UNIVERSIDAD DON BOSCO

DIRECCIÓN DE EDUCACIÓN A DISTANCIA

TRABAJO DE GRADUACIÓN:

PLAN DE DESARROLLO Y PLAN DE IMPLEMENTACIÓN PARA UN SISTEMA DE INFORMACIÓN GERENCIAL DE GEOREFERENCIACION, BASADO EN INTELIGENCIA DE NEGOCIOS, PARA EL PERSONAL EJECUTIVO DE UNA EMPRESA EN EL ÁREA DE CENTRO DE ATENCIÓN TELEFÓNICA EN SAN SALVADOR.

PARA OPTAR AL GRADO DE:

MAESTRO/A EN ARQUITECTURA DE SOFTWARE

AUTORES:

JONATHAN MOISÉS GONZÁLEZ LOVO

INGRID ELIZABETH AGUILAR GONZÁLEZ

MELVIN LEONARDO LANDAVERDE CONTRERAS

ASESOR**:**

Mg. JUAN ANTONIO MIRANDA FIGUEROA

Antiguo Cuscatlán, La Libertad, El Salvador, Centro América

Julio, 2022

Rector Universidad Don Bosco

Dr. Mario Rafael Olmos

Secretaria General

Inga. Yesenia Xiomara Martínez Oviedo

Director de Educación a Distancia

Mg. Eduardo Menjívar Valencia

Coordinador de la Maestría

Mg. Mauricio Orlando Figueroa Chicas

Asesor del proyecto de graduación

Mg. Juan Antonio Miranda Figueroa

Lectora del proyecto de graduación

Mg. Evelyn Lissette Hernández de Jiménez

Agradecimientos

A mi padre celestial, quien me ha dado la vida, además de haberme amado, guiado y cuidado en todo momento, por medio de su hijo Jesucristo y de su Santo Espíritu, permitiendo en su infinita misericordia que pudiera culminar este proceso de maestría y también el trabajo de graduación, para él sea toda la gloria y la honra.

A toda mi familia, en especial a mis padres Eduardo y María Teresa, a mis hermanos Oscar, Douglas, Mauricio, Rubí y Gema, y a mi hija Krissia Fiorella, por la confianza depositada en mi persona, y por haberme apoyado durante todo este desafío, que requiere mucho esfuerzo y dedicación.

A los todos los docentes de la Maestría en Arquitectura de Software de la Universidad Don Bosco, y a su coordinador Mauricio Figueroa, quienes han brindado sus conocimientos y nos han acompañado con sus enseñanzas y concejos, gracias a los cuales tenemos una excelente formación académica y profesional en la rama de las tecnologías de la información.

A mis compañeros de maestría, María Velasco, Xiomara Chicas, Ingrid Aguilar, Moisés Lovo, Cristian Padilla, Jeovanni Hernández y Nahum Portillo, quienes trabajamos juntos y nos esforzamos a lo largo de todo este proceso, desde el primer día de clases, hasta la culminación del trabajo de graduación con Ingrid y Moisés.

Melvin Landaverde.

A Telus International y la Universidad Don Bosco, si no hubiera sido por la oferta ofrecida por medio del convenio, creo que no habría decidido el estudiar esta maestría, ya que la oferta económica fue una gran oportunidad para superarme y descubrir nuevos conocimientos.

A mi familia por siempre brindarme apoyo moral y de salud en todo momento durante estos casi 3 años de proceso, motivándome a no rendirme a medio camino y brindándome el bienestar necesario para poder trabajar lo más tranquilo posible ya que el llevar el trabajo y la universidad al mismo tiempo nunca ha sido algo fácil para nadie y menos a nivel de maestría.

Jonathan González.

A mis padres, Deisy de Aguilar y Carlos Aguilar, por siempre desear y anhelar lo mejor para mi vida, gracias por cada consejo y por cada una de sus palabras. Por el apoyo incondicional en los momentos adecuados, cuando necesitaba un impulso para seguir adelante. Les dedico este triunfo.

A mis compañeros, con los que compartí noche de desvelos, frustraciones, disgustos, tareas y proyectos. Pero también disfruté, reí y aprendí a su lado. Les agradezco a cada uno por todos esos momentos que tienen una parte importante en mí, a los que conocí desde del inicio de esta carrera, como a los que conocí en el trascurso de esta.

A mi estimado Alcides, por ser ese sostén cuando mis ánimos menguan, por confiar en mí y, alentarme para desarrollar mis metas y entre ellas mi culminar la maestría. Gracias por tu ayuda, fue clave para lograr este éxito.

A mis compañeros de trabajo de graduación, por la dedicación, esfuerzo y motivación dado para llegar a este satisfactorio éxito.

Ingrid Aguilar.

Índice

Capitulo 1. Introducción 1
1.1 Antecedentes 2
1.2 Problemática 3
1.2.1 Formulación del problema 3
1.3 Justificación 3
1.4 Supuestos 6
1.4.1 Hipótesis de la investigación 6
1.5 Alcance 6
1.6 Limitaciones 6
1.7 Objetivos 6
1.7.1 General 6
1.7.2 Específicos 7
Capitulo 2. Marco teórico 7
2.1 Sistema de Información Geográfico 7
2.2 Integración de Sistemas 8
2.3 Inteligencia de Negocios 8
2.4 Geolocalización 9
2.5 Centro de Atención Telefónica (Contact Center) 10
2.6 Modelo CMMI 10
Capitulo 3. Metodología de la investigación 11
3.1 Enfoque epistemológico 11
3.2 Tipo y diseño de investigación 12
3.2.1 Tipo de investigación 12
3.2.2 Diseño de la investigación 13
3.3 Técnicas e instrumentos de recolección de datos a utilizar 13
3.3.1 Metodología 14
Capitulo 4. Solución propuesta 15
4.1 Plan de desarrollo 15
4.1.1 Objetivos 15
4.1.2 Alcances 15

4.1.3 Limites 16
4.1.4 Metodología 16
4.1.5 Diseño 24
4.1.6 Construcción 32
4.2 Plan de implementación 33
4.2.1 Objetivo 33
4.2.2 Organización 34
4.2.3 Actividades 34
4.2.4 Aplicación del modelo CMMI 34
4.2.5 Gestión de modificación y errores 42
4.2.6 Aceptación de entrega 43
4.3 Desarrollo de prototipo 45
4.4 Implementación de prototipo 48
Capitulo 5. Presentación de resultados del prototipo 50
Capitulo 6. Discusión 51
Capitulo 7. Conclusiones 56
Capitulo 8. Recomendaciones 57
Capitulo 9. Referencias 58
Capitulo 10. Anexos 59
Anexo 1: Guía técnica de configuración y despliegue del sistema 59
Base de datos 59
Integración de datos 65
Interfaz de usuario 67
Anexo 2: Script de creación de la base de datos 76
Anexo 3: Código fuente de programa de integración de datos 146
Integración.cs 146
MockUpData.cs 149
DBConn.cs 150

Índice de tablas

Tabla 1. Resumen de metodología de investigación 14
Tabla 2. Visión del producto 18
Tabla 3. Declaración visión del producto 19
Tabla 4. Definición de roles 20
Tabla 5. Actividades de desarrollo 20
Tabla 6. Plan de lanzamiento 21
Tabla 7. Historia épica 1 21
Tabla 8. Historia épica 2 22
Tabla 9. Historia épica 3 22
Tabla 10. Historia épica 4 23
Tabla 11. Actividades de implementación. 34
Tabla 12. Ejemplo de evaluación de prácticas específicas “Gestión de la configuración (CM)” 40
Tabla 13. Ejemplo de cuadro resumen de niveles de madurez según área de práctica 41
Tabla 14. Plantilla aceptación de entregables 44
Tabla 15. Casos de pruebas de entregables. 44
Tabla 16. Tipos de licencias. 46
Tabla 17. Características de cada plan de Power BI. 47
Tabla 18. precios de licencias. 47

Índice de figuras

Figura 1. Áreas y procesos de negocio beneficiados con el desarrollo de este proyecto. 4
Figura 2. Beneficios cuantitativos esperados. 5
Figura 3.Beneficios cualitativos esperados 5
Figura 4. pirámide organizativa. 9
Figura 5 Proceso cualitativo. 12
Figura 6 Técnicas e instrumentos de recopilación de datos 13
Figura 7 Descripción de la visión del proyecto. 17
Figura 8 Modelo lógico de la base de datos 24
Figura 9 Modelo físico de la base de datos 25
Figura 10 Diagrama de integración de sistemas 27
Figura 11 Diagrama de componentes. 28
Figura 12 Dashboard general. 29
Figura 13. Mapas 29
Figura 14. Métricas 30
Figura 15. Gráficas 31
Figura 16 Diagrama de flujo de desarrollo 32
Figura 17 Diagrama de flujo de sistema de registro de incidencias. 33
Figura 18. Organización. 34
Figura 19 Bohem Spiral Software Model. 35
Figura 20 Figura 17. Tratamiento de Cambios. 43
Figura 21. Dashboard prototipo. 50
Figura 22. Información por locación. 51

Resumen

La pandemia del covid-19 ha obligado a un inmenso número de organizaciones del mundo, a implementar el esquema de trabajo remoto desde el domicilio de cada empleado, lo que obliga a establecer nuevos mecanismos y controles sobre las condiciones en las que los colaboradores ejercen sus labores cotidianas, de tal manera de asegurar la eficiencia en las operaciones, con la ayuda de las tecnologías de la información y las comunicaciones.

Los Sistemas de Información Gerencial son herramientas clave para los niveles ejecutivos que toman decisiones en las empresas, ya que estos proveen información sobre la base de datos procesada de diferentes fuentes, en los cuales el oportuno tratamiento, exactitud, velocidad, integración y presentación, ofrece una visión estratégica sobre la cadena de valor para cualquier tipo de negocio, centrándose en aquellos elementos que les interesan a los analistas.

Existen conceptos estratégicos que ayudan a abordar de manera holística la solución de esta problemática, por ejemplo, para el caso de estudio, serán la georreferenciación y estrategias de inteligencia de negocios, en donde el enfoque principal se centra en la presentación visual de una manera amigable, de la información de localización de las personas que laboran en una empresa, en donde existen indicadores claves que miden márgenes de eficiencia en los procesos.

Para efectuar el desarrollo de este tipo de sistemas, primero se necesita un proceso eficiente de Ingeniería de Requisitos, el cual ayuda en gran manera a solucionar el problema de asegurar la satisfacción de las necesidades del cliente. Con relación a la implementación, el abordaje de ciertas áreas de prácticas del modelo CMMI, ayuda a establecer una guía sobre la cual una organización debe asegurar el cumplimiento de ciertos elementos para alcanzar el éxito en la operación productiva del sistema.

El abordaje de la construcción y presentación de un prototipo es clave para verificar el funcionamiento base esperado de la solución a la problemática, ya que permite evaluar con antelación, de una manera conveniente, la información que el sistema estará generando posteriormente, cuando se lleve a cabo su desarrollo e implementación. Es fundamental este paso, ya que permitirá evaluar de una manera analítica, los impactos en la organización, permitiendo realizar los ajustes necesarios y asegurando la escalabilidad y estabilidad de la solución hacia muchos niveles, tanto funcional, como de infraestructura y de servicios requeridos para la operación.

