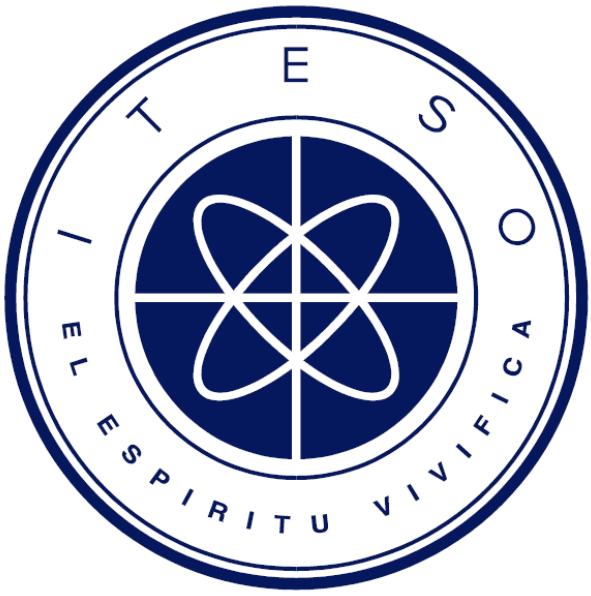
INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

MAESTRÍA EN INFORMÁTICA APLICADA

Reconocimiento de Validez Oficial de Estudios de Nivel Superior según acuerdo secretarial 15018, publicado en el
Diario Oficial de la Federación el 29 de Noviembre de 1976.



Automatización de Tareas y Errores Comunes en Bases de Datos de una Empresa de Telecomunicaciones

Trabajo recepcional que para obtener el título de

Maestro en Informática Aplicada

Presenta:
LTI José Francisco Solís Aguirre

Asesor:
Dr. Jorge Arturo Pardiñas Mir

Tlaquepaque, Jalisco 8 Junio de 2019

I. Tabla de contenidos

II. Índice de tablas

III. Índice de Figuras

Resumen

La utilización de las tecnologías de información como herramienta de apoyo directo a la visión y misión de una empresa no es nada nuevo hoy en día. Este proyecto presenta una iniciativa en la que se utilizan recursos existentes de tecnología para la mejora de procesos y ahorro de tiempo en un ambiente laboral, sin ningún compromiso económico adicional por parte de las empresas involucradas.

Las empresas en cuestión son *B&C Telco*, una de las empresas de telefonía más grandes en el mundo, que últimamente ha incursionado en la televisión de paga, IPTV, la industria del entretenimiento entre otras, y *Swift Solutions*, empresa líder proveedora de soporte, software y servicios a empresas de telefonía.

Tomando en cuenta la tendencia global de las empresas en la actualidad, así como una petición formal por parte de *B&C Telco*, ambas empresas emprenden el viaje de adopción de DevOps, lo cual fue parcialmente el motivo de la puesta en marcha de este proyecto. El objetivo principal es el de automatizar una serie de tareas, corrección de errores y cuantificaciones principalmente relacionados con el manejo de suscriptores y la integridad de las operaciones para utilizar el tiempo invertido en la investigación de problemas más complejos.

Trayendo consigo un conjunto de lecciones aprendidas, historias de éxito y áreas de mejora, el proyecto es considerado un éxito, tomando en cuenta las consideraciones que serán elaboradas a lo largo de este documento.

Palabras clave

- DevOps
- Automatizacion
- Linux
- Base de datos

Introducción

Este documento pretende presentar, de una manera coherente y concisa la realización de un proyecto de automatización en la empresa *B&C Telco*, a través de un equipo de trabajo de la empresa *Swift Solutions*, la cual provee servicios de soporte y desarrollo de aplicaciones para la empresa *B&C Telco*. *B&C Telco* es actualmente una de las compañías telefónicas más grandes del mundo.

El requerimiento principal de este proyecto por parte de *B&C Telco*, la empresa cliente, y de la empresa para la cual laboro, *Swift Solutions* fue, principalmente, el de adoptar un esquema de trabajo apegado a la metodología de DevOps. Es por ello que la mayor parte del documento se enfocará en la automatización, aspecto importante de esta metodología. Adicionalmente, esta sección describirá de una manera más legible los conceptos técnicos necesarios para comprender este reporte de proyecto y sus conclusiones.

Si bien el concepto de DevOps no tiene en la actualidad una definición bien establecida, la realización de este proyecto encamina de manera positiva a *Swift Solutions* a su reciente campaña de adoptar esta metodología de trabajo. Para *Swift Solutions* es de suma importancia adoptar esta metodología de trabajo, ya que como será explicado posteriormente, fue uno de los requerimientos por parte de *B&C Telco* para seguir trabajando con *Swift Solutions* como empresa proveedora de servicios, software y soporte.

Es importante mencionar que, debido a motivos de confidencialidad, los nombres de las compañías involucradas en este proyecto han sido cambiados y se utilizarán nombres ficticios a lo largo de este documento. Esto aplica para ambas empresas involucradas; *B&C Telco* y *Swift Solutions*.

A continuación, se presenta una breve descripción de los capítulos contenidos en este documento:

Capitulo I. Marco de Referencia - En este capítulo se definen los conceptos teóricos y la relación entre ellos que deben ser conocidos por el lector para una mejor comprensión del resto de este reporte.

Capítulo II. Descripción del Proyecto - En este capítulo se establece el entorno del proyecto, las empresas involucradas, la situación a la que se enfrentan, el contexto del ambiente del ramo, así como del equipo de trabajo. Además de esto, se define de una manera clara el problema, su justificación, el alcance y los objetivos del proyecto, la descripción de las actividades realizadas y los resultados obtenidos.

Capítulo III. Análisis del Proyecto - En este capítulo se analiza el proyecto desde diferentes puntos de vista, como lo son el del equipo de trabajo y las empresas involucradas. Adicionalmente se identifican las lecciones aprendidas, áreas de mejora y se presentan las conclusiones del proyecto.

Capítulo 1. Marco de referencia

Como la mayoría de las empresas en la actualidad, la gran mayoría de los servidores que *B&C Telco* utiliza en sus sistemas de producción tienen Linux como sistema operativo. Existe un número pequeño de servidores de Windows y otros sistemas operativos, sin embargo, los servidores utilizados por el equipo de trabajo que realiza este proyecto son en su totalidad de Linux.

Estos servidores Linux almacenan los scripts escritos para cada actividad y error a corregir, además de tener conectividad a la red de *B&C Telco*, que es en donde se almacenan las bases de datos de producción de todos sus suscriptores, servicios y transacciones. Los scripts fueron escritos para la *Shell bash*, debido a su alta versatilidad y la posibilidad de automatizar algunas tareas mediante el *daemon* de Linux, Cron.

Cabe destacar que la elección de proveedor de base de datos a utilizar en el software de facturación desarrollado por *Swift Solutions* fue decisión de la misma empresa, y no de *B&C Telco*. Por razones obvias, las bases de datos están almacenadas en la red e infraestructura de *B&C Telco*, y no en *Swift Solutions*. *Swift Solutions* eligió a *Oracle* como proveedor para la solución de CRM que provee a *B&C Telco*.

La base de datos de *Oracle* es altamente robusta y adaptable, y provee características que no solo proporcionan el mejor rendimiento en un ambiente de producción, sino también ofrecen una alta compatibilidad con una variedad de software existente, así como una alta seguridad e integridad de los datos. Oracle es excepcionalmente capaz de llevar la carga de millones de transacciones simultáneas, manejando un alto volumen de datos y conservando la integridad de la información almacenada.

Volviendo entonces al aspecto técnico de este proyecto, los scripts de *bash* que fueron escritos para la corrección de errores y actividades llaman directamente a un conjunto de *Stored Procedures* que hacen interfaz directamente con la base de datos y la información contenida en la misma. El razonamiento detrás de la decisión de tener un conjunto de *scripts* de *bash* y *Stored Procedures* viene de la versatilidad de automatización de scripts en Linux mediante Cron y la alta manejabilidad de datos e interfaz que un *Stored Procedure*, mediante bloques de lenguaje SQL, tiene con la base de datos de *Oracle*; la combinación de ambas resulta en una solución robusta, adecuada a las necesidades del equipo y la empresa.

Habiendo descrito de una manera muy general las herramientas utilizadas y el proyecto realizado, en las siguientes páginas se definirán de una manera más extensa todos los conceptos arriba mencionados.

1.1 DevOps

El concepto DevOps proviene de la combinación de las palabras *development* (desarrollo de software) y *operations* (operaciones) las cuales nos dan una pista de la idea básica detrás de este movimiento.

El origen de la palabra se le puede atribuir a Patrick Debois, quien labora como desarrollador de software en 2009 y se encuentra con que la aparente desconexión entre el personal de desarrollo y el de operaciones es causa de muchos problemas y frustración en las empresas actuales. Debido a esto, inició conferencias llamadas *devopsdays* para intentar atraer más atención de la industria al problema.

DevOps tiene sus orígenes en los principios de la metodología de desarrollo de software *agile*, la cual establece que los individuos y su interacción tienen precedencia sobre procesos y herramientas. Es por ello que se puede decir que DevOps enfatiza la interacción entre los diferentes miembros de la organización, mejorando y estandarizando herramientas para disminuir en lo más posible la separación entre el personal de desarrollo y el de operaciones,

agilizando su comunicación, mejorando el tiempo y frecuencia de las entregas y, por ende, reduciendo el tiempo de desarrollo de software.

