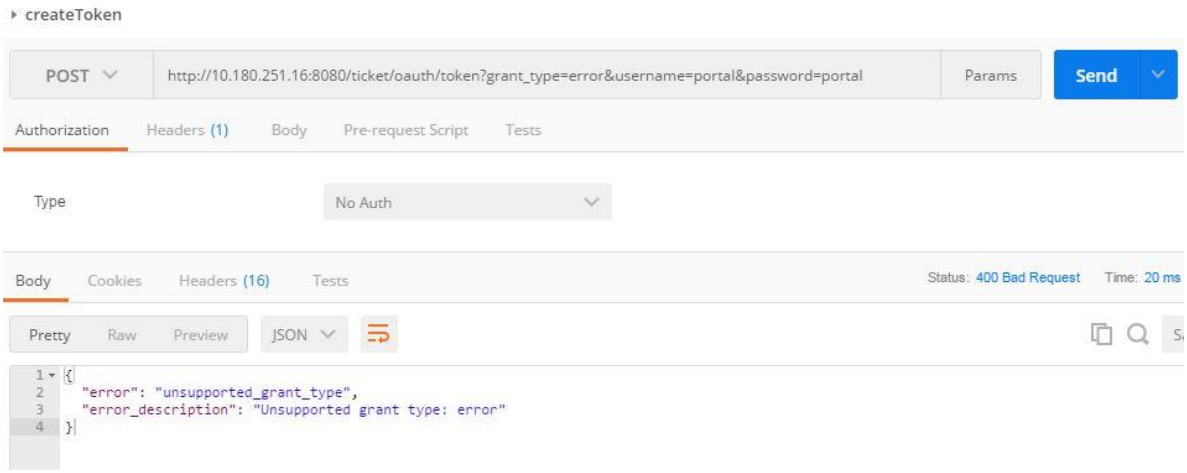


Figura 36. Codificación de pruebas unitarias con JUnit. Credenciales inválidas y URL incorrecta

Fuente: Elaboración propia

4.8.4 Pruebas de rendimiento

Las pruebas de rendimiento consisten en llevar al sistema desarrollado a sus límites de desempeño dentro de un ambiente parecido en cuanto a capacidades al ambiente en el que se liberará en producción, en orden de conocer los límites de respuesta de la aplica

ción y verificar que tenga un comportamiento adecuado de acuerdo a los requerimientos y criterios de aceptación.

Para este tipo de pruebas existen herramientas de acceso a servicios web que nos permitirán automatizar el proceso de envío una petición. Para el caso de aplicaciones independientes, podremos utilizar las pruebas automatizadas, desarrolladas para las pruebas unitarias, con programación paralela para automatizar el alto consumo de este tipo software. Finalmente, en el caso de aplicaciones con interfaz gráfica, ya sean de escritorio, web o para dispositivos móviles, se habrán de usar herramientas de automatización de pruebas de interfaz.

Así mismo para monitorear el rendimiento que presentan las bases de datos o aplicaciones en cuanto a uso de cpu, memoria y manejo exitoso de peticiones, se pueden instalar agentes de monitoreo para el reporteo y generación de estadísticas automático, de acuerdo al desempeño que presenten en cuanto al uso de estos recursos. software que es recomendable usar, no solo para el periodo de pruebas sino también para el monitoreo en producción.

Para hacer las pruebas de rendimiento de la aplicación a nivel de servicios se usó la herramienta SOAP UI, misma que te permite enviar peticiones simultaneas a un servicio REST, durante un periodo de tiempo determinado, para asegurar que la aplicación tiene un comportamiento adecuado bajo estrés.

Una vez lanzando las peticiones automatizadas desde SOAP UI, es importante, revisar en tiempo real, el uso de memoria y cpu, así como los logs y las sesiones en base de datos, para asegurar no solo que todas las peticiones lanzadas cuenten con una respuesta, sino que también el uso de recursos sea coherente con lo que se espera, de acuerdo al volumen de la prueba.

Cabe mencionar, que las pruebas de rendimiento para los servicios web, también estuvieron a mi cargo. En las figuras 37 a 39, se muestra la interfaz de SOAP UI, para este proceso.