El proceso de toma de decisiones de los ejecutivos de la empresa es el que encontrará mayores beneficios al poseer este tipo de soluciones, ya que les permitirá establecer sus estrategias sobre la base de datos procesados, convertidos en información útil a través de diferentes estrategias de inteligencia de negocios, con el manejo de repositorios de datos y presentación de resultados en forma de mapas geográficos, ayudará visualmente en gran manera, examinando diferente tipo de información por zonas; siendo esto muy útil, rápido y fácil de procesar, ya que contribuirá a que el análisis e interpretación, se aborden de manera sistemática e integral.

Lista de abreviaturas y símbolos

ADS: Azure Data Studio (Herramienta de Estudio de Datos Azure).
Apps: Applications.
BI: Business Intelligence.
CAR: Cause Analysis & Resolution (Análisis de la causa y resolución).
CC: Call Center o Contact Center. (Centro de llamadas o centro de contactos)
CLI: Command-line interface (Interface de línea de comandos).
CM: Configuration Management (Gestión de la configuración).
CMMI: Capability Maturity Model Integration (Modelo Integrado de Madurez de Capacidades).
CRM: Customer Relationship Management (Administración de Relación con Clientes)
DAR: Decision Analysis & Resolution (Análisis de decisiones y resolución).
DTU: Database Transaction Unit (Unidades de transacción de base de datos).
EDT: Estructura de desglose de trabajo.
ENQ: Ensuring Quality (Garantizar la calidad).
ERP: Enterprise Resource Planning (Planeador de Recursos Empresarial)
GB: Gigabyte.
GOV: Governance (Gobernanza).
II: Implementation Infraestructure (Infraestructura de implementación).
IP: Internet Protocol (Protocolo de Internet).
ISTQB: International Software Testing Qualifications Board (Comité Internacional de Calificación en Pruebas de Software).
PaaS: Platform as a Service (Plataforma como servicio).
PQA: Process Quality Asurance (Aseguramiento de la calidad del proceso).
PR: Peer Reviews (Revisiones por pares).
REQM: Requirements management (Gestión de requisitos).
RDM: Requeriments Development & Management (Desarrollo y gestión de requisitos).
RRHH: Recursos Humanos
SHP: Sustaining Habit & Persistence (Sostener el hábito y la persistencia).
SI: Suporting Implementation (Apoyar la implementación).
SIG: Sistema de información geográfica.
SLA: Service Level Agreement (Acuerdos de niveles de servicio).
SQL: Structured Query Language (Lenguaje de consultas estructurado).
SSMS: SQL Server Management Studio (Herramienta de Estudio de Administración de SQL Server).
TFVC: Team Foundation Version Control (Control de versiones de equipo de fundación).
T-SQL: Transact SQL (Lenguaje de Transacciones SQL).
VV: Verification & Validation (Verificación y validación).

Capitulo 1. Introducción

Este informe presenta los resultados de una investigación enfocada a un caso de estudio, el cual se centra en el plan de desarrollo e implementación para un sistema de información gerencial de georreferenciación. La solución que se propone en este trabajo de investigación proporciona una respuesta a un problema de la vida real de las empresas del sector de los centros de atención telefónica, dicha propuesta se basa en inteligencia de negocios, y se dirige hacia el personal ejecutivo que toma decisiones hacia los niveles tácticos, estratégicos y operativos, con el fin de ayudar a mejorar la eficiencia de las operaciones y asegurar de manera óptima la facturación de las cuentas de sus empresas clientes.

En primera instancia, se presentan los antecedentes sobre los cuales oscila esta temática, en los que el concepto principal gira en base a los sistemas de información geográficos, posteriormente se define la problemática a solucionar, formulando la pregunta de la investigación, además se desarrolla la justificación, hipótesis, alcances y limitaciones, así como los objetivos, tanto general como específicos. El marco teórico provee una visión de los conceptos claves para proponer esta clase de soluciones, así mismo, la metodología de la investigación, la cual se define de tipo descriptiva y cualitativa, muestra el diseño de esta, así como las técnicas e instrumentos de recopilación de información.

En la solución propuesta, se cumple con la razón principal de esta investigación, la cual es definir un plan de desarrollo e implementación para un sistema de información gerencial, abordando la temática anteriormente expuesta. En adición, se incluye el tema de la elaboración de un prototipo, el cual es una demostración práctica de un software funcional, el cual posee las características fundamentales para manejar la información de georreferenciación del personal que labora en este tipo de empresas. Se presentan los resultados de la investigación a través de un enfoque totalmente práctico, los cuales se centran en el plan de desarrollo e implementación de un Sistema de Información Gerencial para la solución del caso de estudio.

En la discusión de la investigación, se abordan puntos claves, como por ejemplo la evaluación del cumplimiento de la hipótesis, igualmente del prototipo, así como aspectos relacionados con el caso de estudio, es decir el impacto en las operaciones de la empresa, además de la evaluación del fortalecimiento del proceso de toma de decisiones a nivel ejecutivo.

Al final del documento se establecen las conclusiones y recomendaciones de este estudio, el cual busca ayudar el fortalecimiento del negocio, con el objeto de mejorar la eficiencia operativa y el posicionamiento en el mercado de este tipo de empresas, asegurando para sus accionistas, que sean rentables y sostenibles a lo largo del tiempo.

1.1 Antecedentes

Mejía (2019) realizo un artículo sobre los Sistemas de información geográfica y su aporte a la salud pública en El Salvador. Centrándose en su beneficio para el fortalecimiento en la toma de decisiones. La geolocalización de los casos o registros de enfermedades trae muchas ventajas para describir los problemas de salud y tomar cartas en el asunto.

En El Salvador, el uso de SIG (Sistemas de información geográfica) han tenido un avance importante en el sistema de salud pública desde la reforma de salud en 2009, que promovió la integración de un sistema único de salud. Mejía menciona que en el año 2013 se lanzó por primera vez la aplicación de los SIG con la plataforma Geominsal, un visor de mapas que brinda información para mejorar la planificación de los servicios y tomar decisiones respecto a los territorios vulnerables a partir de riesgos y determinantes sociales.

Villalta (2011), realizo una investigación sobre Sistemas de Información Geográfica (SIG) para manejo de información espacial y georreferencial aplicado a los estudiantes de la Universidad Gerardo Barrios (UGB) con el objetivo de determinar las posibles aplicaciones que pueden desarrollarse utilizando la información generada por la comunidad universitaria relacionada a proyectos de investigación, proyectos sociales y domicilios de estudiantes.

El sistema que se desarrolló a partir de este proyecto se orientó a convertirse en una fuente de información para apoyar la toma de decisiones, gracias a facilitar el análisis de múltiples variables de entorno, lo cual puede ahorrar recursos y tiempo que se utilizaría en desplazarse a los lugares para obtener la información.

1.2 Problemática

1.2.1 Formulación del problema

A continuación, se procede a plantear de manera formal el problema que se desea resolver con el desarrollo de este trabajo, es decir, ubicar donde se centra la idea principal, la pregunta de investigación se formula de la siguiente manera:

¿Cómo debe estructurarse un plan de desarrollo y un plan de implementación de un sistema de información gerencial de georreferenciación, basado en inteligencia de negocios, orientado al fortalecimiento del proceso de toma de decisiones para el personal ejecutivo, de una empresa en el área de centro de atención telefónica en San Salvador, para el año 2022?

1.3 Justificación

Se requiere contar con un plan de desarrollo e implementación para un sistema que presente información gerencial de georreferenciación, en forma de mapas de Google, con información contextualizada en tiempo real, así como también reportes y otros tipos de instrumentos de inteligencia de negocios, con el fin de ayudar en la gestión de empleados que atienden problemas de clientes de un centro de atención telefónica.

Este sistema ayudará en el proceso de toma de decisiones para el área ejecutiva, para una mejor gestión de contingencias, por ejemplo, inundaciones, terremotos, falta de energía eléctrica, internet, entre muchos otros eventos.

Este proyecto incidirá en la mejora de procesos de atención al cliente para diferentes tipos de regiones, con posibilidades a futuro de ser aplicado en otros países donde la organización pueda operar, además debe contar con capacidades multilenguaje.

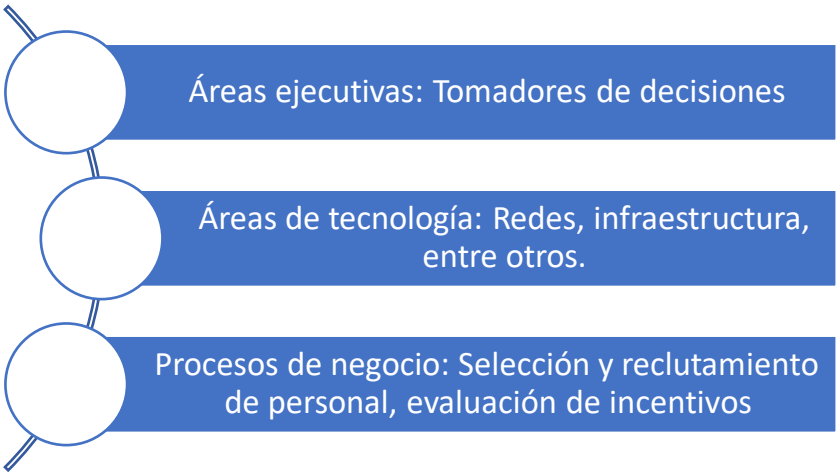


Figura 1. Áreas y procesos de negocio beneficiados con el desarrollo de este proyecto.

Fuente: Elaboración propia

En la figura 1 se muestran aquellas áreas y procesos de negocio que se verán afectadas de manera positiva con el desarrollo de este proyecto, en donde las áreas ejecutivas son los encargados de tomar decisiones estratégicas, las áreas de tecnología son los equipos que trabajan con redes, servidores, equipos de desarrollo de software, entre otros. y los procesos de negocio que se tienen, la selección y reclutamiento de personal, y la evaluación de los incentivos a los empleados por sector geográfico. También se espera generar crecimiento y beneficios tanto cuantitativos como cualitativos, con impactos positivos en las operaciones del negocio.

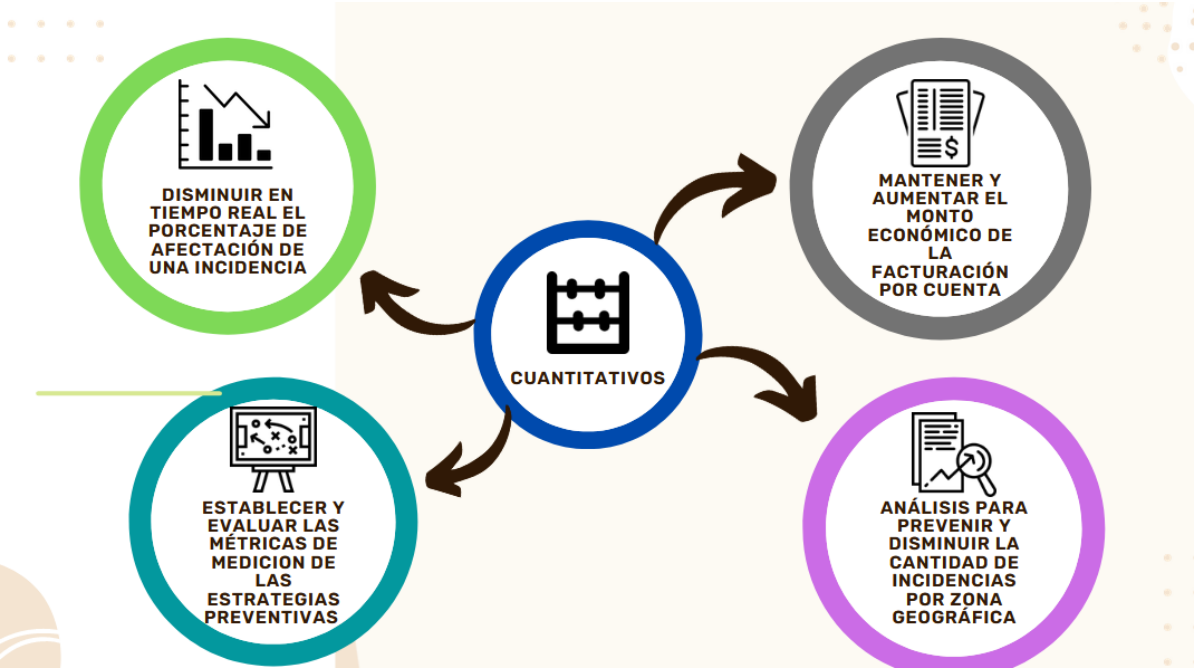


Figura 2. Beneficios cuantitativos esperados.