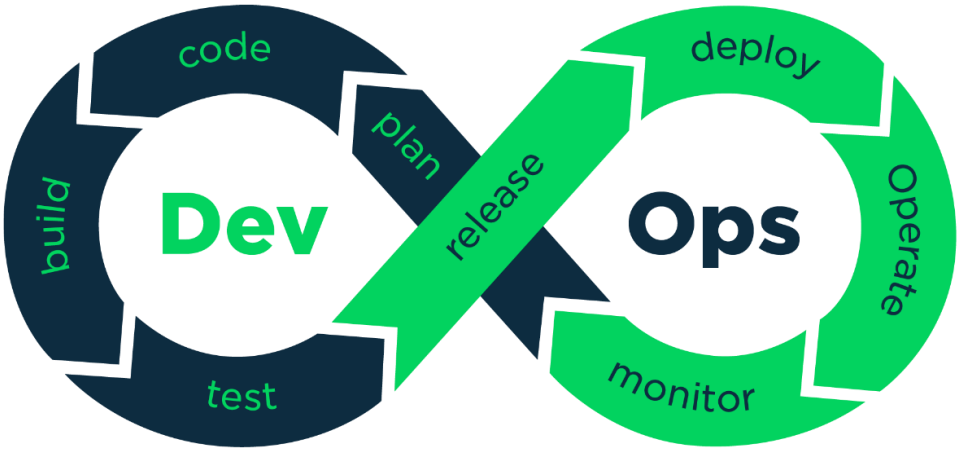


Figura 0.1 Representación gráfica de esquema de DevOps. Tomado de (https://www.quickscrum.com/)

DevOps describe un mundo en donde los desarrolladores, ingenieros de calidad, ingenieros de soporte y administradores de sistema trabajan de una manera más cercana que en cualquier ambiente tradicional. A pesar de que DevOps es reconocido como un acelerador del rápido desarrollo de software y la automatización, otro de los beneficios importantes que brinda es el de la rápida solución de problemas que ocurren cuando todos los equipos pueden colaborar en el *troubleshooting*.

Desafortunadamente, los desarrolladores, ingenieros de calidad, ingenieros de soporte y administradores de sistemas tienen limitaciones en sus habilidades de *troubleshooting* y a menudo culpan a otros equipos por los problemas que atraviesan. Uno de los objetivos de DevOps es el de reducir lo más posible estas limitaciones al fomentar la cooperación entre equipos.

Naturalmente, dada la ya mencionada prioridad de DevOps de fomentar la interacción entre equipos, la automatización es un aspecto fundamental para este movimiento. Después de todo,

la automatización de tareas repetitivas o tediosas libera tiempo valioso que puede ser mejor utilizado en interacción entre equipos.

1.2 Base de datos relacional

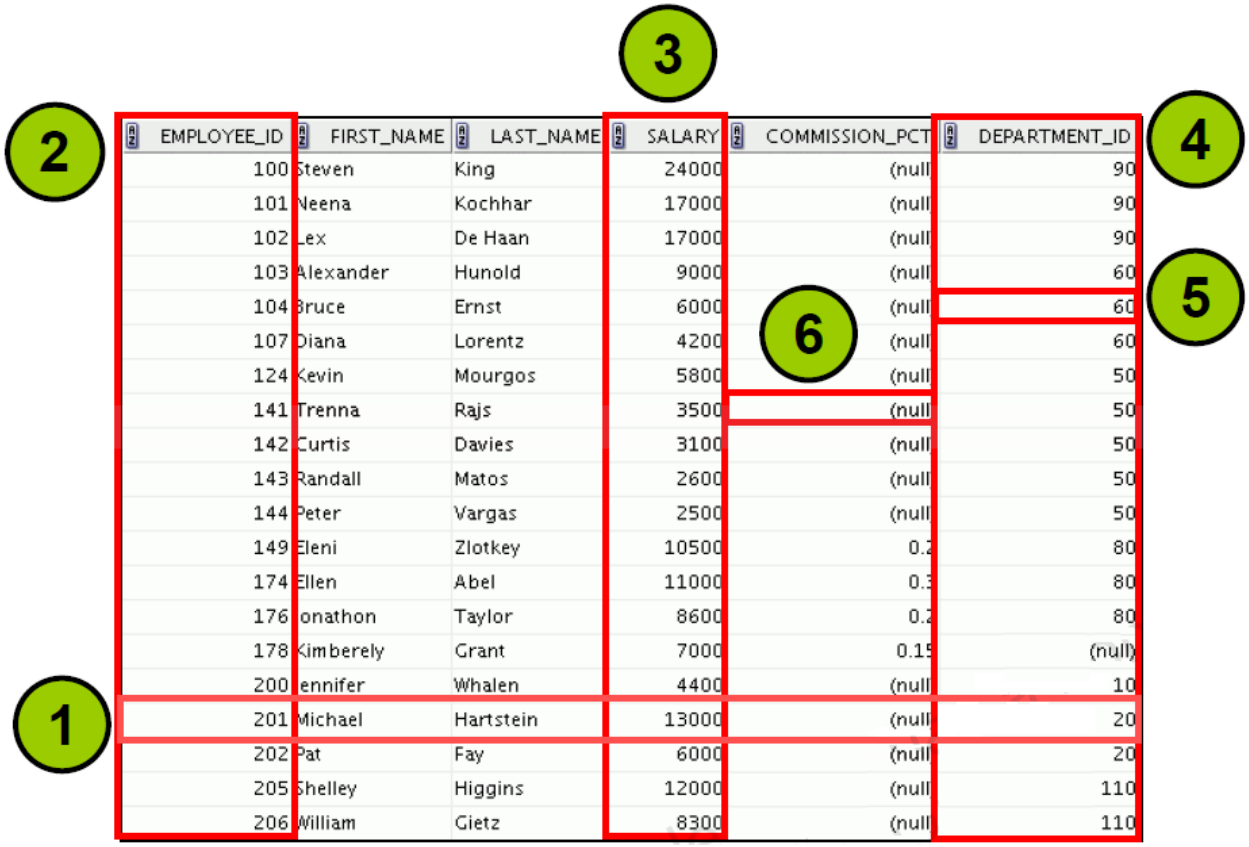
Toda organización posee información que necesita almacenar y administrar para cumplir con sus funciones. En tiempo pasado la información se almacenaba en archiveros y carpetas, pero hoy en día la mayoría de las organizaciones y empresas almacena su información en bases de datos relacionales. Hay diferentes modelos de bases de datos, sin embargo, hoy en día el modelo más aceptado es el relacional.

Una base de datos relacional contiene los siguientes aspectos:

- Estructuras: Objetos bien definidos proveen de almacenamiento y acceso a los datos en la base de datos.
- Operaciones: Acciones definidas permiten a usuarios y aplicaciones manipular la información contenida en la base de datos.
- Reglas de integridad: Dichas reglas gobiernan las operaciones en los datos y estructuras de la base de datos.

Visto de una manera general, suficiente para fines de comprensión de este documento, una base de datos relacional se compone por los siguientes elementos:

- Tablas – La información es almacenada en Tablas, las cuales a su vez se componen por columnas y filas.
- Columnas: Cada tabla es compuesta por una o más columnas. Cada columna es definida desde su creación con un tipo de datos, el cual puede ser de texto o numérico y puede o no contener valores nulos.
- Filas: Cada fila en la tabla representa un registro en la base de datos.



EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	COMMISSION_PCT	DEPARTMENT_ID
100	Steven	King	24000	(null	90
101	Neena	Kochhar	17000	(null	90
102	Lex	De Haan	17000	(null	90
103	Alexander	Hunold	9000	(null	60
104	Bruce	Ernst	6000	(null	60
107	Diana	Lorentz	4200	(null	60
124	Kevin	Mourgos	5800	(null	50
141	Trenna	Rajs	3500	(null	50
142	Curtis	Davies	3100	(null	50
143	Randall	Matos	2600	(null	50
144	Peter	Vargas	2500	(null	50
149	Eleni	Zlotkey	10500	0.2	80
174	Ellen	Abel	11000	0.3	80
176	Jonathon	Taylor	8600	0.2	80
178	Kimberely	Grant	7000	0.15	(null)
200	Jennifer	Whalen	4400	(null	10
201	Michael	Hartstein	13000	(null	20
202	Pat	Fay	6000	(null	20
205	Shelley	Higgins	12000	(null	110
206	William	Gietz	8300	(null	110

Figura 0.2 Ejemplo gráfico de una tabla en una base de datos. Tomado de (https://docs.oracle.com)

1. Una fila representa toda la información que en este caso pertenece a un empleado en particular.
2. Una columna que en este caso contiene la llave primaria de la tabla, el ID de empleado. Este ID se puede utilizar para realizar comparaciones con otras tablas que hagan uso de este ID.
3. Una columna que no contiene un valor clave. En este caso, representa el salario del empleado.
4. La columna de ID de departamento, la cual es un valor clave, o en otras palabras una llave foránea que hace referencia a otra tabla de la base de datos.
5. Un campo de la tabla con un valor numérico. Un campo se encuentra en la intersección de una columna y una fila.
6. Un campo que no contiene información, y por ende tiene un valor nulo.

1.3 SQL

SQL es el *Structured Query Language* por sus siglas en inglés y es el lenguaje de manipulación de base de datos definido por la ANSI. SQL es eficiente, fácil de aprender y utilizar, y además es un lenguaje funcionalmente completo, ya que permite realizar todas las operaciones necesarias en el manejo de una base de datos.