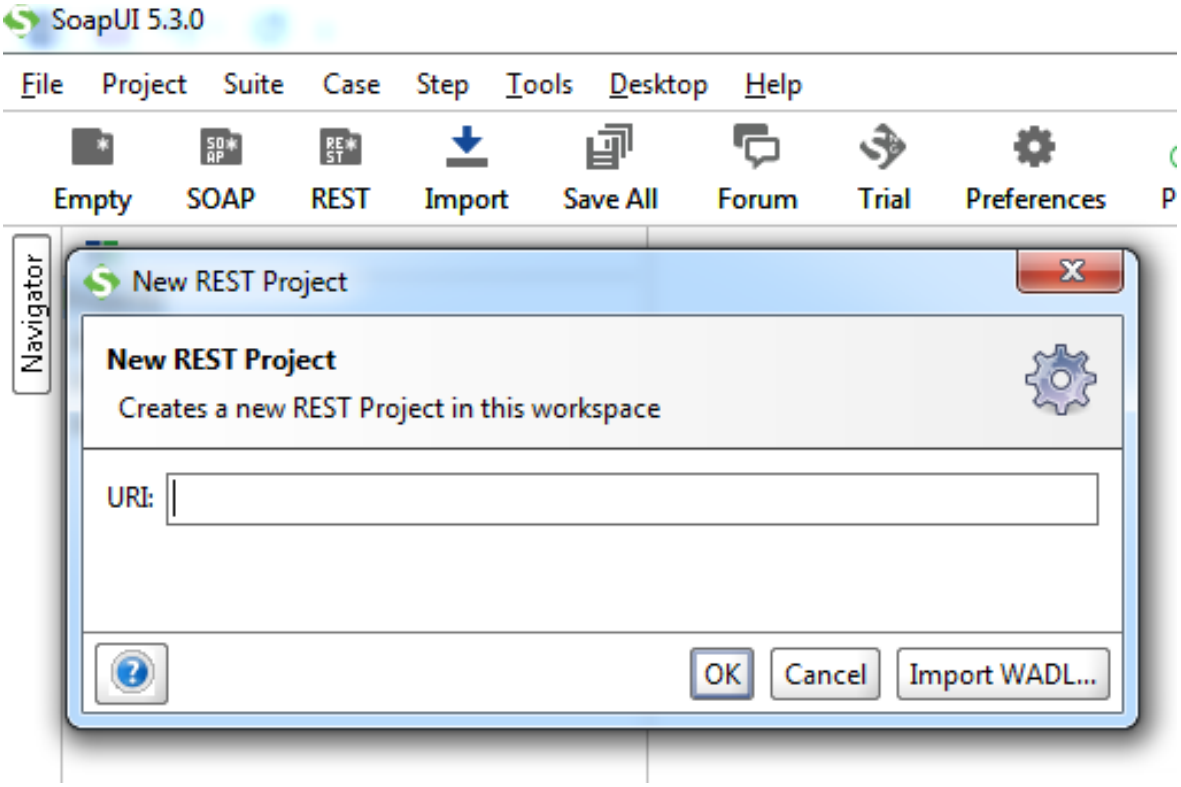


Figura 37. Creación de un proyecto REST en SOAP UI

Fuente: Elaboración propia

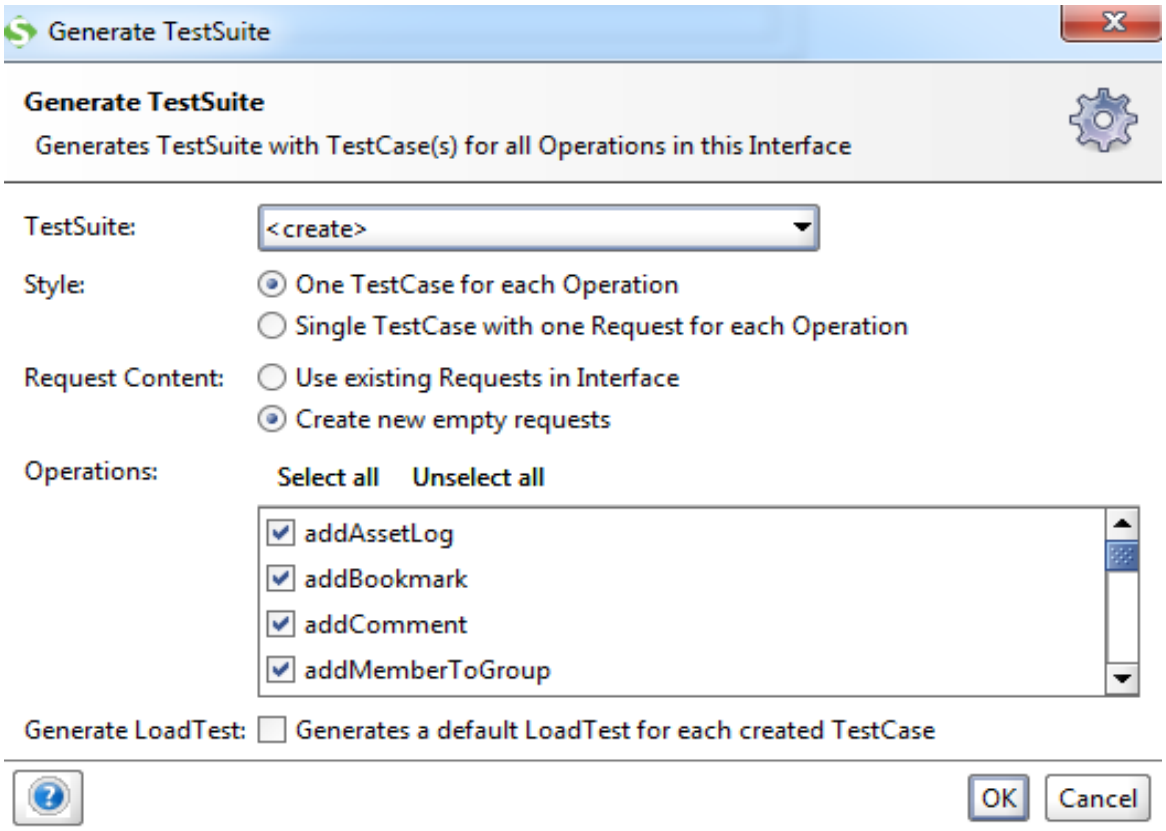


Figura 38. Creación de un proyecto de pruebas, sobre el servicio REST

Fuente: Elaboración propia

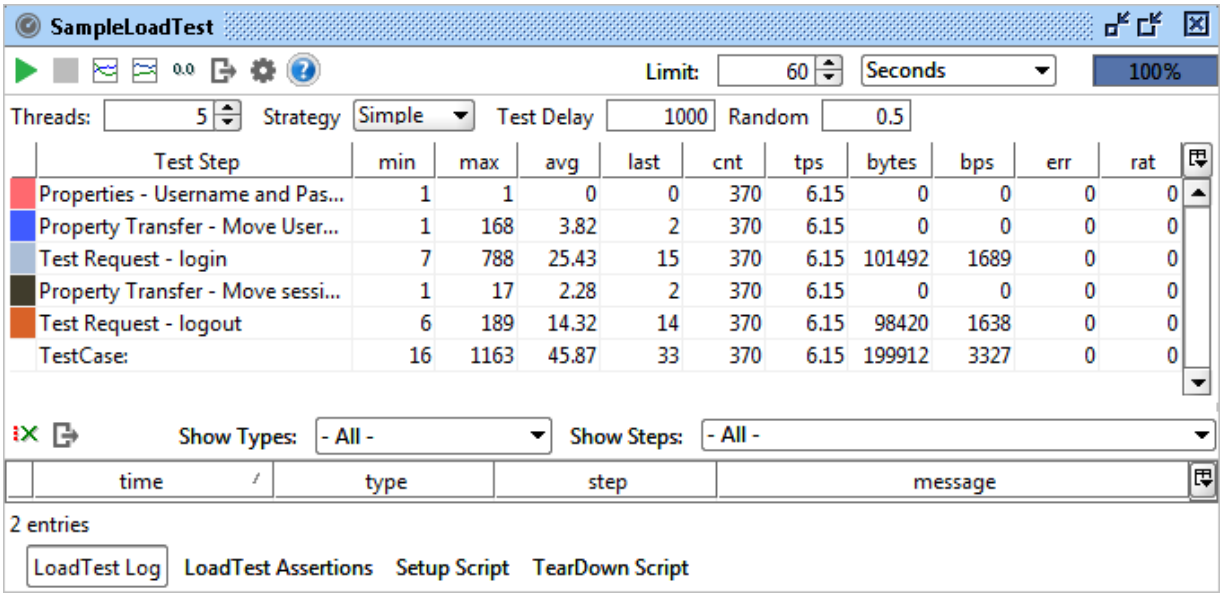


Figura 39. Ejecución de peticiones simultáneas para pruebas de rendimiento

Fuente: Elaboración propia

4.8.5 Pruebas de seguridad

Estas pruebas, consisten en asegurar que las aplicaciones que se liberan a DMZ cuenten con la seguridad suficiente para controlar ataques maliciosos con objetivos de denegación de servicio, extracción de información, robo de identidad o generación de transacciones monetarias a favor del atacante.

En el caso de Java, para el aseguramiento de una aplicación expuesta a Internet, existen diferentes frameworks que permitirán a los programadores una implementación sencilla y fácil de configurar, como el utilizado para este proyecto: Spring Security.

Como se mencionó en la sección 4.4.1, la correcta configuración de Spring Security, protege a las aplicaciones Web en diferentes niveles, con:

- Autenticación
- Manejo de sesiones
- Manejo de tokens caducables
- Administración segura de request/response HTTP
- Denegación de acceso a urls vulnerables
- Denegación de acceso a métodos de acuerdo a roles
- Administración de ataques de denegación de servicios

Entre otros.

Adicionalmente, por la delicadeza de este tema, la empresa tiene como regla, el sometimiento de las aplicaciones por liberar a DMZ ante software especializado que detecta vulnerabilidades en las aplicaciones. Para la liberación de servicios web o aplicaciones web es necesario pasar por este proceso, mismo que recomendará las oportunidades de mejora para mitigar dichas vulnerabilidades.

Para el caso de la solución, la aplicación fue sometida a pruebas de vulnerabilidades Web con la herramienta de HP: Fortify. Misma que permite escaneo preciso de las potenciales vulnerabilidades de las aplicaciones web. En dichas pruebas, se requirió que la aplicación Web, no obtuviera resultados de

amenazas potenciales. En la figura 40, se muestra la interfaz de la herramienta Fortify, de HP.