Fuente: Elaboración propia

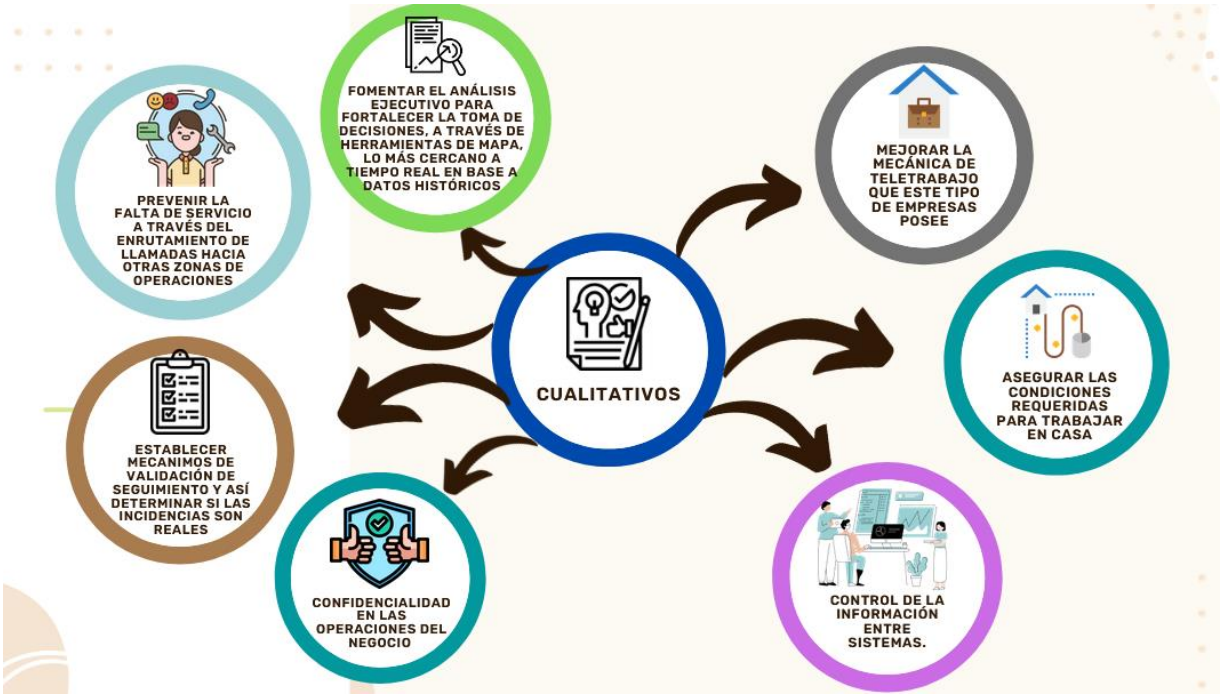


Figura 3.Beneficios cualitativos esperados

Fuente: Elaboración propia

El propósito central de contar con este tipo de herramientas, es focalizar los servicios afectados debido a una contingencia, verificar prevenciones, por ejemplo, zonas con alta incidencia de caídas de internet, falta de energía eléctrica, entre otros, incidiendo de esta forma en el proceso de toma de decisiones en la gestión de personal por zona geográfica, además de fomentar el manejo de otro tipo de estrategias, por ejemplo, la generación de lealtad entre los empleados, la creación de planes efectivos de acción ante contingencias, y mucho más.

1.4 Supuestos

1.4.1 Hipótesis de la investigación

La implementación de un prototipo de un módulo del sistema de información gerencial de localización brindará a la empresa la experiencia sobre el uso de la información de georreferenciación obtenida mediante inteligencia de negocios e integración de sistemas, con el fin de tomar decisiones relacionadas a los agentes y los eventos relacionados al teletrabajo.

1.5 Alcance

El presente trabajo de investigación tiene como alcance definir un plan de desarrollo, y un plan de implementación de un sistema de información gerencial de georreferenciación basado en inteligencia de negocios. Asimismo, se mostrará el desarrollo de cada paso y proceso en la construcción de cada plan. También, abarca la presentación de un prototipo, que se desarrollará como un mínimo producto viable de un módulo del plan que se presentará.

1.6 Limitaciones

Este proyecto incluye el construir una propuesta de plan de desarrollo y un plan de implementación, más no la ejecución de estos. Los planes serán entregados a la empresa donde se está desarrollando la investigación.

1.7 Objetivos

1.7.1 General

Proponer un plan de desarrollo y un plan de implementación de un sistema de información gerencial de georreferenciación, basado en inteligencia de negocios y datos de geolocalización

(inteligencia de ubicación), para el personal ejecutivo de una empresa en el área de centro de atención telefónica en San Salvador.

1.7.2 Específicos

1. Describir componentes y características en la elaboración de sistemas de georreferenciación, integrando tecnologías de geolocalización e inteligencia de negocios.
2. Optimizar el proceso actual de la toma de decisiones a través de un sistema de información gerencial de georreferenciación para los ejecutivos de una empresa en el área de centro de atención telefónica.
3. Construir un prototipo funcional de un módulo del sistema de información gerencial de georreferenciación, a partir del plan de desarrollo e implementación propuesto.

Capitulo 2. Marco teórico

2.1 Sistema de Información Geográfico

Los Sistemas de Información Geográfica (SIG) son aplicaciones que permiten el uso de mapas y bases de datos en conjunto para realizar análisis de información georreferenciada que normalmente se realiza manualmente sobre mapas físicos o digitales alterando su contenido para tener visibilidad de los eventos o datos que se desean estudiar.

Según Olaya (2014), las tres funciones principales de un SIG son las siguientes:

- Lectura, edición, almacenamiento y gestión de datos espaciales.
- Análisis de dichos datos: Puede realizarse desde consultas sencillas a elaboración de complejos modelos, y puede llevarse a cabo sobre la componente espacial de los datos sobre como la componente temática.
- Generación de resultados tales como mapas, informes, gráficos, entre otros.

Otras definiciones de SIG dependen de cuales aspectos se valoren como más importantes, un SIG puede ser un elemento que permite analizar, presentar e interpretar hechos relativos a la superficie terrestre, pero también se puede definir un SIG como software y hardware dedicado a adquisición, mantenimiento y uso de datos cartográficos.

2.2 Integración de Sistemas

La integración de sistemas consiste en la conjunción de la operación de distintos sistemas informáticos que no se intercomunican entre sí, pero que tienen relación a nivel de negocio por medio de un software adicional, ya sea creado o a la medida, lo cual se conoce como middleware. Un middleware es un software que se utiliza como un puente entre distintos sistemas para comunicar la información de un sistema con otro, tomando control de la carga de recursos que esto conlleva por separado.

Normalmente se utilizan Enterprise Resource Planning (ERP), Customer Relationship Management (CRM) u otros programas o técnicas de integración, debido a su potencial de procesamiento de información contenida dentro de su base de datos, a modo de buscar las soluciones más optimas y brindar detalles referentes al negocio o los clientes, que normalmente llevarían mucho esfuerzo en ser calculados de forma manual. Estas bases de datos son pobladas por la información proveniente de otros sistemas.

La integración de datos, por otro lado, es un proceso de combinación de información de distintas fuentes por medio del uso de una aplicación. Esto brinda a los usuarios la posibilidad de fusionar información de documentos, bases de datos, matrices u otros elementos para su posterior análisis y transformación.

2.3 Inteligencia de Negocios

Inteligencia de negocios es el conjunto de estrategias, acciones y herramientas enfocadas a la administración y creación de conocimiento mediante el análisis de datos. Dicho tratamiento de la información es de gran utilidad para las empresas debido a su valor como retroalimentación para tomar decisiones sobre ellas y así tener una idea más clara sobre cómo proceder ante esas eventualidades.

La inteligencia de negocios también es considerada la siguiente etapa en el crecimiento de una empresa, por medio del software, es decir, es el siguiente eslabón en la pirámide luego de haber implementado los sistemas transaccionales. Como se mencionó anteriormente, la inteligencia de negocios busca ordenar la información existente y tratarla de manera que sea capaz de mostrar información que pueda ser analizada, ya sea manualmente, en un sistema para procesar información o en un sistema externo que permita visualizarla de mejor manera para tomar decisiones. En la figura 4 se presenta la pirámide organizativa, la cual muestra los niveles

en los que encaja cada aplicación que se puede utilizar en una empresa para su desarrollo y crecimiento.

Oracle, en una definición para la inteligencia de negocios, según el Data Warehouse Institute, se define como la combinación de tecnología, herramientas y procesos que se utilizan para transformar los datos almacenados en información, dicha información en conocimiento y dirigir este conocimiento a un plan o estrategia. La inteligencia de negocios debe ser parte de la estrategia empresarial, ya que permite la optimización de recursos, monitorear cumplimiento de métricas y apoya a la toma de decisiones para mejorar resultados.

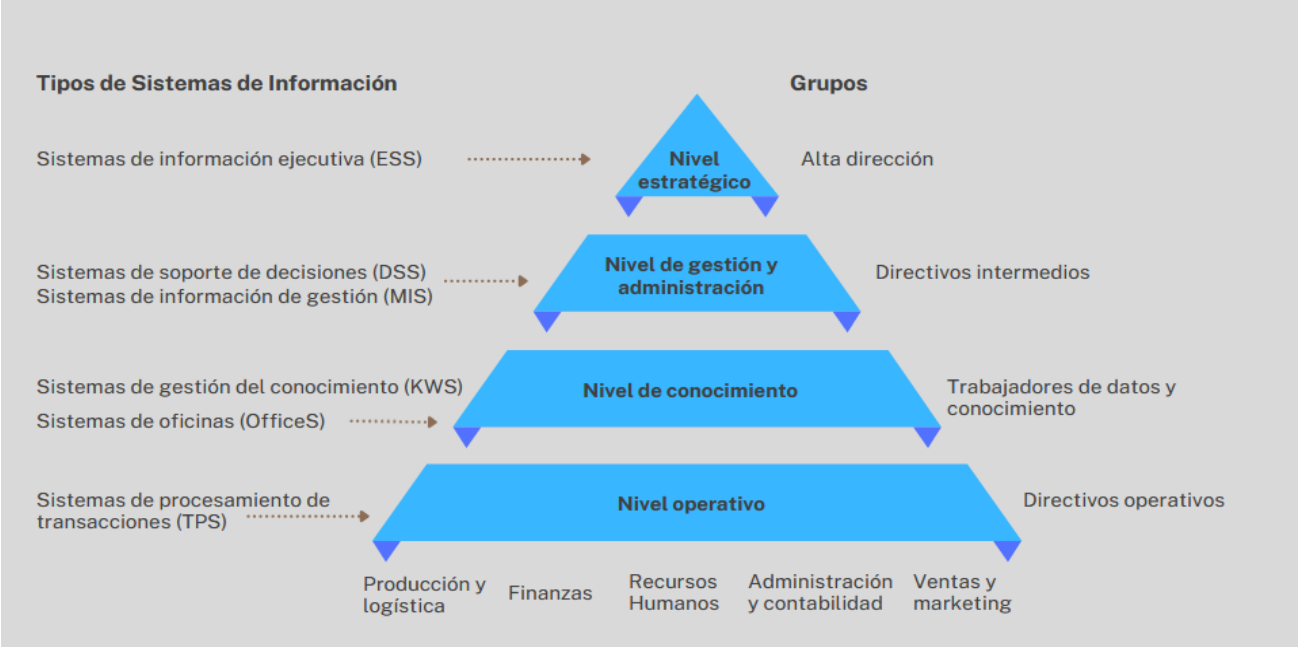


Figura 4. pirámide organizativa.
Fuente: Elaboración propia

2.4 Geolocalización

El termino geolocalización hace referencia a las técnicas o herramientas que se utilizan para determinar la posición de una entidad en un territorio o un mapa mediante el uso de coordenadas u otras unidades que determinan la posición de un objeto.