El lenguaje SQL es un conjunto de sentencias con el cual todos los programas y usuarios de la base de datos realizan operaciones y acceden a los datos almacenados en la misma. Los programas y aplicaciones, así como las herramientas de Oracle usualmente permiten a los usuarios acceder a la base de datos sin la utilización de SQL, sin embargo, internamente todas estas herramientas utilizan sentencias y bloques de SQL para realizar las operaciones en la base de datos.

SQL provee sentencias para una variedad de tareas:

- Consulta de datos
- Insertar, actualizar y borrar registros en la base de datos.
- Crear, reemplazar y editar objetos de la base de datos.
- Controlar acceso a la base de datos y sus objetos.

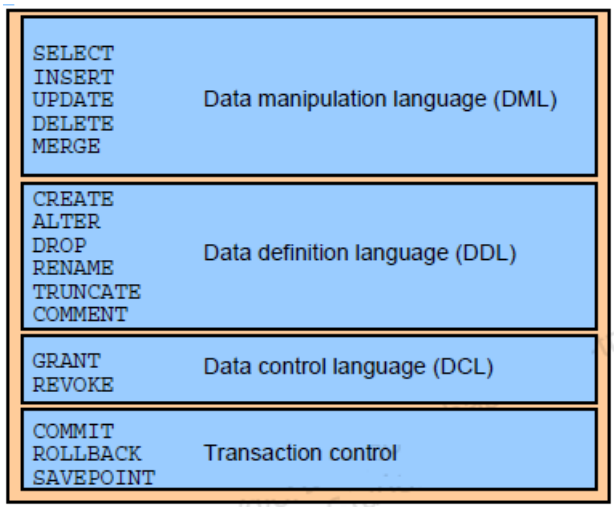


Figura 0.3 Ejemplos de sentencias de SQL. Tomado de (https://docs.oracle.com)

1.4 Stored Procedure

En la base de datos de Oracle, un *Stored Procedure* es un bloque de código en lenguaje PL/SQL que puede contener funciones, procedimientos o bloques de SQL, y pudieran ser vistos como una parte o bloque de una aplicación más grande; precisamente como la aplicación en la que se basa este proyecto.

La utilización de esta herramienta fue crítica para este proyecto, dado que todas las actividades o errores que componen este proyecto contienen uno o más *stored procedures* que se encargan de realizar las actividades de manipulación de datos necesarias para la corrección de los errores en la base de datos.

1.5 Sistema operativo (LINUX)

El Sistema Operativo de un dispositivo es el software primario que administra todo el hardware y software de dicho dispositivo, que en el caso del proyecto siempre es una computadora o un servidor. Se encarga de hacer interface con el hardware y provee servicios que las diferentes aplicaciones pueden utilizar.

Linux inició como una versión gratuita del *kernel* de UNIX, y es un sistema operativo diseñado con multitarea como beneficio principal. UNIX/Linux es actualmente uno de los sistemas operativos más populares del mundo, y la primera opción para la mayoría de las aplicaciones empresariales de hoy en día. Es por ello que este proyecto se aplicó en servidores Linux, apoyándonos de las herramientas que este extenso sistema operativo nos provee.

La variante de Linux más utilizada en ambientes de producción alrededor del mundo es la distribución de *Redhat*. La popularidad de *Redhat* se basa en el modelo de negocios que han adoptado, así como la alta disponibilidad y compatibilidad de software y hardware que se ha desarrollado a su alrededor. *Redhat* provee contratos de soporte empresarial con diversos SLAs (acuerdo de nivel de servicio), que provee a las empresas con una propuesta de valor en caso de algún error fatal. La alta adopción a nivel mundial de esta distribución de Linux también ha traído

consigo una compatibilidad de hardware y software que pocas distribuciones pueden ofrecer. Este proyecto ha sido basado en esta distribución de Linux.

1.6 Shell de Linux (BASH)

La *shell* es un programa de usuario especial que provee de un interfaz al usuario para utilizar el sistema operativo. La *shell* acepta comandos legibles para humanos que después traduce e interpreta en un lenguaje que el *kernel* pueda entender. Es un intérprete de comandos que ejecuta instrucciones que pueden ser introducidas por medio de dispositivos de entrada, como un teclado o por medio de archivos.

La *shell* es independiente del sistema operativo subyacente, lo cual ha fomentado el desarrollo de numerosas *shell*, de las cuales una de las más populares actualmente es bash, o *Bourne Again Shell.* Esta *shell* es la que fue utilizada en este proyecto para la automatización de tareas y corrección de errores.

1.7 Script

Un *script* de *bash* es un bloque de comandos contenidos en un archivo ejecutable dentro de algún directorio de una computadora o servidor cuyo sistema operativo es Linux en este caso. El grupo de comandos incluidos en un *script* pueden ser ejecutados directamente desde la línea de comandos, sin embargo, el contenerlos dentro de un archivo de *script* nos brinda una ventaja importante: la posibilidad de automatización de dichos comandos al combinar el *script* con alguna de las herramientas de automatización que Linux nos provee.

Los *scripts* trabajan muy de cerca con la *shell* para la cual fueron escritos. En este proyecto se utilizaron *scripts* de *bash*, y se hizo uso de la funcionalidad de *bash* de redirigir la salida de los comandos para así implementar la funcionalidad de *logging* por actividad y el *logging* general del *script* maestro.

1.8 Cron

Cron es un *daemon* de Linux que permite a usuarios de Linux ejecutar comandos o *scripts* automáticamente en fechas y horas previamente especificadas. Es una herramienta destinada principalmente para administradores de Linux, pero su alta versatilidad permite su uso en una variedad de escenarios. Es precisamente por esta versatilidad por lo cual nos apoyamos de esta herramienta para automatizar las tareas de este proyecto.

Para demostrar de una manera breve la alta versatilidad de cron es necesario mostrar los cinco elementos que utiliza para agendar tareas en tiempos y fechas específicos. Cron permite agendar tareas en cada minuto de la hora, cada hora del día, cada día del mes, cada mes del año y cada día de la semana. Esto permite combinaciones interesantes, por ejemplo, la ejecución de una tarea cada Domingo, cada primer día del mes o en el minuto 20 de cada hora. Además de estos cinco elementos, se permite la utilización de palabras clave, que permiten la ejecución de tareas en estados específicos del sistema operativo; por ejemplo, se puede agendar la ejecución de una tarea después de un reinicio del sistema.

```
# ┌──────────── min (0 - 59)
# │ ┌────────── hour (0 - 23)
# │ │ ┌──────── day of month (1 - 31)
# │ │ │ ┌────── month (1 - 12)
# │ │ │ │ ┌──── day of week (0 - 6) (0 to 6 are Sunday to
# │ │ │ │ │     Saturday, or use names; 7 is also Sunday)
# │ │ │ │ │
# │ │ │ │ │
# * * * * *  command to execute
```

Figura 0.4 Estructura de archivo de configuración de Cron. Tomado de (https://www.ostechnix.com/)

1.9 Bug o error

Un *bug* es un error, imperfección o falla en un software que causa que el mismo produzca un resultado incorrecto o inesperado. Estos errores van de la mano con el software, y a pesar de los esfuerzos que diversos profesionales y autores de software han dedicado a su erradicación, siguen existiendo. Es por esto que además de las actividades y metodologías preventivas aplicadas durante el desarrollo del software, es necesario contar con planes de mejora continua y corrección de los mismos.

En desarrollos de software muy extensos es común contar con un repositorio de bugs o errores conocidos. En muchas ocasiones, estos errores pueden no ser rápidos de corregir, y es una realidad que, en ocasiones, el mismo error conocido puede estar presente en las aplicaciones durante meses, si no es que años. Es por ello que uno de los objetivos de este proyecto es el de corregir en medida de lo posible, la mayor cantidad de errores causados por *bugs* conocidos de una manera ágil y eficaz.

Capítulo 2. Descripción del proyecto

2.1 Antecedentes

El proyecto se lleva a cabo en una empresa proveedora de servicios de *Customer Relationship Management* (CRM) y facturación para empresas de telecomunicaciones, empresa para la cual laboro como líder de este proyecto. Dicha empresa se llama *Swift Solutions* y es reconocida como líder en el ramo de la prestación de servicios a empresas de telecomunicaciones y posee una cartera de clientes importante. Estos clientes tienen sede en diversos países del mundo. El proyecto fue aplicado directamente en los servidores y bases de datos de uno de sus clientes, *B&C Telco* y el papel que desempeñé en el mismo fue de líder, con un par de compañeros apoyándome con las pruebas.

Como muchas empresas al momento de la redacción de este documento, la empresa proveedora de servicios en cuestión estaba llevando a cabo una iniciativa muy fuerte para transformar sus operaciones de un esquema tradicional a uno impulsado en su totalidad por el movimiento *DevOps*. El esquema de trabajo de *DevOps* propone un esquema de trabajo en el cual el desarrollo de software (*Development*) y las operaciones de IT (IT *Operations*) trabajen conjuntamente durante todo el ciclo de vida de desarrollo, agilizando el proceso de mejoras y correcciones, y a su vez obteniendo una mejor alineación de los equipos de trabajo con los objetivos de la empresa.

Dada una necesidad interna del equipo, la cual elaboraré a continuación, y la intención de *Swift Solutions* de moverse a este esquema de trabajo, la idea inicial del proyecto es establecida. Si bien, el proyecto en sí no se apega en su totalidad al esquema DevOps como se define hoy en día, sí cubre uno de sus aspectos importantes: la automatización.