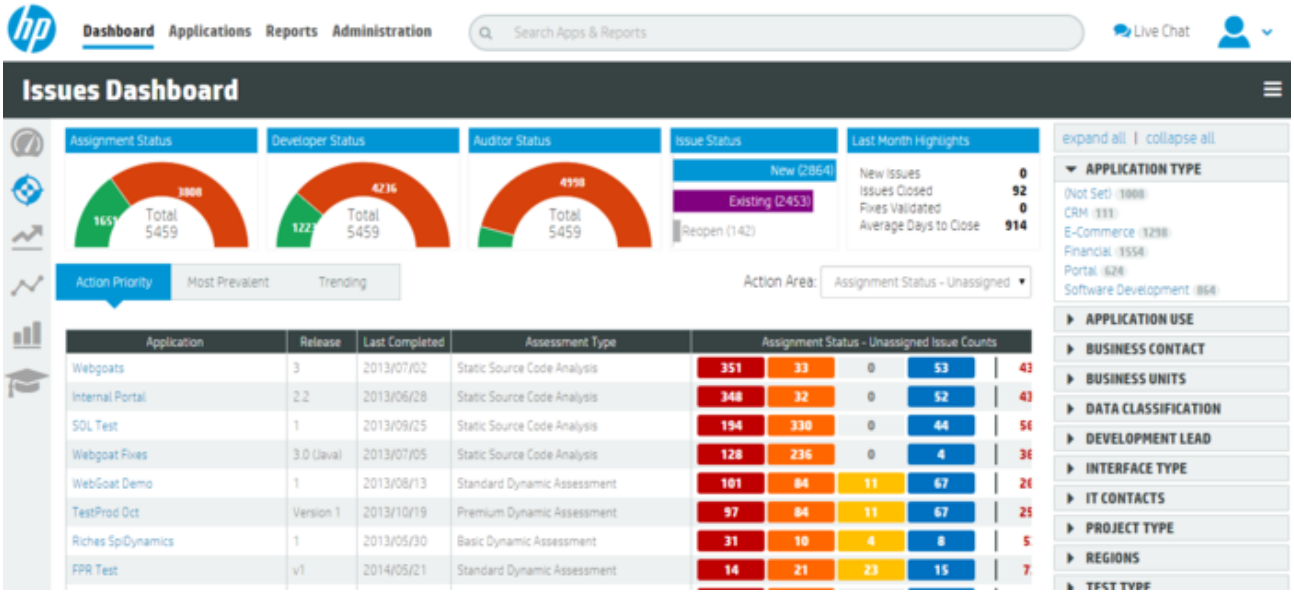


Figura 40. Interfaz gráfica de la herramienta Fortify de HP

Fuente: https://cdn.jaleco.com/gen_screenshots/en-US/windows/hp-fortify-on-demand/large/image-02-700x321.png (9 de Abril de 2017)

Cabe mencionar que el manejo de esta herramienta y la responsabilidad de estas pruebas, es del equipo de seguridad de la información, mismos que ejecutarán las pruebas de penetración y de acuerdo al resultado, recomendarán al equipo de desarrollo aplicar ciertos cambios para evitar las vulnerabilidades. Este proceso se repetirá hasta que la aplicación no tenga vulnerabilidades web.

4.8.6 Pruebas de usuario

Finalmente, las pruebas de usuario son las que certifican a la aplicación por parte de negocio de cumplir con todos los puntos especificados en los requerimientos. De ser así, se dará un visto bueno, mismo que se documentará por escrito vía correo electrónico, y se añadirá al control de cambios en la herramienta de Service Desk.

Las pruebas de usuario pueden consistir en la revisión de la matriz de pruebas construida por el equipo y que documenta todas las pruebas anteriores. Adicionalmente, se pueden solicitar casos de prueba muy puntuales, de acuerdo a las consideraciones del equipo de negocio.

Para este tipo de pruebas, los responsables de la definición del producto, acceden a la aplicación web, mediante un navegador y recorren todas y cada una de las páginas del sitio, asegurando que el sistema cumpla con los requerimientos a nivel de diseño, funcionalidad e información.

4.9 Liberación y mejoras post-producción

4.9.1 Liberación a producción

Una vez realizadas las pruebas correspondientes del sistema y pasadas las aprobaciones pertinentes, se realizó la liberación del sistema a producción, con lo cual, quedaría expuesto a Internet, mediante la DMZ y daría el servicio a los usuarios para el que fue concebido. Este proceso queda estrictamente documentado en un plan de implementación, mismo que es compartido con todo el personal involucrado este proceso, en el cual se especifican las tareas requeridas para la liberación de la aplicación, la validación y de ser necesarias, para revertir los cambios realizados, en caso de que se presente un impedimento que imposibilite la correcta liberación de la solución.

El procedimiento que se sigue para las liberaciones a producción es que se calendariza un evento de liberación productiva, asegurando la participación de todos los involucrados en la misma. Es decir, los responsables de la administración de los elementos con lo que se relaciona la aplicación deberán estar presente (física o virtualmente) para dar soporte a esta promoción.

En el caso de la solución de software descrita en este documento, se requirió la participación de los siguientes responsables en la empresa:

- **Administrador de Unix/Linux.-** apoyo para configurar el ejecutor de tareas automático y
- **Administrador de Bases de Datos.-** apoyo para ejecutar los scripts sql de definición y manipulación de datos (DDL y DML).
- **Responsable de Cambios.-** apoyo para distribuir los elementos compilados y de recursos a las rutas correspondientes en los servidores productivos.
- **Administrador de Servidor de aplicaciones.-** apoyo para configurar de forma correcta la instalación de la aplicación dentro del servidor de web.

Cada uno de los cuales ejecutaría su tarea en el orden correspondiente, asignado en el plan de liberación y validaría que la ejecución de la misma de

acuerdo a lo documentado fuese exitosa. Una vez que las tareas de todo el plan fueron realizadas, se realiza una validación completa del flujo de la aplicación para validar la correcta instalación de la misma y se notifica a las áreas de negocio y los gerentes de tecnología del estatus de la implementación.

4.9.2 Beneficios para los usuarios finales

Después de la construcción de los procesos ETL, la base de datos, los servicios REST y la aplicación web, fueron (en general), vistas como las que se muestran en las figuras 41 a 43.