Inicialmente la geolocalización moderna se utilizaba con fines militares para determinar tanto la posición de las tropas aliadas como la ubicación de bases enemigas por medio de GPS. Luego estos dispositivos fueron siendo liberados al público para ayudar a resolver problemas relacionados a la ubicación, como ciertos GPS usados en vehículos, o también el poder ubicar un lugar por medio de aplicaciones como Google maps.

La evolución de la geolocalización ha ayudado mucho a las personas, tanto en términos de ubicar lugares, como al estudio del comportamiento de una población respecto a un determinado factor de estudio, en este punto es donde se combinan las técnicas de geolocalización con los Sistemas de Información Geográfica o SIG.

Según Cascón, Ruiz y Alberich (2017), las tecnologías de geolocalización y georreferenciación y los términos que las designan tienen su origen previo en la geografía, su popularización está ligada con la aparición y uso a gran escala a lo largo de la última década en aplicaciones web y redes sociales que brindan servicios dependiendo de la necesidad del usuario y lo que lo rodea. Algunos ejemplos son Foursquare, TripAdvisor, Google Maps, Wallapop y Pokemon GO. Mas allá de sus usos comerciales, también se utiliza la georreferenciación dentro del campo de la cartografía debido a la importancia de su área de estudio.

2.5 Centro de Atención Telefónica (Contact Center)

Los Centros de Atención telefónica, también conocidos como Call Center o Contact Center son empresas o áreas de las empresas destinadas a la centralización de las llamadas recibidas por una empresa para optimizar la atención telefónica bajo un mismo número telefónico.

Inicialmente los call centers eran manejados por operadores llamados conmutadores, los cuales redirigían manualmente las llamadas a las áreas correspondientes. Luego esto fue evolucionando al uso de extensiones para llamar directamente a áreas determinadas de una empresa, hasta llegar al punto actual en el cual, por medio de una planta telefónica privada, se centralizan las llamadas en un solo número y son atendidas por múltiples agentes dentro de la empresa. Estas tecnologías van desde el uso de telefonía normal hasta el Voice over IP (VoIP).

Según Micheli (2012), los servicios producidos por los call centers (CC) han sido emblemáticos en la interacción entre la empresa y el consumidor, la cual, es un factor intangible de competitividad para las primeras organizaciones.

2.6 Modelo CMMI

CMMI es un modelo de mejora de rendimiento para organizaciones y proyectos que desean lograr un rendimiento cada vez mayor, así como abordar y resolver los desafíos de negocio. Con más de 25 años de demostrar ser globalmente efectivo en negocios y gobierno, el

CMMI es un marco de trabajo integrado de mejores prácticas que pueden rápidamente mejorar y mantener el desempeño de cualquier organización para elevar su calidad, rentabilidad y competitividad.

El modelo de CMMI no es prescriptivo; describe qué se debe hacer para mejorar las capacidades de una organización, no cómo hacerlo. Esto hace que el modelo sea muy flexible para cumplir las necesidades únicas de cualquier empresa. Además, el CMMI complementa y aumenta la mejora de rendimiento en conjunción con otros modelos y estándares de la industria. CMMI se utiliza para establecer procesos que le ayudarán a la organización o al proyecto a cumplir con sus objetivos de negocio y mejorar el rendimiento de las formas que más importan. (CMMI Institute, 2018).

Capitulo 3. Metodología de la investigación

3.1 Enfoque epistemológico

Enfoque seleccionado: Enfoque de investigación cualitativo.

Este enfoque tiene como objeto comprender los fenómenos y explicar la situación estudiada, mediante el uso de recolección y análisis de datos para afinar la pregunta de investigación o generar una nueva. El proceso se mueve de forma dinámica entre lo que se obtiene y su interpretación. (Hernández Sampieri et al., 2014).

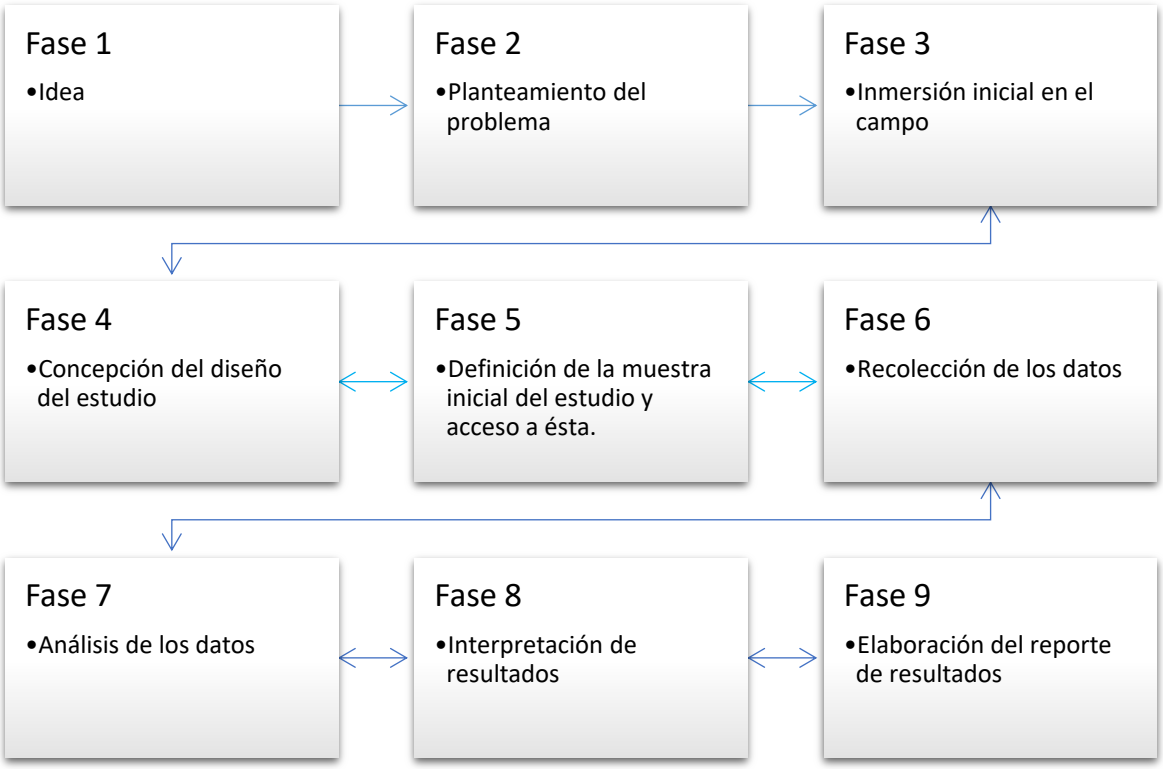


Figura 5 Proceso cualitativo.

Fuente: Hernández Sampieri et al (2014)

En la Figura 5, se puede observar los pasos que describe Sampieri (2014) para realizar una investigación cualitativa, este tipo de enfoque parte de una idea, en la cual se busca resolver un problema, se utiliza una fuerte base de información para el marco teórico para la inmersión inicial, con esto diseñar el tipo de investigación, la muestra, la recolección y el análisis son fases que se realizan prácticamente de manera simultánea; para este tipo de enfoque no se prueban hipótesis, en cambio se generan durante, o al final de la investigación.

3.2 Tipo y diseño de investigación

3.2.1 Tipo de investigación

Descriptiva, de acuerdo con Hurtado de Barrera (2015, citado por Atencio, 2020) este tipo de investigación enfatiza el reporte de comportamientos o características precisas del evento estudiado.

En vista de lo anterior, la investigación presente pretende, identificar y describir los componentes para establecer un plan de desarrollo y un plan de implementación orientado a sistemas gerenciales de localización de personal e inteligencia de negocio.

3.2.2 Diseño de la investigación

El diseño de la investigación proporciona los criterios necesarios que marcan la ruta para la extracción de la información que se requiere. Para la presente investigación se aplicarán los siguientes criterios.

Documental, este tipo de diseño conduce a la recolección de datos desde diversas fuentes bibliográficas y no bibliográficas. (Atencio, 2020, p.14)

Un solo momento, Se enfoca en el proceso de recolección de datos en un único periodo. (Atencio, 2020, p.14)

Un solo evento, Se centra en el abordaje de un único evento o fenómeno de estudio. (Atencio, 2020, p.14)

3.3 Técnicas e instrumentos de recolección de datos a utilizar

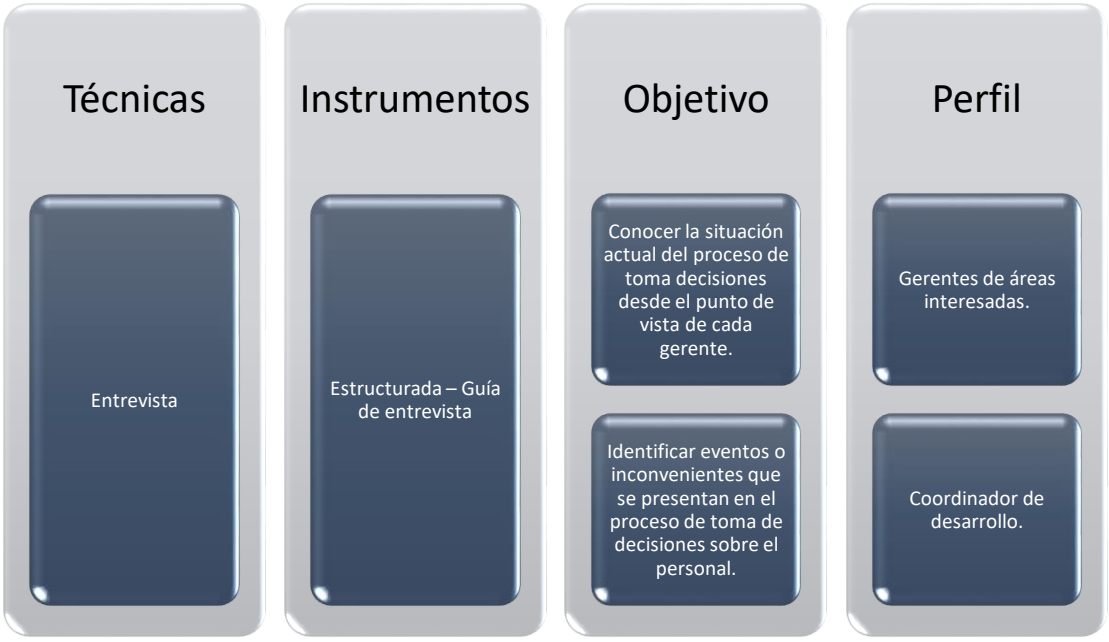


Figura 6 Técnicas e instrumentos de recopilación de datos

Fuente: Elaboración propia.