2.1.1 B&C Telco

Importante empresa de telecomunicaciones a nivel mundial, con presencia principalmente en el continente americano. Inició como una empresa de telefonía fija a principios del siglo XIX y con la llegada de la telefonía celular, incursionó en este mercado y se convirtió en una de las compañías de telecomunicaciones más grande del mundo.

Con el rápido cambio del mercado que se ha dado desde finales de la década de los 90, se ha visto obligada a diversificarse e incursionar en nuevos mercados, como lo son la televisión por cable, canales de televisión de paga, televisión por IP, telefonía IP de nivel consumidor y empresarial, entre otras. Su rápida expansión ha sido posible gracias a una serie de estrategias que incluyen crecimiento propio, así como diversas adquisiciones de empresas que anteriormente proveían estos servicios.

2.1.2 Swift Solutions

Swift Solutions es la empresa para la cual laboro y en donde se llevó a cabo este proyecto. Es una empresa transnacional líder en el ramo de desarrollo de software y servicios para empresas de telecomunicaciones. De la misma manera que *B&C Telco*, *Swift Solutions* ha también sentido las consecuencias del cambiante mundo de las telecomunicaciones y el entretenimiento, y por ello también ha intentado diversificarse. No obstante, la mayor fuente de ingreso de la compañía son sus soluciones de facturación y gestión de clientes (CRM) que provee a empresas de telecomunicaciones en todo el planeta.

Swift Solutions es una empresa que, al igual que sus clientes, enfrentan un panorama rápidamente cambiante y necesita adaptarse rápidamente a todos estos cambios. La naturaleza de la relación de *Swift Solutions* con sus clientes hace que esta capacidad de adaptarse sea aún más retadora, ya que además de sus propios retos y problemas internos, también debe sobrellevar los problemas y exigencias de sus clientes.

2.1.3 La relación entre B&C Telco y Swift Solutions

B&C Telco es el cliente más grande de *Swift Solutions* y si bien no son la misma empresa, esto causa que ambas empresas trabajen de una manera muy cercana lo cual resulta en una interacción simbiótica entre ambas. Un ejemplo de ello son las consecuencias directas que los resultados de *B&C Telco* tienen sobre *Swift Solutions*. Dado el mercado rápidamente cambiante, *B&C Telco* ha sufrido pérdidas importantes en las divisiones con menos crecimiento, como lo es el sector de la televisión de paga.

Las consecuencias de esto son sentidas de una manera muy real en *Swift Solutions* y, como fue mencionado anteriormente, se esparcen de manera muy rápida a través de toda la jerarquía de *Swift Solutions*. Estas consecuencias pueden englobar desde reorganizaciones fuertes y recortes de personal, hasta cambios importantes en la filosofía y metodología de trabajo de la empresa a petición de *B&C Telco*.

Tal es la influencia de *B&C Telco* sobre *Swift Solutions*, que solo bastó la petición de B&C para que *Swift Solutions* adoptara la metodología de DevOps, para que *Swift Solutions* aprobara el cambio y estableciera un plan de rápida adopción.

2.1.4 Problema o necesidad

El equipo al que pertenezco y donde el proyecto fue aplicado provee soporte a las operaciones diarias de la base de datos de la empresa de telecomunicaciones cliente, *B&C Telco*. Las tareas que realizamos son varias, pero se pueden dividir en tres grupos principales: El primero engloba las correcciones diarias a las bases de datos, como lo son la inconsistencia de información, errores en tablas y transacciones. La mayoría de estos errores son comunes y no requieren de mucha investigación o actividades complejas siempre y cuando se deriven de problemas conocidos del software. El segundo grupo de actividades consiste en la cuantificación de clientes afectados por dichos errores al término de cada ciclo de facturación.

El tercer grupo de actividades involucra la solución e investigación de problemas específicos, generalmente solicitados mediante peticiones de servicio o *service requests*. La investigación y solución de estos incidentes requiere de una mayor inversión de tiempo. En muchas ocasiones el problema puede afectar a un solo cliente o porcentaje pequeño de la población del sistema afectado por alguno de los errores mencionados en el párrafo anterior, causado por alguna actividad o mantenimiento extraordinario que se les haya realizado a los clientes en cuestión. Es por esto que esta actividad no se puede automatizar.

Al igual que otros softwares de esta magnitud, el software proporcionado por *Swift Solutions* contiene una variedad de problemas conocidos, y nuevos problemas se descubren día con día. A estos problemas se les conoce como *bugs*, problemas conocidos o *known issues*. Una de las necesidades por parte de la unidad de negocio es la de poder discernir qué clientes están siendo afectados por problemas conocidos, operaciones de mantenimiento mal ejecutadas o transacciones fallidas.

Dada la ajetreada naturaleza de las operaciones diarias de una empresa de telecomunicaciones de esta magnitud, y los niveles de acuerdo de servicio (SLA) estrictos, el tiempo invertido en la corrección de errores comunes podría ser mejor aprovechado en otras actividades.

2.1.5 Justificación

Empezando por el aspecto de crecimiento personal y profesional, este proyecto fue para mí muy deseable. El poder aportar en un ambiente empresarial y de producción, de tal manera que mis ideas y desarrollos influyan positivamente en mi trabajo y el de los demás, es algo que me interesó desde un principio. Si le aumentamos a esto que el proyecto incluye cierto grado de automatización y ahorro de tiempo, la decisión fue fácil.

En términos de *Swift Solutions*, el proyecto desde su concepción demostró ser altamente justificable debido a la combinación de los siguientes factores:

- La realización del proyecto se alinea de una manera satisfactoria con la filosofía actual de la empresa.
- La exitosa automatización de actividades diarias libera un porcentaje de tiempo de cada ingeniero el cual se puede aprovechar mejor en otras tareas.
- Dicha automatización es aplicada, además de la corrección de errores comunes, a la cuantificación de clientes que son afectados a su vez por estos errores, información que es requerida de manera diaria por la unidad de negocio en cada ciclo de facturación.
- El poder realizar más actividades en una menor cantidad de tiempo es algo que naturalmente le interesa tanto a *B&C Telco* como a la administración y directivos de *Swift Solutions*.
- Como beneficio adicional, se aprecia una mejor (mas no total) claridad con respecto al origen de los errores que afectan a los clientes. Se puede inferir, en algunos casos, qué fue lo que causó la falla de un proceso en específico.

2.1.6 Alcance

Se definió como alcance de proyecto la creación de *scripts* de *bash* y *stored procedures* a ser ejecutados en SQL*PLUS para la corrección de los errores más comunes de la operación de la base de datos de nuestro cliente de telecomunicaciones, *B&C Telco*. Dichos *scripts* estarían almacenados directamente en los servidores host de las bases de datos. La disposición exacta de dichos *scripts* no fue definida propiamente desde un inicio, pero se calculaba que se iba a hacer uso de *scripts* de *bash*, *stored procedures* y la funcionalidad del cron de Linux.

Además de la corrección de errores en la base de datos, se definió también la automatización de las cuantificaciones de clientes que son afectados por los errores conocidos de más alto perfil. Ningún esfuerzo se realizó para automatizar o definir un proceso de trabajo para la investigación de errores más específicos, dada la naturaleza compleja de dichas investigaciones.

2.2 Objetivos del proyecto

2.2.1 Objetivo general

La automatización de las tareas y errores comunes en la base de datos de una empresa de telecomunicaciones, para de esta manera liberar el tiempo invertido en actividades repetitivas y utilizarlo en la investigación de errores más complejos.

2.2.2 Objetivos específicos

- Alinearnos como equipo con la dirección actual de *Swift Solutions* al adoptar un esquema más apegado a la metodología de trabajo que la empresa ha adoptado.
- Liberar el tiempo utilizado en la corrección y cuantificación de errores comunes de la base de datos originados de la operación diaria de la empresa de una manera que va de lo semi-automatizado a completamente automatizado, dependiendo de la complejidad del error en específico.
- Automatizar la mayor cantidad de tareas posibles mediante un sistema de prueba y error. En caso de las tareas que no se puedan automatizar, determinar la causa.

- Crear un sistema de tablas de respaldo estandarizado, para poder realizar un mejor rastreo de los cambios realizados en las tablas de producción, así como proveer la disponibilidad de un banco de información de respaldo con el cual se pueda reconstruir la información modificada en las tablas de producción en caso de alguna falla en las tareas.
- Aprovechar el tiempo ahorrado para la investigación de errores más complejos que usualmente involucran grupos reducidos de clientes y requieren una investigación más a fondo.
- Proveer una mayor claridad al momento de identificar qué errores afectan a los clientes que no pudieron ser corregidos mediante las tareas automatizadas.

2.3 Metodología

En un inicio, se definió un grupo de tareas pequeño para automatizar. El objetivo era determinar si la automatización era viable y los resultados deseables. Una vez automatizadas las tareas iniciales se continuó añadiendo tareas adicionales a medida que las anteriores fueran automatizadas.

Cabe destacar que no hubo un orden lógico en lo que respecta a la inclusión de nuevas tareas. Al momento de observar la conveniencia y éxito de las tareas automatizadas en un inicio, se hizo aparente la viabilidad de la automatización de más tareas, aún algunas que no precisamente involucran la solución de algún error, como las cuantificaciones que fueron añadidas durante la parte final del proyecto.

Una vez determinadas las actividades y errores a automatizar, se comenzó a recolectar toda la información acerca de los mismos. Esta información consiste desde información básica hasta *scripts* o *stored procedures* preexistentes al proyecto.