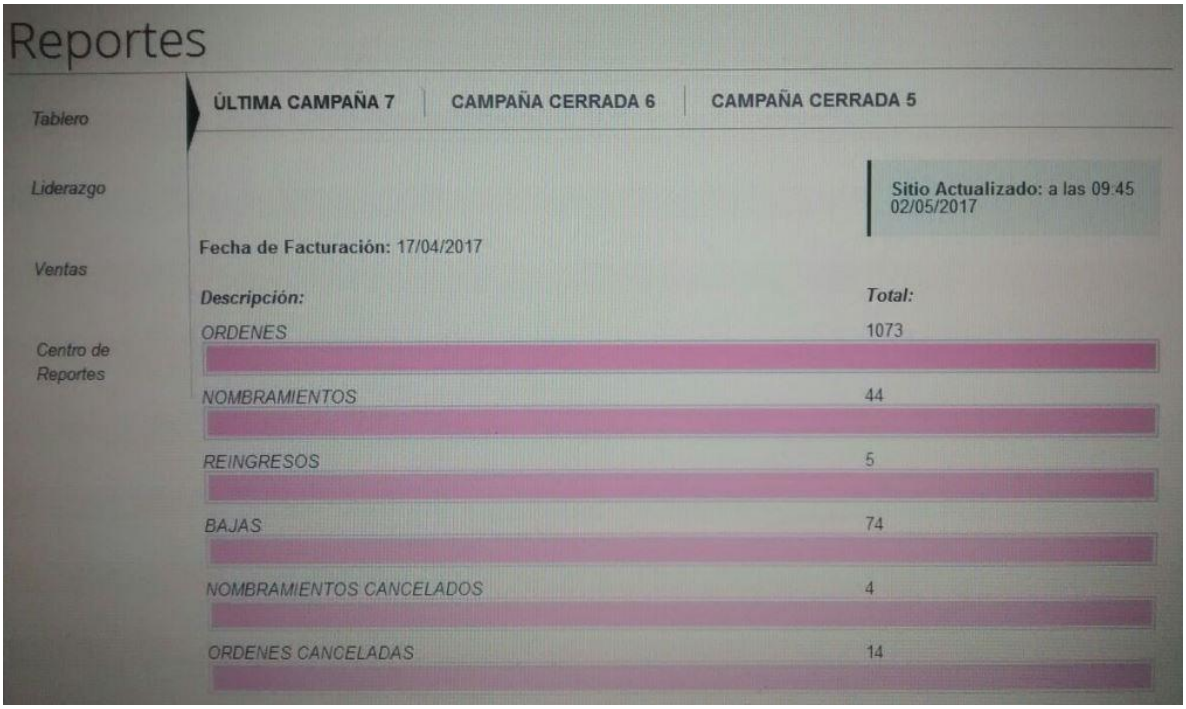


Figura 41. Tablero principal del sistema

Fuente: Sistema implementado en la empresa

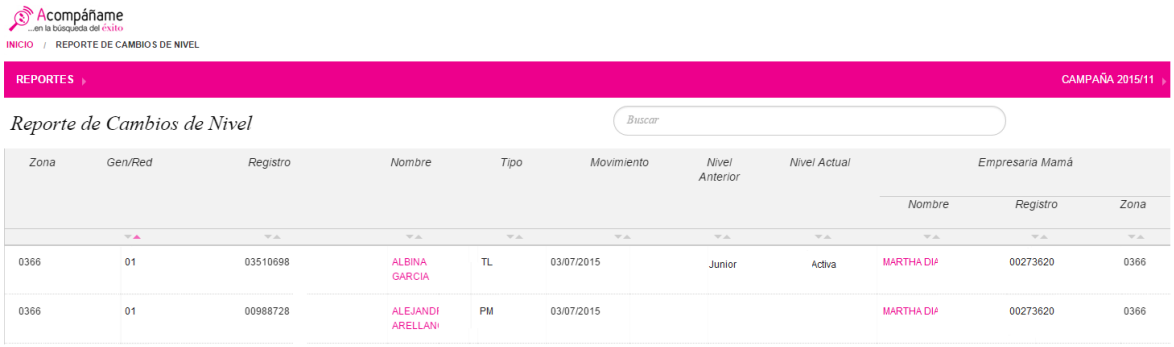


Figura 42. Reporte específico del sistema

Fuente: Sistema implementado en la empresa

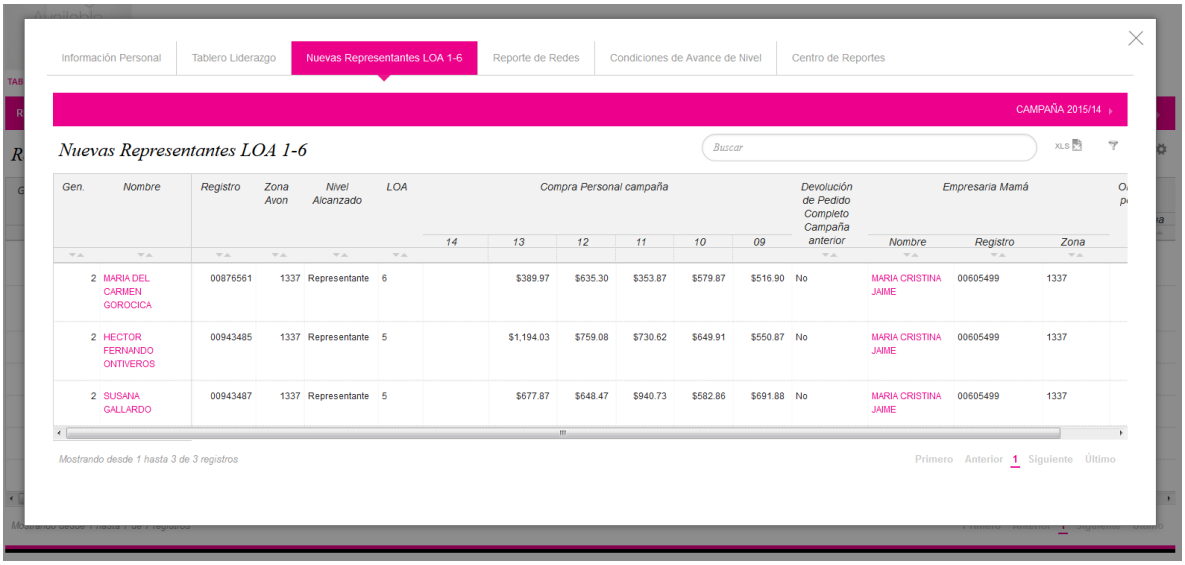


Figura 43. Reporte impersonalizado del sistema

Fuente: Sistema implementado en la empresa

4.9.2 Decisiones orientadas por métricas

Para la toma de decisiones de negocio y la evaluación del impacto del sistema con los usuarios finales, se hizo uso del sistema de analíticas, implementadas en la aplicación web.