En la figura 6, se presenta las técnicas e instrumentos de recolección de datos para esta investigación, se detallan el tipo de técnica de la entrevista con el fin de obtener los datos, y posteriormente generar y procesar la información requerida.

3.3.1 Metodología

Tabla 1. Resumen de metodología de investigación

Plan de desarrollo y plan de implementación de un sistema de información gerencial de georeferenciación, basado en inteligencia de negocios y datos de geolocalización, para el personal ejecutivo de una empresa en el área de centro de atención telefónica en San Salvador

Objetivo general

Proponer un plan de desarrollo y un plan de implementación de un sistema de información gerencial de georeferenciación, basado en inteligencia de negocios y datos de geolocalización (inteligencia de ubicación) para el personal ejecutivo de una empresa en el área de centro de atención telefónica en San Salvador

Objetivos Específicos

- Describir componentes y características en la elaboración de sistemas de localización de personal integrando tecnologías de geolocalización e inteligencias de negocios.
- Optimizar el proceso actual de la toma de decisiones a través de un sistema de información gerencial de localización de personal para los ejecutivos de una empresa en el área de centro de atención telefónica.
- Construir un prototipo funcional de un módulo del sistema de información gerencial de localización de personal a partir del plan de desarrollo e implementación propuesto.

Variables y dimensiones

- **Independiente,** Inteligencia de negocios y datos de geolocalización.
- **Dependiente,** Proceso de toma de decisiones de la empresa.
- **Intervinientes** Cultura organizacional de la empresa

Metodología por emplear

Alcance de la investigación	Descriptiva
Tipo de investigación	Cualitativo
Técnica de investigación	Inductivo

Fuente: Elaboración propia

Capitulo 4. Solución propuesta

4.1 Plan de desarrollo

Esta sección del proyecto está enfocada, en la forma de trabajo que manejan en la empresa, en específico el área de desarrollo de software, esta aplica la metodología ágil SCRUM, la cual es la base para el plan de desarrollo, además, se propone un agregado para generar valor, incluyendo el modelo de capacidad de madurez integrado para el desarrollo (Capability Maturity Model Integration dev, CMMI dev).

Al ser dos tipos de metodologías de trabajo diferentes, se toma como guía el Manual para la implementación de CMMI a nivel de capacidad 2 con SCRUM en el área de desarrollo de software (García, N., Linares, J. 2020), donde presentan una manera de aprovechar las ventajas, no solo de agilizar la creación de software, sino también mejorar la eficiencia de los procesos involucrados.

4.1.1 Objetivos

4.1.1.1 General

Crear un plan de desarrollo para un sistema de información gerencial de georreferenciación, basado en inteligencia de negocio, aplicando SCRUM y CMMI a nivel de capacidad 1.

4.1.1.2 Específicos

1. Determinar los requerimientos de acuerdo con los productos y artefactos que se necesitan para la aplicación de SCRUM y CMMI.
2. Diseñar un esquema de infraestructura donde se aplique inteligencia de negocios en el manejo de los procesos de extracción, limpieza y carga.
3. Describir las fases y procesos para el desarrollo de este plan.

4.1.2 Alcances

El presente plan de desarrollo tiene como alcance su aplicación en tres departamentos de la empresa de atención al cliente, entre estos se encuentran el área administrativa, recursos humanos y operaciones.

4.1.3 Limites

Este proyecto de plan de desarrollo incluye la construcción de la propuesta, más no la ejecución de esta. Dicho plan será entregado a la empresa donde se está desarrollando la investigación, en donde posteriormente, dicha empresa lo implementará de forma interna.

4.1.4 Metodología

Como se mencionó al inicio de este apartado, se ha optado por utilizar SCRUM como metodología ágil para la construcción del proyecto y CMMI para la mejora en la eficiencia en los procesos del proyecto. La guía por seguir será el manual para la implementación de CMMI a nivel de capacidad 2 con SCRUM. En la tabla 3, se detalla los productos y procesos a tomar en cuenta para el plan de desarrollo.

4.1.4.1 Gestión de requisitos (REQM)

4.1.4.1.1 Practica especifica: Comprender requisitos

Algunos productos de trabajo que se producirán son:

- Listas de criterios para distinguir a los proveedores apropiados de requisitos.
- Tipificación de los requisitos.
- Criterios para la evaluación y la aceptación de los requisitos.
- Resultados del análisis frente a los criterios.
- Un conjunto de requisitos aprobados.

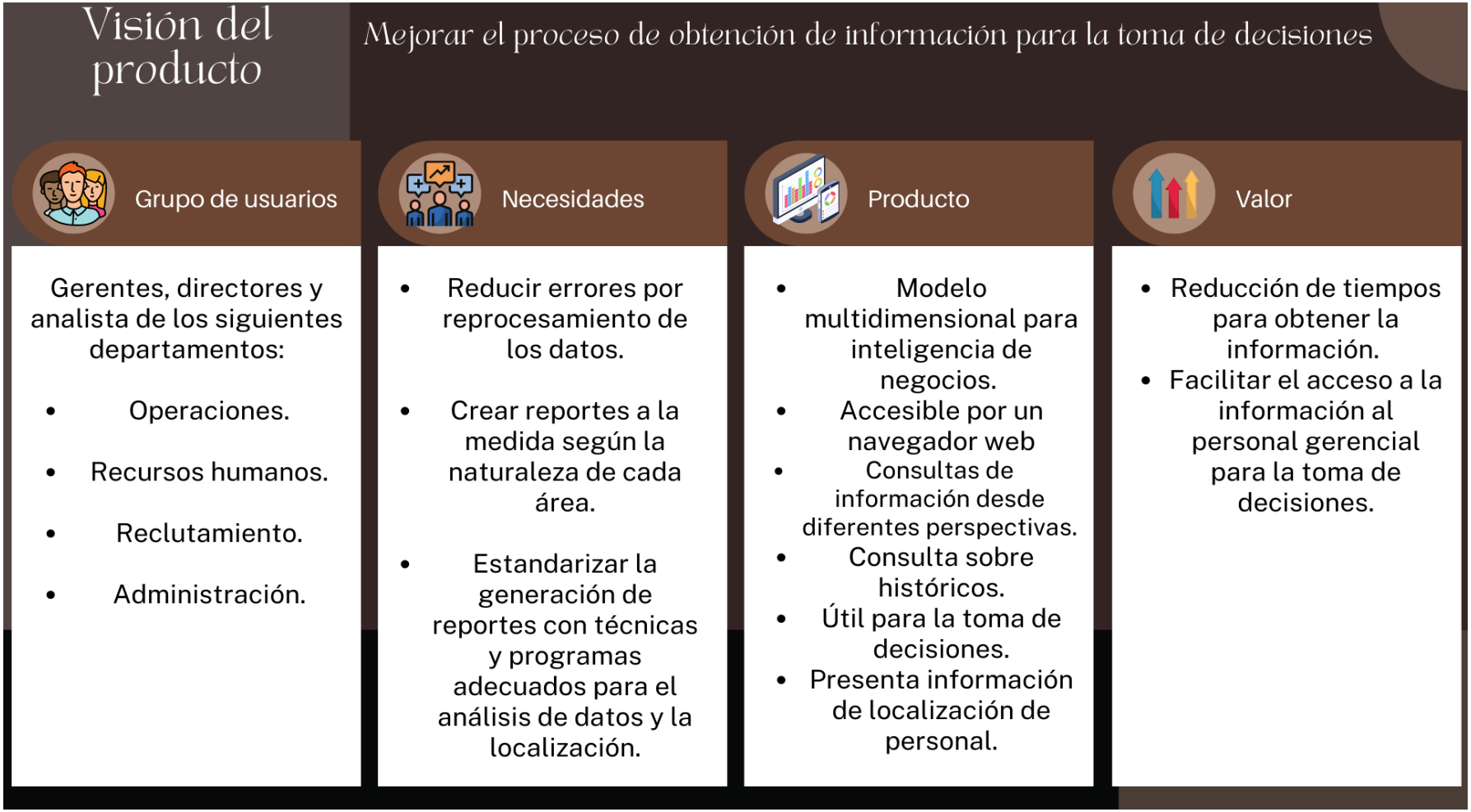
1. Visión del producto.

En esta sección se presenta la visión del producto, la cual brinda un enfoque conciso como un punto de partida y una ruta a seguir en el desarrollo del proyecto. Para poder determinar la visión del producto para este proyecto, se usará el Product Visión Board (PVB) (Pitcher 2017, citado por Hiromoto, 2017) como herramienta para mostrar la visión de forma fluida y puntos clave para entender y utilizar la visión. Este consta de 6 partes las cuales son: Nombre del proyecto, Frase representativa, Grupo de usuarios, Necesidades, Producto, Valor, tal como se muestra en la Figura 6.

Figura 7 Descripción de la visión del proyecto.

Fuente: Elaboración propia.

Tabla 2. Visión del producto



Fuente: Elaboración propia a partir de Hiromoto, 2017

Para plasmar la visión del producto de manera fluida, y utilizando la información que se tiene en el Product Vision Board, se implementará en base a la plantilla de Geoffrey Moore como se ve a continuación.

Tabla 3. Declaración visión del producto

PARA	Las direcciones y gerencias de la empresa de atención al cliente
QUIENES	Necesitan mejorar el proceso de obtención de información para la toma de decisiones
EL	Modelo multidimensional
ES	Una solución de inteligencia de negocios
QUE	Potenciará el proceso de obtención de información, gracias a su diseño orientado a la analítica de datos, utilizando la información generada por el sistema de administración, de incidencias y datos de geolocalización
A DIFERENCIA DE	Reportes estáticos generados por los sistemas actuales de la empresa
NUESTRO PRODUCTO	Brindará información útil para la toma de decisiones facilitando la analítica y la personalización de la información

Fuente: Elaboración propia

2. Estructura organizacional

Para este apartado se muestra la estructura organizacional para el plan de desarrollo y la asignación de roles para la aplicación de SRUM

Tabla 4. Definición de roles

Rol	Cargo
Stakeholder 1	Director de operaciones
Stakeholder 2	Directora de recursos humanos
Stakeholder 3	Gerente de reclutamiento
Product Owner	Gerente Administrativo
Scrum Master	Coordinador de desarrollo de software
Scrum team	Analista de desarrollo de software
Scrum team	Analista de desarrollo de software
Scrum team	Analista de desarrollo de software

Fuente: Elaboración propia

3. Desarrollo de épicas

A continuación, se detallan las principales actividades a ejecutar en cada Sprint.

Tabla 5. Actividades de desarrollo

No	Actividades
1	Refinamiento de requerimientos
2	Diseño del modelo multidimensional
3	Diseño del proceso ETL
4	Diseño de vistas
5	Creación de script SQL modelo multidimensional
6	Desarrollo del proceso ETL
7	Desarrollo de cubos OLAP
8	Desarrollo de vistas
9	Implementación e integración de componentes
10	Pruebas
11	Documentación

Fuente: Elaboración propia

Plan de lanzamiento

Para la ejecución del proyecto se ha decidido realizar un Sprint por departamento y se detalla el orden, duración de cada uno.