Dada mi experiencia previa en otras compañías, así como la disponibilidad de algunos *scripts* preexistentes, se optó por la creación de uno o más *scripts* por actividad, acompañados por *stored procedures*. Entonces, en su forma más básica, cada error o tarea está compuesto por dos

o más archivos: el primero es el *script* de *bash*, que es llamado por el sistema operativo al momento de ser ejecutado, ya sea manualmente o por medio de cron. Este *script* a su vez ejecuta uno o más *stored procedures* que realizan la modificación de datos directamente en la base de datos.

Aún en los casos en los que ya se contaba con un *script* o *stored procedure* preexistente, se tuvieron que hacer modificaciones extensas a los mismos. Estas modificaciones incluían cambios en el SQL o en las instrucciones del *script*, la inclusión de más variables, parámetros posicionales, etc., para garantizar su correcta ejecución de manera automatizada, en otras palabras, sin intervención humana.

Una vez que se cuenta con el *script* de *bash* y sus correspondientes *stored procedures*, se procede a añadirlo al *script* maestro, el cual se encarga de consolidar todas las tareas en un solo script, así como realizar el *logging* de actividades.

Una vez que el número de tareas y errores rebasó cierto umbral, se optó por crear un *script* maestro para acelerar el proceso de ejecución de las distintas tareas y así no requerir al usuario a familiarizarse con la estructura de directorios de todos los *scripts* que forman parte del proyecto. De esta manera se puede ejecutar cualquier *script* de manera rápida sin necesidad de navegar al directorio donde se encuentren los archivos relevantes para la ejecución de cada tarea.

Además de un conjunto de retos que fueron revelándose a lo largo del proyecto, como lo fueron la falta de un sistema de respaldo o de un sistema de rastreo de actividad, los cuales serán explicados en la sección de descripción de actividades, el siguiente punto central del proyecto fue la inclusión de las cuantificaciones diarias en el proyecto de automatización. Automatizar las cuantificaciones fue uno de los aspectos que más tiempo ahorra a cada ingeniero de soporte durante su día de trabajo. Posteriormente se procedió con las verificaciones y pruebas de cada actividad y error.

2.4 Planeación del proyecto

A continuación, se describirá la planeación del proyecto, así como las actividades realizadas en el mismo. Cabe destacar que la manera en la que este proyecto fue concebido no fue enteramente tradicional. Esto es, el proyecto surgió a partir de la idea de automatizar un grupo particular de actividades, y en medida de que el éxito de estas automatizaciones se hacía aparente, se tomó la decisión de incluir actividades adicionales al mismo.

2.4.1 Cronología

La cronología de actividades siguiente ha sido actualizada para que represente de la manera más fidedigna posible la duración de cada actividad, simplificada para su mejor comprensión. Cabe destacar también que, por la misma razón, las actividades no fueron realizadas de manera secuencial en la mayoría de los casos, por lo que la creación de los *scripts* respectivos para cada actividad fue la actividad que más tiempo tomó.

0.1 Cronograma de actividades

Tarea	**Duración**	**Comienzo**	**Fin**
Creación de un script de BASH por actividad	179 días	04/06/2018	30/11/2018
Creación del (o los) stored procedure(s) correspondiente(s)	179 días	04/06/2018	30/11/2018
Creación de script wrapper maestro	119 días	03/07/2018	30/11/2018
Creación de conjunto de CRON jobs para las tareas que lo requieren	83 días	18/07/2018	09/10/2018
Implementación de rastreo de actividad	11 días	06/08/2018	17/08/2018
Implementación de logging en script maestro	11 días	20/08/2018	31/08/2018
Implementación de logging en script de actividad	23 días	03/09/2018	26/09/2018
Creación de un sistema de tablas de respaldo	25 días	06/08/2018	31/08/2018
Inclusión de cuantificaciones diarias en el proyecto	18 días	03/09/2018	21/09/2018
Pruebas preliminares y validación de automatización	18 días	24/09/2018	12/10/2018
Corrección de errores	18 días	24/09/2018	12/10/2018
Validación o pruebas finales	46 días	15/10/2018	30/11/2018
Creación de documentación básica de scripts y tareas	11 días	03/12/2018	14/12/2018

2.5 Descripción de actividades

2.5.1 Creación de un script de bash por actividad

Se desarrolló un *script* de *bash* para cada actividad o error a corregir. Como se verá a continuación, este es el único paso que fue consistente para todas las actividades, ya que el script de *bash* de actividad es el que invoca a los demás archivos necesarios, como los *stored procedures*. Además de esto, el *script* de actividad es el que genera el archivo de *logging* de actividad.

2.5.2 Creación del (o los) stored procedure(s) correspondiente(s)

En lo que respecta a los *stored procedures*, cada actividad requiere de al menos uno o más, ya que estos archivos son los que realizan las modificaciones directamente en la base de datos. Para las actividades más simples, un solo *stored procedure* es suficiente, sin embargo, las actividades más complejas pueden requerir de dos o más.

2.5.3 Creación de script wrapper maestro

Se optó por el desarrollo de un *script* maestro desde el cual se manda llamar a los *scripts* individuales de actividades. El motivo principal de esta decisión fue el de simplificar la ejecución de las tareas al evitarle al usuario el navegar al directorio específico de cada tarea. Además de unificar a todas las tareas en un punto único de ejecución, este *script* genera el *logging* maestro de ejecución y almacena la documentación de las tareas.

Dependiendo de la opción seleccionada el *script* corre la actividad correspondiente. El *output* desplegado por la actividad, así como la base de datos es presentado en pantalla y a su vez guardado en un archivo de texto plano, en el cual se realiza el *logging* de cada actividad, la hora, el resultado y el usuario que ejecutó dicha actividad. El *logging* se realiza en dos niveles: El primero es el *logging* a nivel de script maestro y el segundo es el *logging* de tarea. El *log* de tarea se almacena en el mismo directorio que la tarea, y el *log* del script maestro se almacena en el directorio maestro. El *log* maestro se ejecuta al momento de seleccionar la tarea a ejecutar.

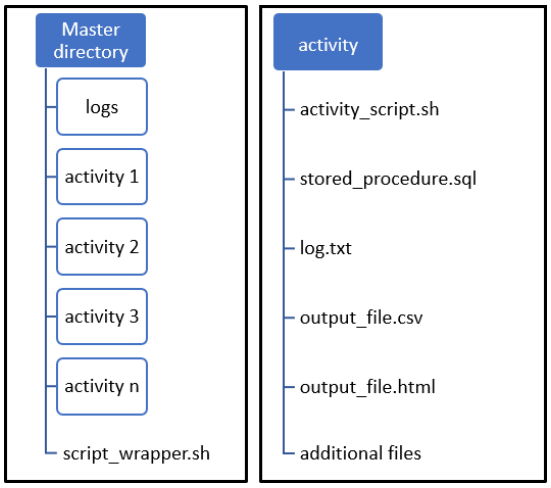


Figura 0.1 Estructura de directorios de directorio maestro y actividad

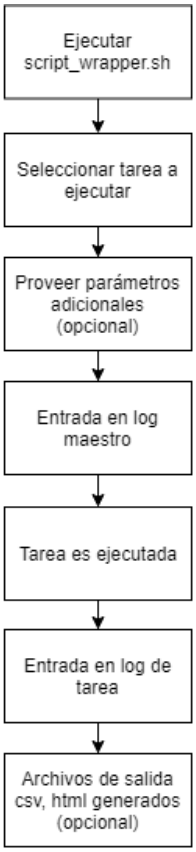


Figura 0.2 Ejemplo de flujo de trabajo de una actividad

2.5.4 Creación de conjunto de cron jobs para las tareas que lo requieren

Una vez que se contó con la fundación del script maestro y los *scripts* y archivos necesarios para cada tarea, se crearon entradas en el *crontab* de los servidores *host* de la base de datos para la ejecución automática de estas tareas. Es importante mencionar que, debido a la naturaleza variada de las tareas, no todas se pueden automatizar de esta manera, así que no todas requirieron la realización de este paso.

2.5.5 Implementación de rastreo de actividad

Cabe mencionar que el sistema de producción no cuenta con una manera robusta para rastrear los cambios realizados en cada registro de la base de datos. Debido a que los cambios en la aplicación y estructura de la base de datos de producción de *B&C Telco* nos son imposibles como equipo de soporte, decidimos utilizar un par de alternativas rudimentarias ya presentes en la base de datos para rastrear las actividades realizadas mediante el *script*.

La base de datos de la aplicación cuenta con dos campos que decidimos utilizar para implementar un sistema de rastreo rudimentario y así poder identificar qué proceso o aplicación en particular realizó el último cambio en cada campo. El primero es el campo *update_date* el cual es un simple campo de fecha o *timestamp* y el segundo es *call_id*. El primer campo es sencillo, un simple *timestamp* que es de evidente utilidad. El segundo campo es el cual nos permitió un poco más de flexibilidad en lo que respecta al rastreo de cambios. Este campo puede contener un valor de hasta 8 caracteres *varchar* (este tipo de valor puede contener letras y números) el cual puede ser definido por el usuario que realiza los cambios, o en este caso, el script ejecutado.

Es por eso que se desarrolló la idea de crear un *call_id* único para cada actividad. En otras palabras, cada que un determinado script se ejecute, además de los registros actualizados por la tarea específica que el *script* realice y el campo de *update_date*, se actualizará el campo *call_id* con el *call_id* asignado al script en particular, de esta manera se cuenta con un registro simple, pero efectivo que nos muestra qué actividad realizó un cambio en algún campo de la base de datos en determinada hora y fecha.