A partir de las estadísticas obtenidas en este sistema se pudo determinar lo siguiente:

- Reportes más visitados por las usuarias del sistema.

- Reportes en los que las usuarias pasaban más tiempo.
- Horas del día con más tráfico en el sitio.
- Días de la semana con más tráfico en el sitio.
- Qué reportes eran más exportados (descargados) a Excel.

Gracias a los resultados obtenidos, se pudo proponer mejoras en el orden en el que se presentaban los reportes, el detalle que debía tener cada uno, las horas más convenientes para actualizar la información y las métricas que hacían falta de agregar a los reportes.

4.9.3 Resultados para el negocio

De acuerdo con el sistema de analíticas y los reportes de negocio, gracias a la solución descrita en este documento, el pedido de orden vía Web aumentó del 85% al 96%, debido al seguimiento que podían ahora realizar las líderes y gerentes de ventas. Cabe mencionar que, en la empresa, se incentivan los pedidos vía Web ya que esto ahorra gastos a la empresa y favorece el cuidado del medio ambiente, al reducir costos de personal y de uso de papel.

Adicionalmente, se detectó un aumento muy positivo en el monto total de las órdenes que se ingresaban, pues al igual que con las ordenes vía web, la empresa incentiva los pedidos de cantidades más grandes. Gracias a la solución de software, las líderes ahora podían dar un seguimiento más puntual a la cantidad ingresada de su personal.

4.9.4 Resultados tecnológicos

Debido a la magnitud y alcance de este proyecto, se tuvo la oportunidad de implementar tecnologías de muy alta calidad dentro de la solución, no solo a nivel de programación de aplicaciones y de bases de datos, sino también a nivel de colaboración, compilación y automatización de procesos. Pues se implementó el control de versiones de código con Git y Gitlab, el manejo de dependencias, compilación y construcción del proyecto con Maven y se buscó la automatización de pruebas unitarias y de funcionalidad con JUnit.

4.9.5 Mejora continua

De acuerdo a las políticas de la empresa, solo se pueden destinar esfuerzos de desarrollo a una solución si se tiene un proyecto destinado a ello o la solución sufre de algún incidente en producción el cual haya que agregar. O bien, si la

solución tiene la suficiente importancia como para considerarse propia de mejora continua. Es decir, si en conjunto con las áreas de negocio se destina el presupuesto para mantener personas dedicadas al desarrollo de mejoras y optimización. Este es el caso de la solución descrita en este documento, ya que es una aplicación que beneficia directamente a las áreas de venta.

4.9.6 Mejoras realizadas, a nivel de negocio

Una vez liberada la aplicación, las áreas de negocio continuaron estudiando las formas en las que podrían mejorar la forma en la que se muestran los reportes y las revisando la relevancia de las métricas que eran presentadas, así como mejorando los procesos de negocio y las tablas de comisiones.

Estos cambios y mejoras a nivel de negocio, representaron una necesidad de cambio, a nivel de la solución de software, por lo que se hicieron nuevos requerimientos de negocio, en cuanto a los datos que se consumían, los estilos de las vistas, el orden en que se presentaban los tableros y los mensajes mostrados dentro de la aplicación.

4.9.7 Mejoras realizadas, a nivel de desarrollo

Los nuevos requerimientos recibidos por las áreas de negocio de acuerdo a las mejoras que se identificaron, fueron trabajados y construidos por el mismo equipo que liberó la primera versión.

Así mismo, el equipo de desarrollo identificó mejoras de rendimiento en los procesos de consumo y carga de información, que se refactorizaron para obtener mejores tiempos de respuesta. Las mejoras en cuanto a la navegación del sistema web fueron incluidas en esta segunda versión. El proceso de desarrollo, pruebas, documentación y liberación fue el mismo que para la primera versión.

4.9.8 Cliente móvil e implementación en Costa Rica

Gracias al éxito obtenido en la solución de software, en términos de beneficios para el negocio y aceptación con sus usuarios, se decidió lanzar la aplicación móvil (teléfonos celulares) del sistema, para los sistemas operativos Android y iOS. Gracias a la arquitectura de la aplicación no se requirió desarrollo o modificaciones en los servicios ya que estaban optimizados para la consulta mediante métodos HTTP, la única modificación necesaria fue exponer estos servicios en DMZ, en orden de ser alcanzados por el cliente móvil desde Internet con la autenticación del usuario en directamente en la aplicación móvil.

Adicionalmente, el mercado de la empresa en Costa Rica, decidió tomar las bases de este proyecto para adaptar la base de código desarrollado y adaptarla de acuerdo a las necesidades y procesos de negocio, en orden de dar a sus usuarios un reporteador eficiente y confiable.