Tabla 6. Plan de lanzamiento

Release	Sprint	Área	Duración estimada (semanas)
Lanzamiento 1	Sprint 0	Investigación y preparación del entorno de desarrollo (servidores, tecnologías)	4
Lanzamiento 1	Sprint 1	Gerencia Administrativa	4
Lanzamiento 2	Sprint 2	Dirección de operaciones	4
Lanzamiento 2	Sprint 3	Dirección de recursos humanos	4

Fuente: Elaboración propia

Tabla 7. Historia épica 1

Historia épica #1	**Implementación del entorno de trabajo y arquitectura**	
Responsable	Analistas de desarrollo de software	Sprint: 0
Descripción	Como Administrador necesito: Diseñar la arquitectura de servidores a utilizar para el desarrollo del modelado multidimensional de las 3 áreas que forman parte del alcance del proyecto. Implementar dicho diseño para que el equipo pueda tener acceso a los recursos de dicha arquitectura.	
Criterios de aceptación	Deberá existir un servidor separado para fuente de datos. Los servidores deberán tener 1 dirección IP pública. Los servidores deberán poderse acceder de forma remota (Remote Desktop o SSH). Los servidores deberán poderse comunicar entre sí. Los servidores deberán tener los gestores de bases de datos instalados y configurados correctamente de forma que los servicios a utilizar puedan ser accedidos de forma remota a través de internet.	
Estimación	20	

Fuente: Elaboración propia

Tabla 8. Historia épica 2

Historia épica #2	**Modelo para Gerencia Administrativa**	
Responsable	Analistas de desarrollo de software	Sprint: 1
Descripción	Como gerente administrativo necesito: Un mecanismo para analizar la información de las disposiciones y localización del personal desde diferentes perspectivas.	
Criterios de aceptación	Las disposiciones y localización deben poder analizarse de las siguientes perspectivas: • Por cuenta • Por zona • Por disposición • Por fecha • Por empleado	
Estimación	20	

Fuente: Elaboración propia

Tabla 9. Historia épica 3

Historia épica #3	**Modelo para Dirección de Operaciones**	
Responsable	Analistas de desarrollo de software	Sprint: 2
Descripción	Como director de operaciones Necesito: Un mecanismo para analizar la información de eventos (incidencias, gestiones, entre otros) registrados por el personal de operaciones desde diferentes perspectivas. Para: la evaluación de resultados de dichos eventos en diferentes temporadas o períodos de tiempo y localización.	
Criterios de aceptación	Los eventos deben poder ser analizadas desde las siguientes perspectivas: • Por fecha • Por zona • Por tipo de evento • Por empleado. • Por cuenta. • Por cargo • Clasificación (Sector y Categoría) • Motivo	
Estimación	20	

Fuente: Elaboración propia

Tabla 10. Historia épica 4

Historia épica #4	**Modelo para Dirección de recursos humanos**	
Responsable	Analistas de desarrollo de software	Sprint: 3
Descripción	Como Dirección de Recursos Humanos necesito: Un mecanismo para analizar la cantidad de eventos registrados, las disposiciones y la localización del personal. Para: realizar evaluaciones comparativas de rendimiento del personal de operaciones.	
Criterios de aceptación	La cantidad de eventos registrados deben poder visualizar por las siguientes perspectivas: • Por fecha • Por lugar móvil • Por tipo evento • Por forma de recepción • Por cuenta • Por clasificación (Sector y Categoría) • Por motivo • Por zona • Por ubicación del empleado • Por empleado La cantidad de disposiciones junto a la cantidad de tiempo de los eventos ingresado, tiempo de duración del evento, deben poder analizarse desde las siguientes perspectivas: • Por fecha • Por zona • Por ubicación del empleado • Por empleado • Por clasificación (Sector y Categoría) • Por motivo • Por forma de recepción • Por cuenta • Por tipo evento • Por tipo de disposición	
Estimación	20	

Fuente: Elaboración propia

4.1.5 Diseño

4.1.5.1 Diseño lógico de la base de datos

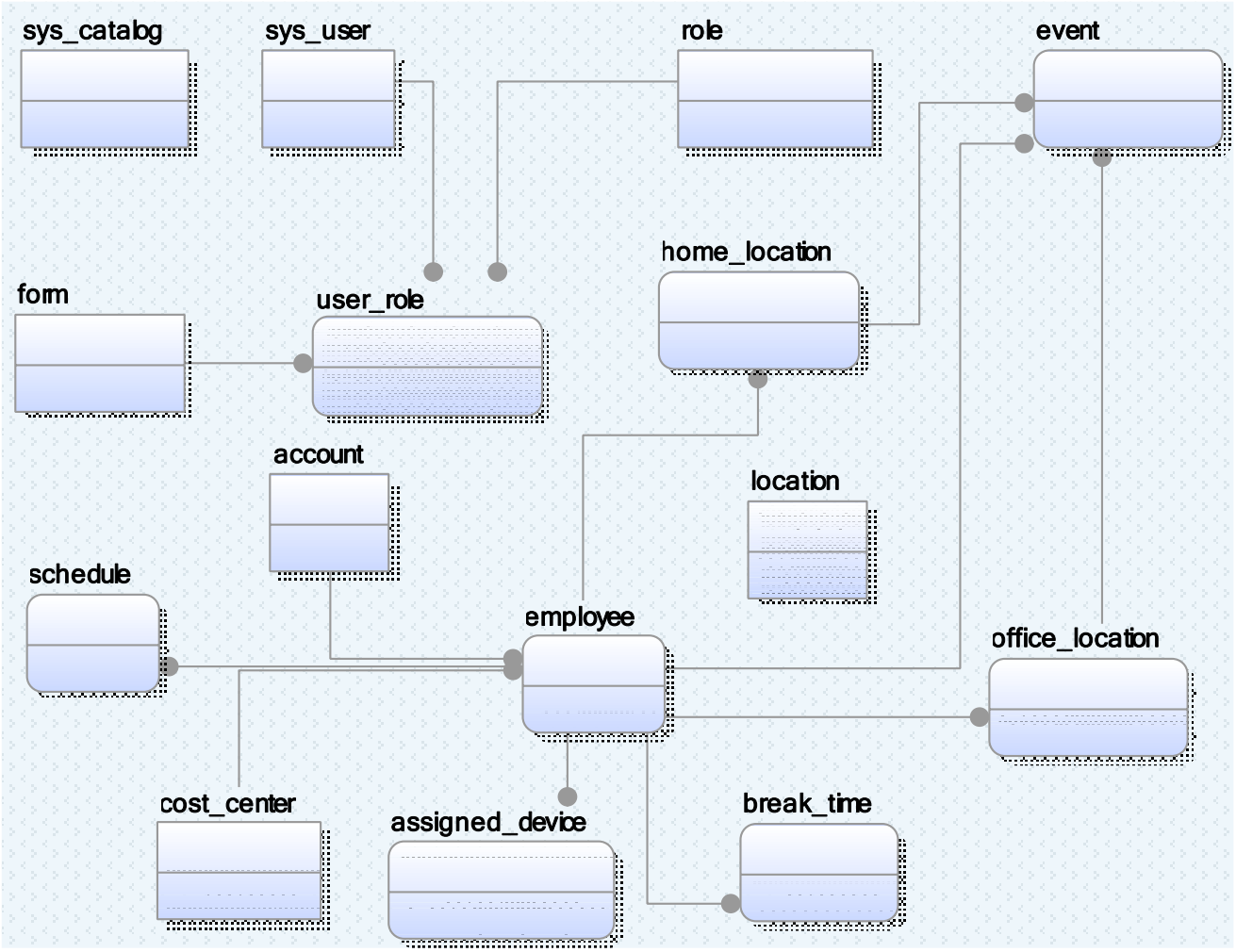


Figura 8 Modelo lógico de la base de datos

Fuente: Elaboración propia

En la figura 8, se presenta a alto nivel el modelo lógico de la base de datos, en donde se muestran las entidades que participaran en el modelo y sus respectivas relaciones.

4.1.5.2 Diseño físico de la base de datos

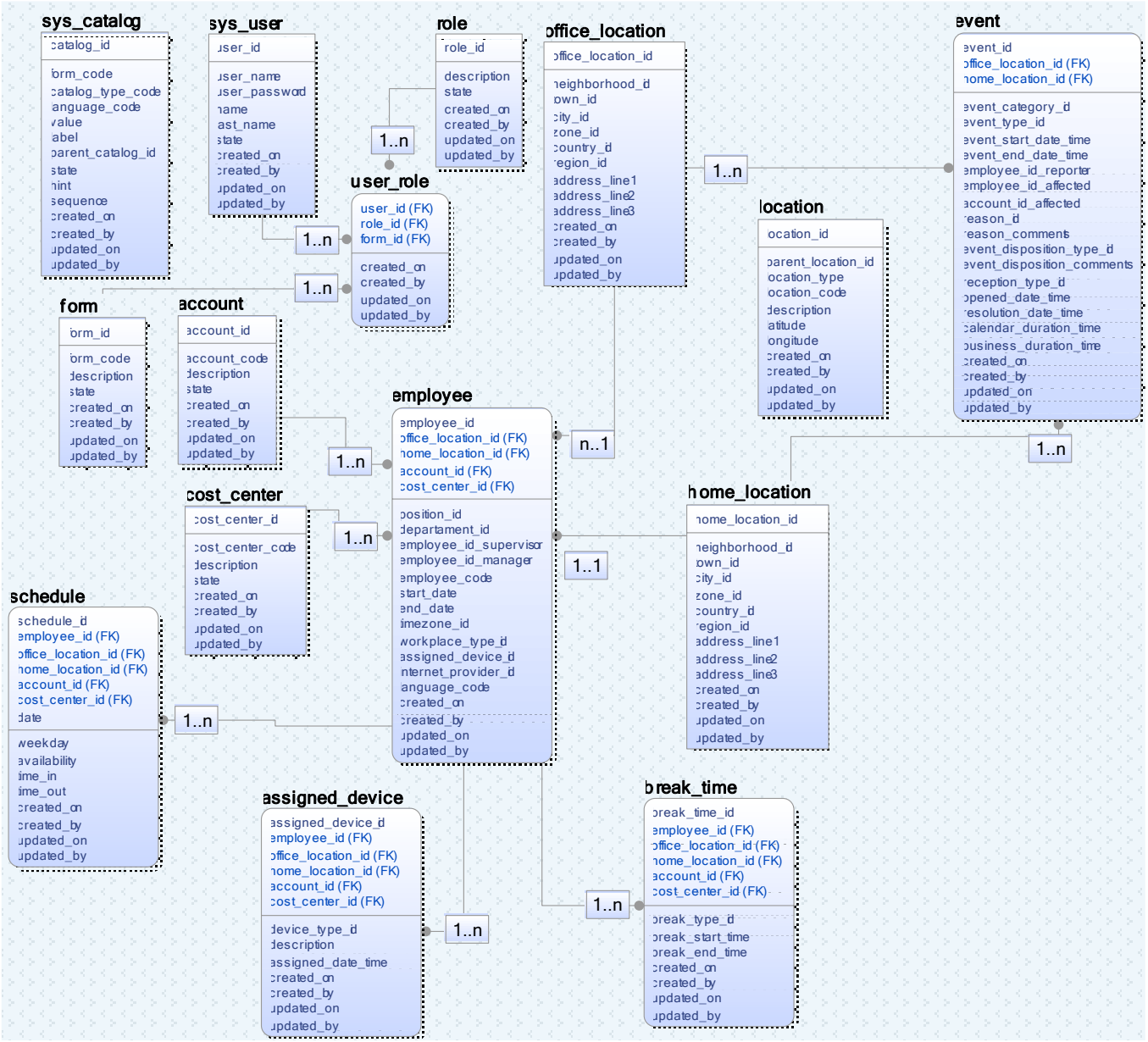


Figura 9 Modelo físico de la base de datos

Fuente: Elaboración propia

En la figura 9, se presenta el modelo físico de la base de datos, en donde se muestran las tablas a implementar, campos, y relaciones entre las mismas.