2.5.6 Implementación de logging en script maestro

Con los pasos anteriores realizados, se pudo comenzar la implementación de *logging* en el *script* maestro. Como fue mencionado anteriormente, la actividad a realizar es seleccionada desde el *script* maestro. Una vez que la actividad es seleccionada se inicia el *logging*. Este es guardado en un archivo de texto plano en el directorio maestro e incluye la actividad, hora, resultado y usuario que ejecutó la actividad.

2.5.7 Implementación de logging en script de actividad.

El *logging* de actividad varía demasiado dependiendo de la misma, y se guarda en un archivo dentro del mismo directorio de la actividad. Este archivo puede contener desde la salida de la terminal al momento de ejecutarse la actividad, hasta información específica de la misma, con la cual se evalúa si la ejecución fue exitosa.

2.5.8 Creación de un sistema de tablas de respaldo.

Se determinó desde una etapa temprana del proyecto que era necesario contar con un sistema estandarizado de tablas de respaldo para poder rastrear de una manera confiable los cambios realizados, así como recuperar la información modificada por las tareas y reestablecerla en las tablas de producción en caso de una falla en las mismas.

El sistema de respaldo elegido fue el de una inserción en la tabla de respaldo cada que la actividad es ejecutada. Cabe destacar que la información de respaldo de ejecuciones pasadas de la tarea no es eliminada cuando la tarea se ejecuta de nuevo. Esto es debido a la necesidad potencial de una reconstrucción de datos con base en ejecuciones anteriores en caso de una falla mayor.

Se crearon un grupo de tablas cuya estructura fue copiada directamente de las tablas de producción, pero sin la información contenida en las mismas. Mediante un análisis de cada *script* de automatización y cada tarea a realizar, se creó una hoja maestra en donde rastreamos cada una de las tablas de producción cuya información es modificada por las tareas y se creó una contraparte con la convención de nombre siguiente: el prefijo BKUP seguido por el nombre de la

tabla de producción separada por guiones bajos; esto para identificar fácilmente qué tabla de respaldo corresponde a cada tabla en la base de datos de producción.

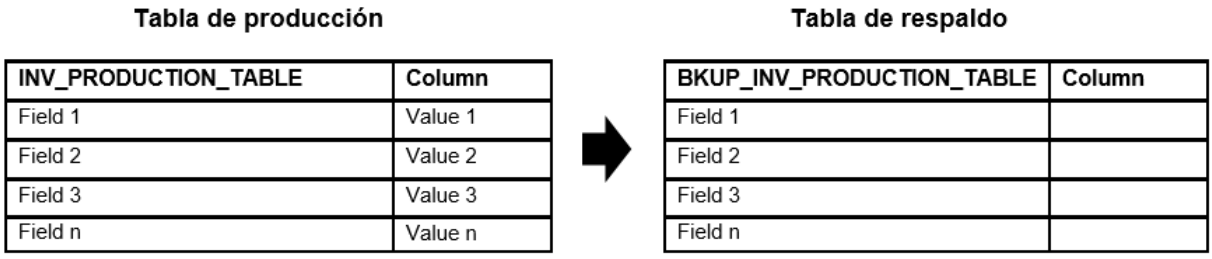


Figura 0.3 Visualización de tablas de producción y respaldo

2.5.9 Inclusión de cuantificaciones diarias en el proyecto

Al percatarnos del éxito que habíamos tenido con las actividades anteriores hasta el momento se decidió empezar a trabajar en la inclusión de las cuantificaciones diarias en el proyecto. La inclusión de las mismas significó por supuesto la creación de su propio *script* de actividad, *stored procedure*(s) y demás archivos necesarios para su ejecución.

Las actividades de cuantificación son más sencillas, pero igual de importantes para la operación diaria de la aplicación y los flujos de negocio. Las cuantificaciones necesitan ejecutarse de manera diaria para cada problema conocido de la aplicación. Estas cuantificaciones son después desplegadas a manera de reporte y presentadas a la unidad de negocio.

Es importante destacar que la automatización de esta actividad es la que más tiempo ahorra al equipo, dado que la corrección de errores es algo que se realiza conforme se necesite, y las cuantificaciones se realizan diariamente.

Dichas cuantificaciones deben ejecutarse todos los días para cada ciclo de facturación. El flujo básico para una cuantificación incluye la creación una serie de tablas, cada una con condiciones diferentes para así recolectar los clientes que están siendo afectados por los errores que están siendo cuantificados.

Una vez realizada la cuantificación, se sube el reporte al repositorio y la actividad termina. Las tablas creadas para la cuantificación del ciclo anterior son eliminadas al momento de ejecutar la cuantificación del ciclo siguiente.

2.5.10 Pruebas preliminares y validación de automatización

Una vez preparado el *script* de *bash*, así como el *stored procedure* de cada determinada actividad, se procedió a realizar la prueba preliminar de los mismos. Las pruebas consistieron en la ejecución de la tarea mediante el script preparado, así como la ejecución manual de la tarea de manera concurrente. En otras palabras, de la misma manera en la que se venía ejecutando la tarea antes del inicio de este proyecto de automatización.

Se estableció un período de prueba de cinco ejecuciones concurrentes por actividad automatizada, de tal manera que se pudo documentar y comparar los resultados de la ejecución de la actividad automatizada con los resultados de la tarea manual. Naturalmente, los resultados debían de ser idénticos en ambas actividades durante cinco ejecuciones para considerarse automatizada exitosamente.

Salvo algunos errores pequeños en algunas de las actividades, la mayoría de las mismas solo requirieron de un par de ejecuciones para poder identificar el problema e iniciar la corrección. Dado que las tareas a ejecutar difieren de una manera considerable en cuanto a la frecuencia de su ejecución, así como las circunstancias en las que deben ser ejecutadas, el tiempo de validación de cada tarea fue variado. Algunas tareas, por ejemplo, requieren ser ejecutadas diariamente, mientras que algunas duraron alrededor de 3 semanas para cumplir las cinco ejecuciones establecidas.

2.5.11 Corrección de errores

En la mayoría de las ocasiones, los procesos fallaban en la primera ejecución y la mayor corrección de errores sucedía después de la misma. Para la tercera y cuarta ejecución la corrección era mínima.

2.5.12 Validación o pruebas finales

Una vez que las actividades se ejecutaban consistentemente sin errores, se estableció un periodo de pruebas de cinco ejecuciones simultáneas. Si la información y los resultados arrojados coincidían en ambas ejecuciones en cinco ocasiones seguidas, la prueba se consideraba exitosa. Dada la diferente frecuencia en la que cada tarea se necesita ejecutar, este periodo de validación varió considerablemente, sin embargo, fue completado para todas las actividades.

2.5.13 Creación de documentación básica de scripts y tareas

Como paso final y debido a algunas aclaraciones y sugerencias del equipo, se procedió a crear documentación básica para cada tarea, esto dentro del *script* maestro. Esta incluye una descripción básica de la tarea, así como qué parámetros y orden de los mismos se necesitan para su ejecución.

2.7 Resultados obtenidos del proyecto

Dados los resultados del proyecto, este se considera un éxito. Lo que empezó como una iniciativa individual de la automatización de un grupo pequeño de actividades evolucionó a la inclusión de tareas y actividades, errores y cuantificaciones diarias. La gran mayoría de actividades elegidas para el proyecto pudieron ser exitosamente automatizadas, y su inclusión exitosa en el proyecto influenció de manera positiva las labores diarias de un equipo de tres personas trabajando para *Swift Solutions*, así como el soporte provisto a la empresa cliente, *B&C Telco* como será descrito a continuación.

2.7.1 Resultados en función de la filosofía de Swift Solutions y B&C Telco

La exitosa realización del proyecto mejora de manera considerable nuestra alineación con la filosofía actual de *Swift Solutions* de adoptar un modelo de trabajo más afín a DevOps. El resultado en función de la filosofía de las empresas es exitoso en dos puntos de vista diferentes; el primero sería el punto de vista de la empresa para la cual nuestro equipo labora, dado que la automatización de estas tareas tediosas y repetitivas se alinea mejor con DevOps.

El segundo punto de vista es, por supuesto, el de la empresa cliente, *B&C Telco*. Desde unos meses atrás, *B&C Telco* ha llevado a cabo una iniciativa muy fuerte de DevOps internamente, lo cual se ha permeado a los requerimientos que *B&C Telco* exige a las empresas que le proveen *outsourcing*. *B&C Telco* exige que toda empresa que provea de servicios de *Outsourcing* trabaje bajo un esquema DevOps. La realización exitosa de este proyecto nos lleva un paso más cerca al cumplimiento de este requerimiento por parte de *B&C Telco*.

2.7.2 Resultados en función de automatización de corrección de errores

Si bien la corrección de errores es una actividad meramente reactiva estrictamente hablando, la extensa cantidad de errores diversos durante las transacciones diarias de una base de datos de clientes de telecomunicaciones de esta magnitud hace que esta propuesta de automatización haya sido valiosa en términos de ahorro de tiempo para el ingeniero de soporte de producción corrigiendo estos errores.

Cabe destacar que, dada la naturaleza de los errores, esta es la única actividad que no puede ser automatizada en su totalidad, esto es debido a varios factores, pero el más importante es el hecho de que los suscriptores que presentan errores de transacción en un día determinado nunca son los mismos en ciclos siguientes. En otras palabras, para tomar acción sobre suscriptores con errores, se necesita una lista de suscriptores acompañada del error que los está afectando. No obstante, al proveer como entrada dicha lista, la corrección de estos errores se ha refinado considerablemente.