Conclusiones

Dentro del proyecto de desarrollo de software de la empresa, colaboré con el diseño y modelado de la base de datos, así como los procesos de carga, el diseño de la arquitectura de la aplicación, el desarrollo y codificación del proyecto, el mantenimiento de la aplicación y la mejora continua del mismo, apegándose a las normas de gestión de servicios de ITIL y la metodología de desarrollo en cascada.

Toda la experiencia que adquirí y las aportaciones que realicé dentro del proyecto, fueron hechas desde un marco de formación académica adquirida en la Facultad de Ingeniería de la Universidad Nacional Autónoma de México, cursando la Carrera de Ingeniería en Computación con módulo de salida de Ingeniería de software. Formación que permitió al autor desempeñarse de forma exitosa en cuanto a sus actividades y superar las expectativas esperadas en cuanto a los resultados de su trabajo. Así mismo, le permitió al autor proponer e implementar soluciones óptimas para el proyecto en cuanto a diseño y desarrollo de software.

Beneficios del proyecto para la organización

El proyecto desarrollado ayudó a más de 5 mil líderes y gerentes de ventas de la empresa con alrededor de 1 millón de representantes de ventas a su cargo en cuanto a sus procesos de gestión y toma de decisiones basadas en comparativas. Centralizando, ordenando y mostrando de forma ergonómica la información y los indicadores de sus progresos de ventas durante cada campaña en tiempo real, facilitando la comparación de los objetivos trazados para ellas. Además, ayudó a las áreas internas de negocio a identificar los problemas con los que se enfrenta el campo, en orden de dar seguimiento y apoyo en cuanto a la mejora de los procesos y traza de objetivos. Dentro de este proyecto pude desempeñar mis labores técnicas y administrativas para el progreso de la solución planteada.

Recomendaciones para futuros proyectos

De acuerdo a la experiencia técnica y administrativa que adquirí en torno a las áreas de desarrollo orientadas a ITIL y al desarrollo en cascada dentro de ambientes corporativos, se recomienda ampliamente la implementación de los siguientes puntos:

- Uso de tecnologías de desarrollo (bibliotecas, frameworks, gestores de versiones, etc) estándar en la industria para el fácil acceso a documentación, guías de uso y soporte dentro de la comunidad mundial de programación en Internet, además de confiabilidad y credibilidad de cada una de las herramientas.

- Uso adecuado de las tecnologías estándar implementadas en la empresa, como bases de datos licenciadas, repositorios globales de código, middlewares de comunicación, etc. En orden de explotar al máximo y de la forma más óptima los recursos con los que ya se cuenta en la empresa.
- Automatizar los procesos de desarrollo, construcción, pruebas, configuración y publicación de código, con el fin de evitar errores por descuido.
- Orientar completamente el desarrollo a las necesidades de negocio, en orden de construir software de mucha más relevancia para los objetivos de la empresa.
- Apegarse a las especificaciones marcadas por ITIL y la metodología de desarrollo que el equipo esté ocupando, en cuanto a reuniones, procedimientos, documentación y actividades. Lo cual facilitará la gestión del equipo de desarrollo y el software que construye, además de contar con una documentación y una calidad de los entregables mucho más alta.

Bibliografía

Elias, J. (13 de Abril de 2017). *Techtarget*. Obtenido de http://searchsqlserver.techtarget.com/definition/ACID

Fernández, C. (14 de Abril de 2017). *Dataprix*. Obtenido de http://www.dataprix.com/que-es-un-datawarehouse

Hannan, E., & Wilson, S. (11 de Abril de 2017). *Techtarget*. Obtenido de http://searchcloudstorage.techtarget.com/definition/RESTful-API

Hbernate. (16 de Abril de 2017). Obtenido de http://hibernate.org/orm/what-is-an-orm/

Hp. (22 de Abril de 2017). Obtenido de http://www8.hp.com/us/en/business-services/it-services.html?compURI=1143843#.WRKifXFOnIU

Kearn, M. (11 de Abril de 2017). *Microsoft*. Obtenido de https://blogs.msdn.microsoft.com/martinkearn/2015/01/05/introduction-to-rest-and-net-web-api/

Linthicum, D., & Giza, M. (8 de Abril de 2017). *Techtarget*. Obtenido de http://searchmicroservices.techtarget.com/definition/service-oriented-architecture-SOA

Margaret, R. (1 de Abril de 2017). *Techtarget*. Obtenido de http://searchitoperations.techtarget.com/definition/ITSM

Margaret, R. (1 de Abril de 2017). *Techtarget*. Obtenido de http://searchcio.techtarget.com/definition/ITIL-v3

Margaret, R. (4 de Abril de 2017). *Techtarget*. Obtenido de http://searchsoftwarequality.techtarget.com/definition/waterfall-model

Microsoft. (5 de Abril de 2017). Obtenido de https://msdn.microsoft.com/en-us/library/ee658098.aspx

Mullins, C., & Christiansen, S. (13 de Abril de 2017). *Techtarget*. Obtenido de http://searchsqlserver.techtarget.com/definition/database-management-system

Muycomputer. (10 de Abril de 2017). Obtenido de http://www.muycomputer.com/2014/01/13/que-es-dmz-dlink/

Naveen. (3 de Abril de 2017). *Testingfreak*. Obtenido de http://testingfreak.com/waterfall-model-software-testing-advantages-disadvantages-waterfall-model/

Oracle. (6 de Abril de 2017). Obtenido de https://docs.oracle.com/javase/tutorial/java/concepts/