4.1.5.3 Listado de servicios o interfaces entre sistemas

Para realizar una correcta integración, hay que definir primero que sistemas son los que estarán incluyendo, con el fin de determinar qué elementos son necesarios para realizar una correcta implementación.

Como primer paso, se debe definir de qué lugar se obtendrán los datos, es decir la fuente. Se tomará información directamente de 3 bases de datos por medio de conexión directa utilizando un usuario especial que se utilizará únicamente para la migración. Las bases de datos son:

Recursos Humanos (RRHH): De esta base de datos se obtendrá la información de los agentes que están trabajando remoto, por ejemplo, su ubicación, nombre y otra información útil de contacto

Operación: Esta base de datos contiene la información operativa por agente, dicha información es importante para determinar indicadores respecto a la disponibilidad y niveles de servicio del agente.

Incidentes: Esta base de datos registra todos los detalles sobre reportes de fallos relacionados con incidentes que afectan o impiden que un agente que trabaja remoto pueda realizar sus labores diarias, como por ejemplo fallas de internet, inundaciones, fallos eléctricos entre otros. Y tener un registro del tipo y frecuencia de incidentes de los agentes.

Ya que se necesita crear una base de datos cubo para poder almacenar la información de los agentes cada cierto tiempo, se debe contar con un proceso que lea la información de todas las bases de datos y, al tener todos los datos, organice la información y proceda a insertar los datos obtenidos en la base de datos del cubo, todo esto se planea hacer inicialmente por JDBC pero puede utilizarse otros métodos disponibles en C#, el cual será el lenguaje a utilizar debido a ser un lenguaje de programación afín con los productos de Microsoft ya que la empresa ya maneja este tipo de productos.

4.1.5.4 Diseño de integración de servicios

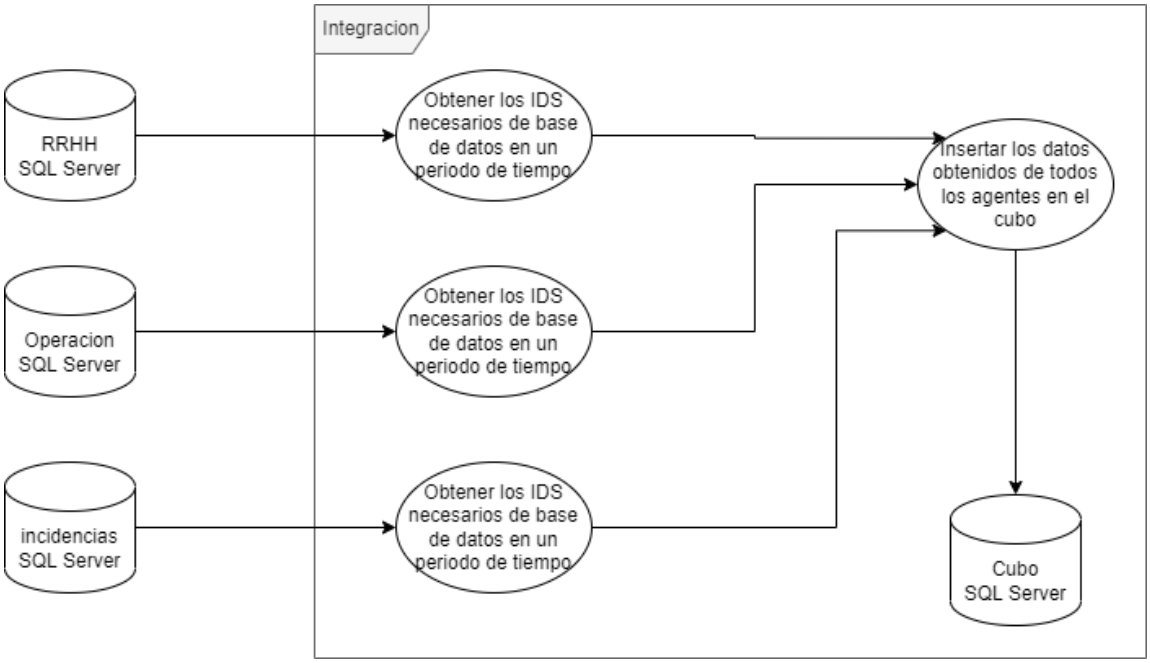


Figura 10 Diagrama de integración de sistemas

Fuente: Elaboración propia

En la figura 10, se logra apreciar de una manera grafica como se realizará la integración de los sistemas, debido a que es necesario crear un cubo que ayude a agilizar la lectura de datos en Power BI. La idea principal del proceso de integración en C#, es poder crear un proceso automático que ejecute el proceso cada cierto tiempo, con el fin de obtener los datos actualizados y que la información sea lo más cercana en tiempo real.

4.1.5.5 Diagrama de componentes de software y aplicaciones involucradas

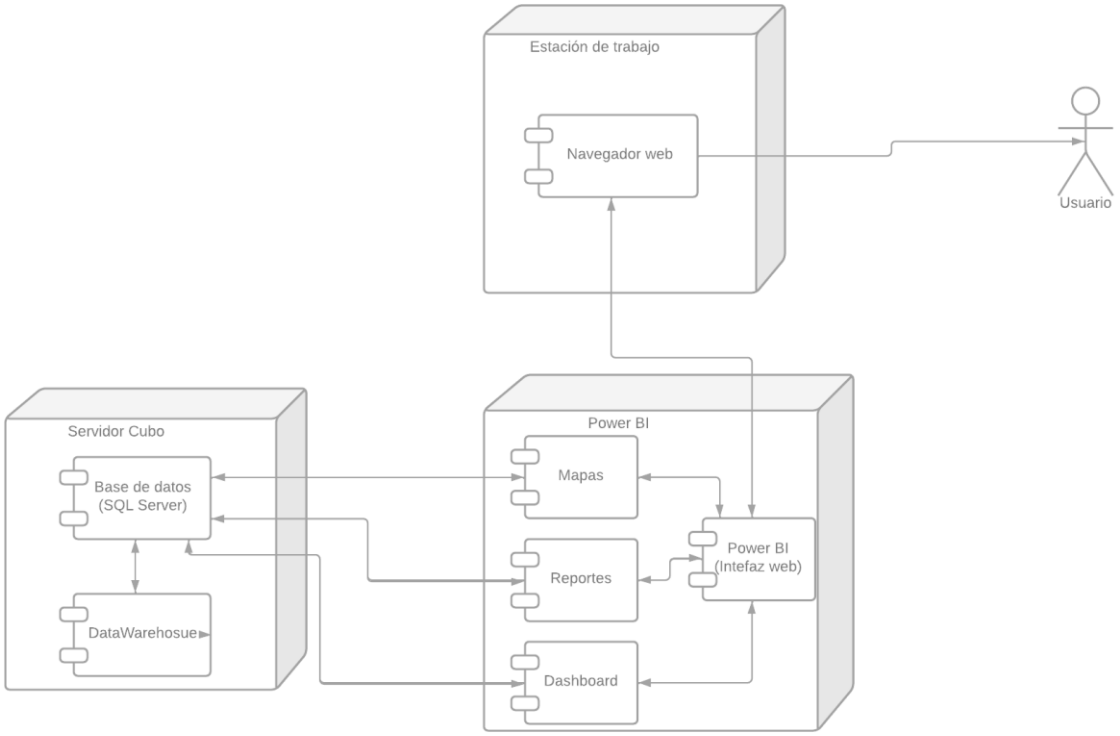


Figura 11 Diagrama de componentes.

Fuente: Elaboración propia

Como se puede apreciar en la Figura 11, se presenta el diagrama de componentes de software, esta consta del servidor donde configurará el cubo con el gestor de base de datos de SQL Server, obteniendo los datos desde el DataWarehouse, a partir de acá se agrupará los datos para generar información con valor que serán utilizados por los componentes de mapas, dashboard y reportes para su presentación y uso a través de un interfaz web.

4.1.5.6 Diseño de interfaz de usuario

En esta sección se presenta el diseño de interfaz de usuario, cabe aclarar que el usuario va a interactuar con la aplicación Power BI en el espacio de trabajo online para usuarios finales. A partir de ello, los mockups o pantallas que se presenten a continuación son ejemplos de lo que se espera realizar con este proyecto. Dashboard.

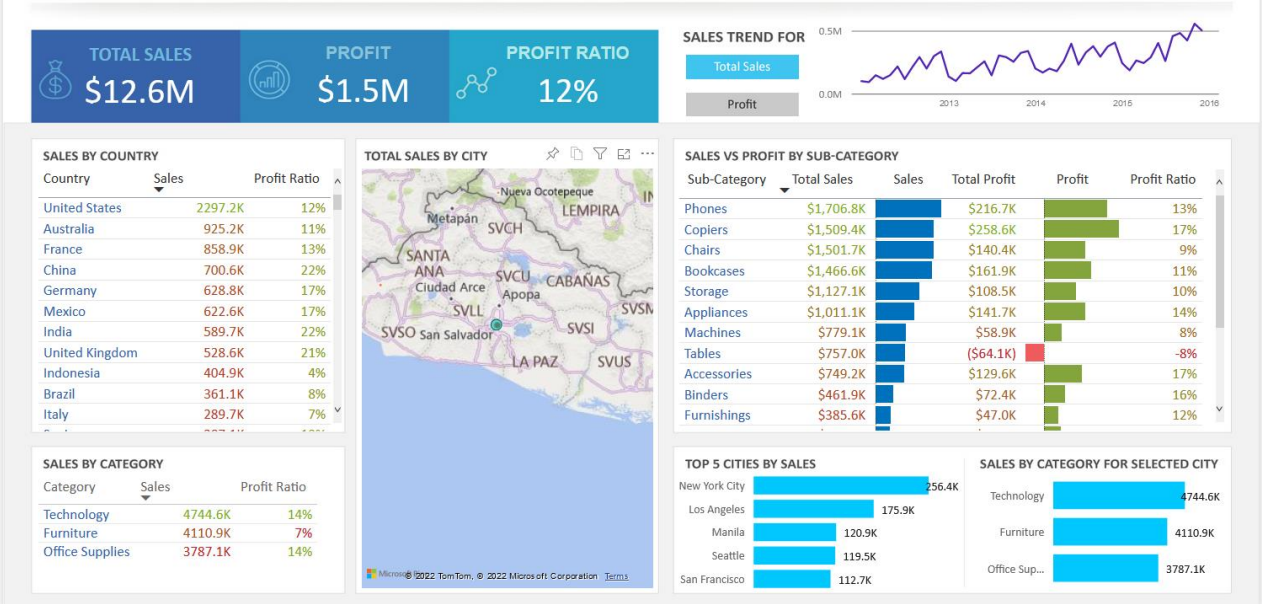


Figura 12 Dashboard general.