2.7.3 Resultados en función de automatización de tareas comunes

Las tareas comunes incluyen principalmente tareas que deben de aplicarse a todos los ciclos de facturación independientemente de si estos presentan errores de suscriptores o no. Esto convirtió a este tipo de tareas en candidatos perfectos para su inclusión en el proyecto.

Dada esta información, un gran porcentaje de estas tareas sí pudieron ser automatizadas en su totalidad y exceptuando un grupo pequeño de actividades de mayor cuidado, la ejecución automatizada de estas actividades fue el segundo mayor factor de ahorro de tiempo para el ingeniero de soporte. Es por ello que también se puede decir que, en este aspecto, el proyecto fue exitoso.

2.7.4 Resultados en función de cuantificaciones diarias

Es importante mencionar que las cuantificaciones diarias fueron las últimas actividades que se incluyeron en el proyecto, y por mucho son las que más beneficios han traído a los ingenieros de soporte en términos de ahorro de tiempo. Estas cuantificaciones deben ser ejecutadas en todos los ciclos de facturación sin excepción. Éstas cuantifican el número de suscriptores que están siendo afectaos en el punto del tiempo en el que la cuantificación es ejecutada por los errores conocidos en ese momento.

Hay una combinación de factores que hacen que la inclusión de las cuantificaciones en el proyecto haya provisto un beneficio tan grande. El primero, es que las bases de datos tienen cientos de millones de suscriptores, y a veces la ejecución de una sentencia de SQL sencilla puede

tomar varios minutos. El segundo es que las sentencias de SQL tienden a ser muy complicadas, y hay cuantificaciones que pueden tomar hasta 45 minutos en ejecutarse. El tercer factor es que los bloques de SQL ejecutados en estas cuantificaciones se deben de ejecutar de una manera serializada. Esto es que un bloque se ejecuta después de otro, y antes de automatizarse esto requería de intervención humana. Ahora, con la simple ejecución de la cuantificación mediante el *script* maestro, la cuantificación se ejecuta de principio a fin sin necesidad de intervención, asumiendo que la ejecución haya sido exitosa.

2.7.5 Resultados generales

Dada la naturaleza del proyecto y el personal reducido del equipo, no se realizaron pruebas formales para determinar la cantidad de tiempo ahorrada por cada ingeniero de soporte en el trabajo de día a día, sin embargo, se estima empíricamente que el tiempo ahorrado puede ir desde 30 minutos a una hora diaria por ingeniero de soporte, dependiendo del número de actividades a realizar y la cantidad de suscriptores que presentan errores en los ciclos de facturación.

Además de esto, la retroalimentación recibida por parte de nuestros supervisores directos ha sido muy positiva, y esperan que otros equipos que proveen soporte a otros módulos de la aplicación y que también realizan actividades diarias similares puedan adoptar un modelo similar y llevar a cabo sus propias iniciativas de automatización.

Capítulo 3. Análisis del Proyecto

El proyecto fue, visto desde diferentes puntos de vista, un éxito. Se logró la semi-automatización de la mayoría de los errores y actividades en la base de datos, así como la completa automatización de algunos de los errores y tareas, así como las cuantificaciones diarias. Esto resulta en menos tiempo invertido por parte de los ingenieros de soporte en tareas repetitivas, lo cual libera tiempo para la investigación de problemas más complejos.

Como será mencionado posteriormente en esta sección, no se realizó una medición de resultados objetiva, aunque esta hubiera sido ideal. Aun así, considerando que el equipo está compuesto por tres ingenieros de soporte y que cada uno ahorra en promedio de media hora a una hora diaria gracias a la automatización de estas tareas, se puede apreciar también el éxito del proyecto en cuestión de tiempo ahorrado.

El *management* de nuestro equipo de soporte, además, expresó su satisfacción con nuestro proyecto y la idea detrás del mismo. Realizar labores de automatización no solo libera tiempo mejor empleado en otras actividades, sino también ayuda a *Swift Solutions* a estar más alineada con su objetivo de convertirse en una empresa DevOps, dado que esta fue una de las peticiones de *B&C Telco* para continuar siendo aliados estratégicos.

Si bien el proyecto fue exitoso, trajo consigo una variedad de problemáticas que se fueron revelando conforme el proyecto avanzaba. La primera de ellas fue la absoluta falta de documentación por parte de *Swift Solutions* con respecto a la solución de software y sus personalizaciones por parte de *B&C Telco*.

Las siguientes problemáticas encontradas fueron la falta de un sistema adecuado y estandarizado de respaldo de información, así como la falta de un sistema de rastreo de actividad, encargado de llevar un registro de qué actividad realiza modificaciones en la base de datos. Cuando se suman estas problemáticas a la falta de privilegios para la modificación de la aplicación, es cuando se empieza a apreciar la dificultad por la que el proyecto atravesó desde su creación.

3.1 Metodología

La metodología empleada durante el proyecto pudo haber sido mucho mejor. Debido a que este proyecto nació como una idea individual con muy poco alcance, el proyecto no fue bien planeado desde un inicio. En lugar de definir desde un inicio de una manera clara qué actividades iban a ser incluidas en el proyecto, estas se fueron incluyendo conforme otras actividades iban terminando. Además de esto, el orden de inclusión de las mismas no siguió ningún patrón lógico, lo cual generó algo de desorden en la realización de estas actividades.

Con respecto a la metodología no todo fue negativo. Aún con el elemento de desorden muy presente en el proyecto, cada actividad se trató como una burbuja; esto es, cada actividad se trabajó de manera individual de inicio a fin, lo cual resultó en algo positivo, ya que el desarrollo de cada actividad consistió de un tiempo individual para cada una, lo cual produjo resultados de calidad para cada una. Las pruebas y validaciones también fueron exitosas en la gran mayoría de las actividades, siendo un porcentaje menor de actividades las que requirieron más de 5 iteraciones para ser corregidas.

3.2 Medición de resultados

Haciendo eco a lo escrito en la sección anterior, la falta de una planeación desde un inicio del proyecto resultó en una medición de resultados no muy precisa. Es decir, es muy claro que la automatización de estos errores y actividades reducen el tiempo invertido por parte de los ingenieros de soporte en actividades repetitivas y tediosas, lo cual libera tiempo para la investigación de errores más complejos.

Lo que no se hizo, sin embargo, es tomar mediciones de tiempo previas y posteriores a la automatización de estas tareas. Si bien, empíricamente es claro que el proyecto tuvo beneficios tangibles, es difícil cuantificar de una manera más determinante qué tanto fue el beneficio del mismo. Esta es, claramente, una de las áreas de mejora más grandes para este proyecto, ya que, de haber tenido información precisa del ahorro de tiempo, este hubiera sido de mucho más interés para la administración.

3.3 Falta de documentación

Desde el inicio del proyecto fue claro que no existe una documentación adecuada para la solución de software de *Swift Solutions*. La solución es enorme; un solo módulo podría tomar un libro completo de documentación, y a pesar de esto no existe un repositorio centralizado de documentación.

La poca documentación disponible se encuentra en documentos que son distribuidos mediante correo electrónico cada que los mismos son requeridos, sin sistema de versionamiento o manera de corroborar la veracidad de la información, considerando que la aplicación cambia después de cada nueva versión de la aplicación.

Gran parte del conocimiento colectivo de cada módulo de la aplicación viene del conocimiento individual de los integrantes de cada equipo de soporte. Este conocimiento no es plasmado en ningún tipo de repositorio. Existe un grupo pequeño de arquitectos de solución que poseen un gran conocimiento de la aplicación, sin embargo, este tampoco es documentado.

3.4 Falta de un sistema de respaldo de información estandarizado

Dada la falta de documentación, la investigación realizada por el equipo arrojó que no existía un sistema de respaldo de información estandarizado. Diferentes equipos de soporte utilizan diferentes métodos y tablas de respaldo, pero no de manera estandarizada. Es por ello que se optó por crear nuestro propio sistema de respaldo.

La creación de este sistema obtuvo especial atención por parte de nuestra administración, y se mostró un interés por parte de la misma para tratar de estandarizar este sistema y propagarlo a otros equipos de soporte. Esta iniciativa, sin embargo, se sale de nuestras manos, y el éxito o fracaso de la misma es ajena a este proyecto. No está fuera de lugar asumir que esta iniciativa no será exitosa, dada la resistencia al cambio general por parte de la mayoría de los empleados de la empresa.

3.5 Falta de un sistema de rastreo de actividad

Fue durante la lluvia de ideas de la recuperación de datos en caso de algún desastre que salió a la luz la falta de un sistema de rastreo de actividad. Yo tenía conocimiento de la falta del mismo, sin embargo, también tenía conocimiento de la existencia de un campo en las tablas de la aplicación llamado *call_id*. Este campo ofrece una manera muy rudimentaria de realizar un rastreo de actividad, ya que se le puede insertar un valor de hasta 8 caracteres alfanuméricos, el cual decidimos utilizar para nuestra implementación de rastreo de actividad.

3.6 Falta de privilegios para modificar la aplicación

Como era de esperarse, la modificación de la aplicación provista a *B&C Telco* estaba totalmente fuera de nuestro alcance. Es por ello que problemáticas como la anterior debieron ser solucionadas de una manera creativa utilizando únicamente los recursos preexistentes en la base de datos de la aplicación.