Orafaq. (12 de Abril de 2017). Obtenido de http://www.orafaq.com/faq/what_are_the_difference_between_ddl_dml_and_dcl_commands

Orenstein, D. (11 de Abril de 2017). *Computerworld*. Obtenido de http://www.computerworld.com/article/2593623/app-development/application-programming-interface.html

Popyack, J., Zoski, P., & Salvage, J. (6 de Abril de 2017). *Drexel*. Obtenido de https://www.cs.drexel.edu/~introcs/Fa15/notes/06.1_OOP/Advantages.html?CurrentSlide=3

Richards, M. (9 de Abril de 2017). *Apple*. Obtenido de https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html

Richards, M. (8 de Abril de 2017). *Safaribooksonline*. Obtenido de https://www.safaribooksonline.com/library/view/software-architecture-patterns/9781491971437/ch01.html

Rouse, M. (15 de Abril de 2017). *Techtarget*. Obtenido de http://searchsqlserver.techtarget.com/definition/data-warehouse

Rouse, M. (13 de Abril de 2017). *Techtarget*. Obtenido de http://searchsqlserver.techtarget.com/definition/relational-database

Sinnexus. (15 de Abril de 2017). Obtenido de http://www.sinnexus.com/business_intelligence/datawarehouse.aspx

softwaretestingfundamentals. (3 de Abril de 2017). Obtenido de http://softwaretestingfundamentals.com/software-development-life-cycle/

Sourcemaking. (7 de Abril de 2017). Obtenido de https://sourcemaking.com/design_patterns

Tuprogramacion. (16 de Abril de 2017). Obtenido de http://www.tuprogramacion.com/glosario/que-es-un-orm/

Ucisa. (1 de Abril de 2017). Obtenido de https://www.ucisa.ac.uk/~/media/Files/members/activities/ITIL/service_operation/problem_management/ITIL_a%20guide%20to%20problem%20management%20pdf

Verma, E. (1 de Abril de 2017). *Simplilearn*. Obtenido de https://www.simplilearn.com/itsm-general-framework-and-implementation-rar291-article

Wayner, P. (8 de Abril de 2017). *Techbeacon*. Obtenido de https://techbeacon.com/top-5-software-architecture-patterns-how-make-right-choice

Acrónimos

- **IT (TI).-** Information Technology (Tecnologías de la Información).
- **NOLA.-** North & Latin America (Norte y Latino América).
- **ITSM.-** Information Technology Service Management (Manejo de los servicios de Tecnologías de la Información).
- **COBIT.-** Control Objectives for Information and related Technology (Control de Objetivos para para las Tecnologías relacionadas con la Información).
- **ITIL.-** Information Technology Infrastructure Library (Librería de infraestructura de Tecnologías de la Información).
- **MOF.-** Microsoft Operations Framework (Marco de Referencia de Operaciones de Microsoft).
- **HP.-** Hewlett-Packard.
- **CRM.-** Customer Relationship Management (Manegador de las relaciones con el cliente).
- **FTP.-** File Transfer Protocol (Protocolo de transferencia de archivos).
- **SFTP.-** Secure File Transfer Protocol (Protocolo de transferencia de archivos seguro).
- **MVC.-** Model View Controller (Modelo Vista Controlador).
- **DMZ.-** Demilitarized Zone (Zona desmilitarizada).
- **DBMS.-** Data Base Management System (Sistema Manejador de Base de Datos).
- **RDBMS.-** Relational Data Base Management System (Sistema Manejador de Base de Datos Relacionales).
- **ORDBMS.-** Object Relational Data Base Management System (Sistema Manejador de Base de Datos Objeto Relacional).
- **SQL.-** Standard Query Language (Lenguaje estadar de consulta).
- **DDL.-** Data Definition Language (Lenguaje de Definición de Datos).
- **DML.-** Data Manipulation Language (Lenguaje de Manipulación de Datos).
- **DCL.-** Data Control Language (Lenguaje de Control de Datos).
- **TCL.-** Transaction Control Language (Lenguaje de control de Transacciones).
- **ETL.-** Extract Transform Load (Extracción Transformación Carga).
- **SE.-** Standard Edition (Edición Estandar).
- **EE.-** Enterprise Edition (Edición Empresarial).
- **SOAP.-** Simple Object Access Protocol (Protocolo de Acceso Simple de Objeto).
- **REST.-** Representational State Transfer (Transferencia de estado representacional).
- **RESTful.-** Representational State Transfer ful (Servicios basados en el protocol Transferencia de estado representacional).

- **IDE.-** Integrated Development Environment (Ambiente de Desarrollo Integrado).
- **ORM.-** Object Relational Mapping (Mapeo de Objeto Relacional).
- **HTTP.-** Hypertext Transfer Protocol (Protocolo de Transferencia de HyperTexto).
- **HTTPS.-** Hypertext Transfer Protocol Secure (Protocolo Seguro de Transferencia de HyperTexto).
- **XML.-** eXtensible Markup Language (Lenguaje de Marcas Extensibles).
- **HTML.-** HyperText Markup Language (Lenguaje de Marcas de Hyper Texto).
- **JSON.-** JavaScript Object Notation (Notación de Objetos de Javascript).
- **API.-** Aplication Programming Interface (Interfaz de programación de Aplicación).
- **URL.**- Uniform Resources Locator (Localizador de Recursos Uniforme).
- **URI.-** Uniform Resources Identifier (Identificador de Recursos Uniforme).