Cabe destacar que se exploró la posibilidad de realizar cambios en la aplicación siguiendo los canales establecidos, sin embargo, la alta burocracia y la poca disponibilidad de recursos para la inclusión de nuevas características y cambios en el código hizo que esta alternativa fuera descartada.

Capítulo 4. Conclusiones

4.1 Lecciones aprendidas

4.1.1 Planeación del proyecto

Realizar una correcta planeación de un proyecto desde su concepción es una lección que me llevo para cualquier proyecto que decida emprender en un futuro. Aun si las circunstancias del próximo proyecto son similares en un inicio, esto es, que el proyecto surja a partir de una actividad aislada que yo decida emprender, sabré identificar el momento en el cual deba de tomar una pausa y empezar a elaborar un plan de proyecto.

4.1.2 Documentación de actividades

Después de haber sobrellevado la total falta de documentación por parte de la empresa, es aparente que adoptar el hábito de la documentación, así se trate de la tarea o actividad más sencilla es sumamente importante. Formar este hábito es difícil en un inicio, dado que el beneficio inmediato que se recibe es menor, sin embargo, al momento de revisitar una actividad que no se ha realizado en algo de tiempo, o la ejecución de alguna tarea por parte de un nuevo ingeniero, se aprecia su valor. Es por ello que los últimos días del proyecto fueron destinados a la documentación de todas las actividades y tareas que fueron incluidas en este proyecto.

4.1.3 Mejor identificación de procesos que se pueden automatizar

En un inicio, se consideraban tareas y actividades para el proyecto que no necesariamente eran buenos candidatos para su automatización. Las razones pueden ser varias, desde que el proceso es demasiado complejo hasta que el número de entradas y salidas es muy alto como para automatizar con un alto porcentaje de éxito en todas sus ejecuciones. A lo largo del proyecto, el proceso de elección de actividades se fue refinando debido a la experiencia empírica que se fue adquiriendo conforme las actividades se automatizaban o se intentaban automatizar. Es mejor identificar desde un inicio si una actividad puede ser automatizada o no, nos ahorra tiempo valioso.

4.2 Aspectos de mejora

4.2.1 Medición de tiempos

Considerando que el objetivo principal del proyecto fue el de ahorrar tiempo utilizado en actividades tediosas para así aprovecharlo en actividades de investigación más complejas, es claro que el aspecto de mejora más significativo es el de realizar una medición formal de tiempos por actividad antes y después de haber realizado su automatización.

De haberse realizado esta medición, se hubiera contado con un mejor reporte final, basado en cifras exactos y no las mediciones empíricas que nosotros realizamos después de haber finalizado el proyecto. Al final del proyecto, realizamos un cálculo empírico del tiempo que nos tomaba realizar una actividad antes de su automatización y lo comparamos contra el tiempo que nos toma después de la misma. Estas fueron las cifras que fueron presentadas a nuestra administración como reporte del proyecto.

4.2.2 Selección de actividades

De la misma manera que la falta de planeación del proyecto se hizo notar en este proyecto, sucedió con la metodología de selección de actividades. Si bien la metodología de este proyecto en lo que respecta a la elaboración de cada actividad sí siguió un orden relativamente lógico, fue la selección de actividades a incluir en el proyecto la que siguió un orden relativamente arbitrario. En un inicio, las actividades incluidas en el proyecto fueron siendo añadidas conforme la petición de las mismas llegaban al equipo.

A pesar de que este método funcionó, un orden más lógico hubiera sido ideal. Un claro ejemplo es el de la inclusión de las cuantificaciones en el proyecto, la cual ahorró mucho tiempo a los ingenieros. De haberse establecido un método formal de selección de actividades, estas no se hubieran incluido en el proyecto en una etapa tan tardía y se habría ahorrado ese tiempo unos meses antes.

4.3 Conclusiones

En esta era de tecnología, las compañías deben hacer un uso total de todas las ventajas y funcionalidades que el uso de la tecnología les puede brindar. Este proyecto es un claro ejemplo de esta iniciativa, la cual todos los equipos de trabajo deberían tener. Se hizo uso de características de tecnología ya adquirida por la empresa. En otras palabras, no se realizó ninguna adquisición de software, hardware o servicios externos. El resultado final fue tangible, logrando un ahorro de tiempo por cada ingeniero de soporte, sin un compromiso económico adicional por parte de la empresa.

Algo que frecuentemente es ignorado por parte de los empleados de una empresa es su filosofía, así como hacia dónde se dirige. Es importante tener conocimiento de ambas, para así intentar alinear en medida de lo posible las actividades diarias y los proyectos a realizar por parte de los equipos de trabajo. Dada la reciente exigencia por parte de *B&C Telco* para que *Swift Solutions* adopte DevOps, y el cambio de dirección de *Swift Solutions* debido a esto, la realización de este proyecto contribuyó a una mejor alineación por parte de nuestra empresa con *B&C Telco*, por menor que esta sea. Es por ello que visto desde este punto de vista el proyecto fue un éxito, a pesar de los errores y áreas de mejor previamente mencionados.

La variedad de retos encontrados durante el proyecto nos convirtió a su vez en un equipo de trabajo más fuerte y mejor preparado para abordar proyectos y dificultades futuras. La experiencia personal, profesional y técnica que este proyecto me deja es también importante. La magnitud de este proyecto fue algo que experimenté por primera vez, y el hecho de que una o más de mis aportaciones a la empresa sean aprovechadas por otros miembros del equipo hace que el trabajo y resultado obtenido sean muy satisfactorios.

La conclusión final a la que llego con este proyecto, es la de no subestimar en el futuro ninguna actividad en la que participe, ya sea iniciativa individual o grupal. La posibilidad de que esta actividad evolucione en algo más grande debe ser identificada desde una etapa temprana, para así realizar los ajustes correspondientes y seguir una metodología más propia de un proyecto formal.

Bibliografía

Verona, Joakim (2018). *Practical DevOps*. Packt Publishing.

Rankin, Kyle (2012). *DevOps Trobleshooting: Linux Server Best Practices*. Addison-Wesley Professional.

Rosenblatt, Bill & Newham, Cameron (1998). *Learning the Bash, Second Edition*. O'Reilly Media, Inc.

Welsh, Matt – Kaufman, Lar & Kalle Dalheimer, Matthias (1999). *Running Linux, Third edition*. O'Reilly Media, Inc.

Ashdown, Lance & Kyte, Tom (2015). *Oracle Database Concepts*. Recuperado el 24 de abril de 2019 de https://docs.oracle.com/cd/E11882_01/server.112/e40540/intro.htm

Sybase, Inc. *Relational Database Concepts*. Recuperado el 24 de abril de 2019 de http://www.upi.pr.it/docs/easfg/easvrfgp7.htm

Unix Geeks (1999). *Intro to cron*. Recuperado el 30 de abril de 2019 de http://www.unixgeeks.org/security/newbie/unix/cron-1.html

Geeks for Geeks. *Introduction to Linux Shell and Shell scripting*. Recuperado el 30 de abril de 2019 de https://www.geeksforgeeks.org/introduction-linux-shell-shell-scripting/

Glosario

ANSI. *American National Standards Insititute*. Organización sin fines de lucro que supervisa el desarrollo de estándares voluntarios de servicios y sistemas en los Estados Unidos.

CRM. *Customer Relationship Management* o Administración de Relaciones con Clientes es una tecnología utilizada para administrar todas las relaciones de una compañía con sus clientes. Un sistema de CRM le ayuda a una empresa a seguir conectado con sus clientes, agilizar procesos y maximizar ganancias.

Daemon. Tipo de programa en sistemas operativos UNIX / Linux que corre silenciosamente en el sistema esperando a ser activado mediante una condición o evento específico.

Hardware. Componentes físicos como el cableado, máquinas y tarjetas de una computadora o sistema electrónico.

Input. La entrada de un proceso o comando ejecutado en el servidor. En ocasiones la entrada de un proceso puede ser la salida de otro.

Kernel. Elemento central de los sistemas operativos UNIX / Linux. Se encarga de la interface entre el software y el hardware, y puede ser configurado o reemplazado por el administrador de ser necesario.

Output. La salida o resultado de algún proceso o comando ejecutado en el servidor. En ocasiones la salida de un proceso es utilizada como entrada de otro.

Outsourcing. Obtener bienes o servicios por parte de un tercero, en especial si el tercero reemplaza una fuente interna.

SLA. Un Acuerdo de Nivel de Servicio o *Service Level Agreement* es un contrato de un proveedor de servicios o empresa con sus clientes que documenta qué servicios se van a prestar, los tiempos de entrega y las condiciones generales de los mismos. Son importantes para manejar las expectativas de los clientes, así como para definir en qué situaciones particulares las condiciones del acuerdo no son aplicables.

Software. Colección de datos o instrucciones que le indican a una computadora o servidor como operar.

SQLPLUS. Herramienta interactiva incluida en todas las instalaciones de la base de datos de Oracle. Contiene una línea de comandos que hace interface con la base de datos, permitiendo diversas operaciones aplicadas a la información contenida en la misma.

Tecnologías de información. Se refiere al uso de computadoras, almacenamiento, redes y otros dispositivos y herramientas electrónicas para crear, procesar y almacenar datos de la operación de una empresa.

Timestamp. Secuencia de caracteres que identifican cuándo un evento ocurrió. Usualmente consiste de la fecha y hora del día, y puede tener un nivel de precisión de milisegundos.

Troubleshooting. Una forma de resolución de problemas frecuentemente aplicada a la reparación de productos o procesos en un sistema computacional. Es una búsqueda lógica y sistemática de la fuente de un problema para resolverlo.