Fuente: Elaboración propia

Como se puede observar en la figura 12, se pretende visualizar de forma general en un dashboard gráficas, métricas y mapas para la presentación de la información.

Mapas.

Figura 13. Mapas

Fuente: Elaboración propia

En la figura 13 se muestran los mapas usados como la forma de presentar la información de localización que se procese en la empresa relacionada a eventos, dispositivos, entre otros.

Métricas.

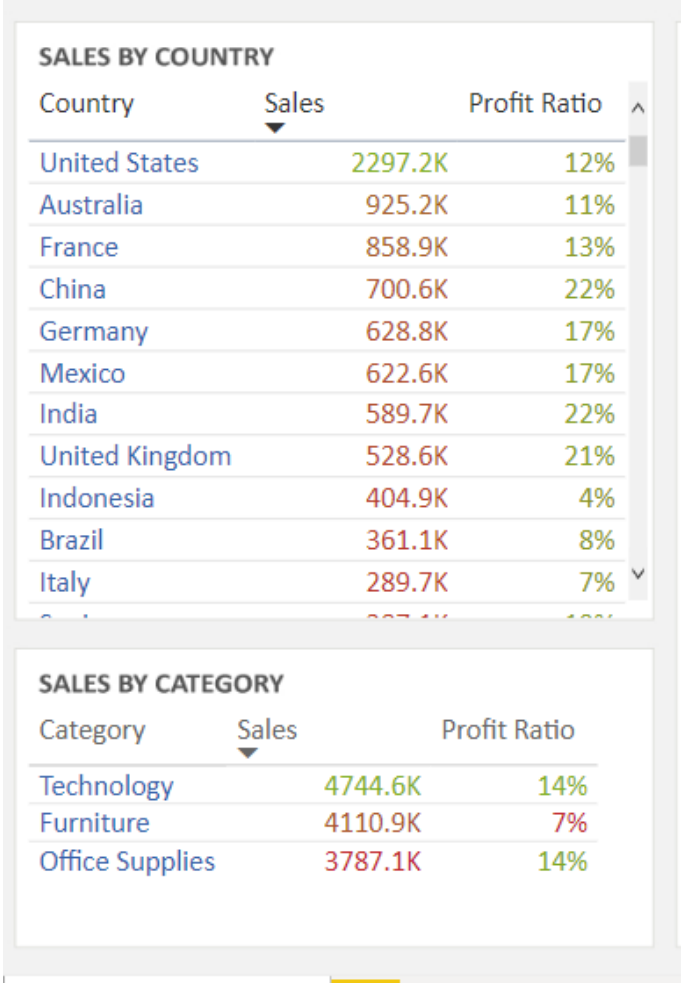


Figura 14. Métricas

Fuente: Elaboración propia

En la figura 14 se muestra una presentación de métricas a través de tablas donde se visualice las cantidades y porcentajes de forma oportuna para una mejor toma de decisiones.

Gráficas.

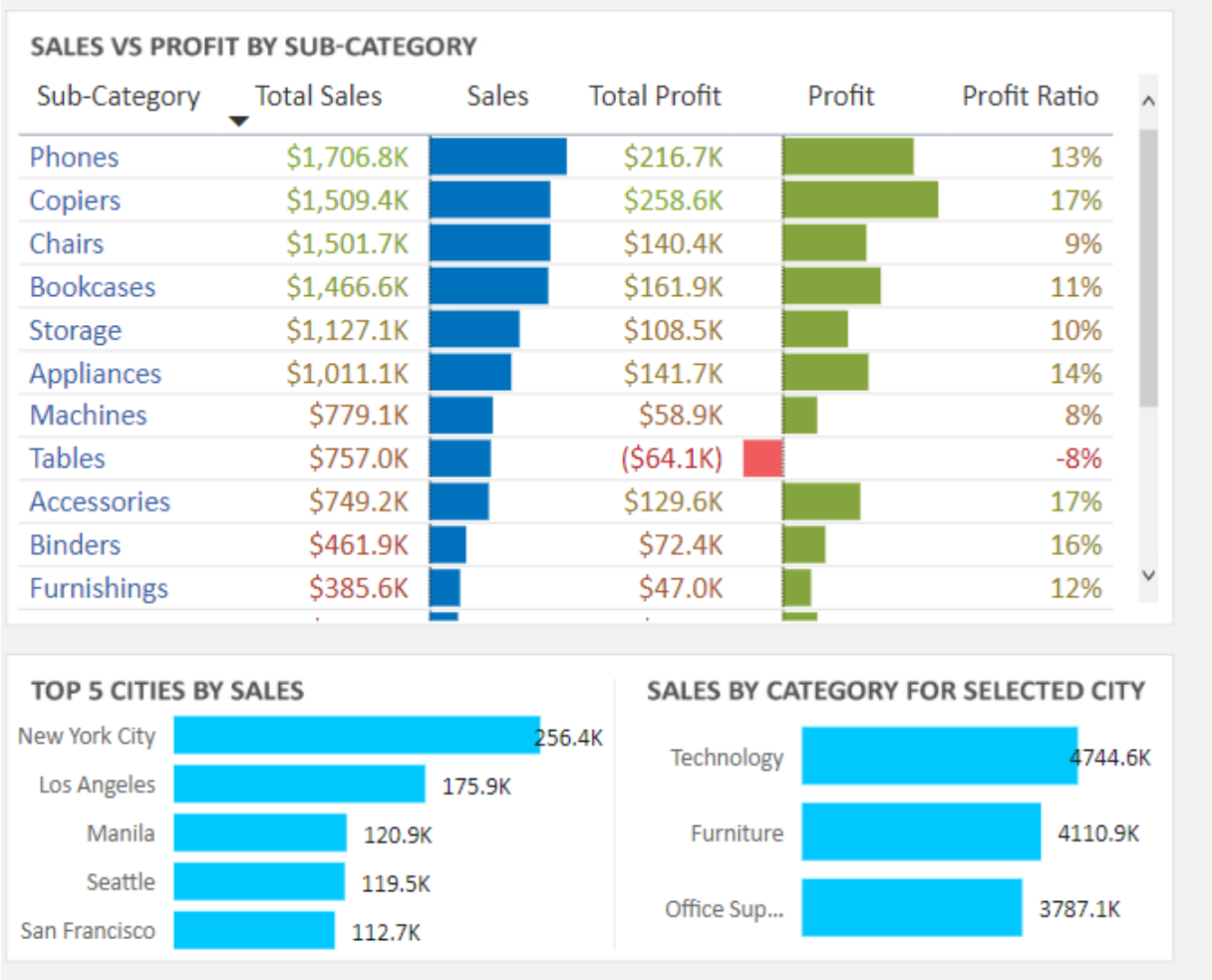


Figura 15. Gráficas

Fuente: Elaboración propia

En la figura 15 se muestra como el uso de gráficas para la presentación de información brinda un mayor entendimiento para la interpretación del tema que se presenta.

4.1.6 Construcción

4.1.6.1 Sistema de información gerencial basado en georreferenciación

En el diagrama de la figura 16 se logra apreciar el enfoque general del sistema a desarrollar:

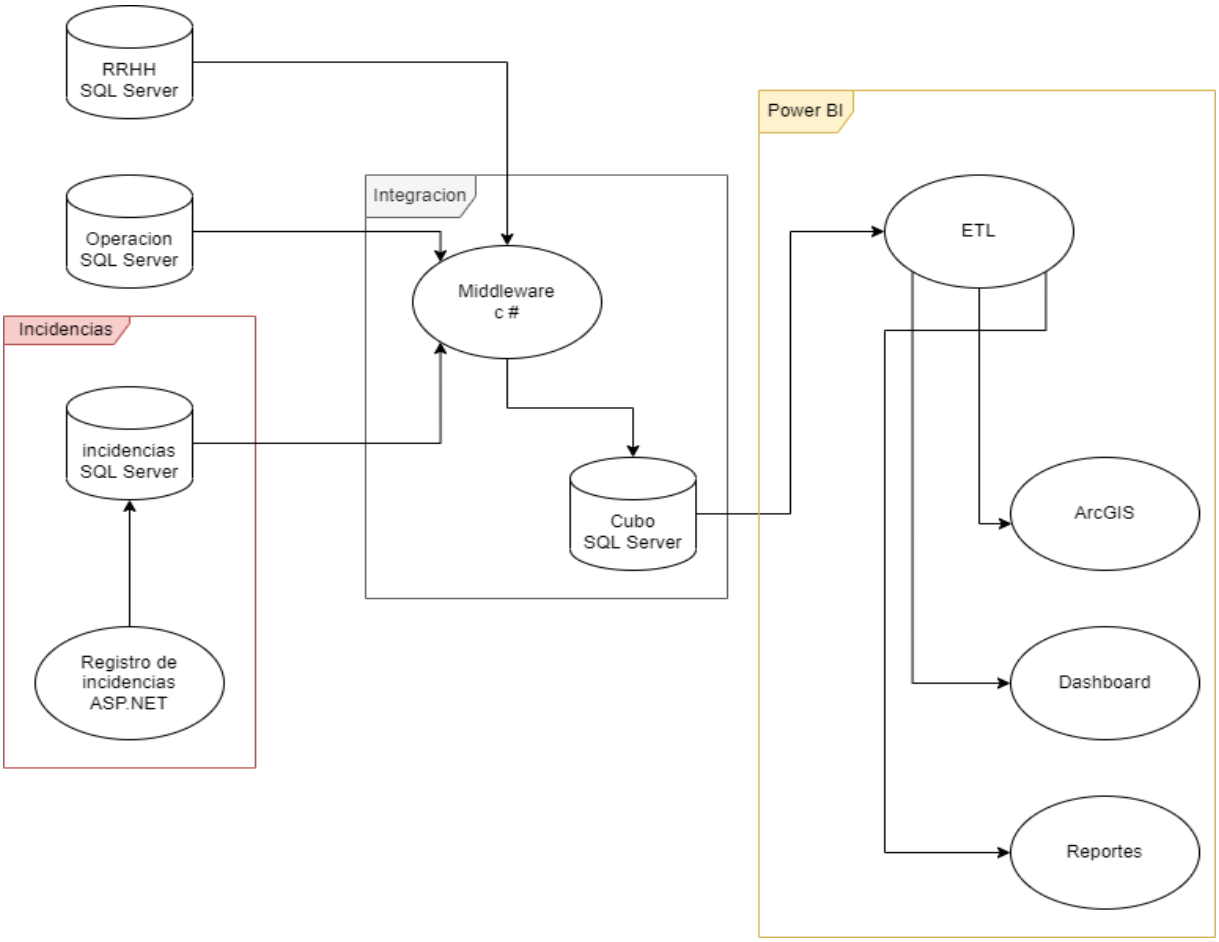


Figura 16 Diagrama de flujo de desarrollo

Fuente: Elaboración propia

El sistema de información gerencial basado en georreferenciación consta de varias secciones y depende de recursos de bases de datos existentes que contienen la información de los agentes, en cuestiones de ubicación e información sobre su desempeño en la operación.

El sistema de registro de incidencias tiene la intención de registrar cualquier evento que pueda ocurrir a los agentes que pueda afectar la operatividad de los agentes, principalmente los que están en modalidad teletrabajo, esto brindará a la empresa la oportunidad de conocer los tipos de incidentes que suelen inhabilitar a los empleados y también la frecuencia con que dichos eventos ocurren. En el diagrama de la figura 17 se puede apreciar lo que abarca la funcionalidad del sistema de integración:

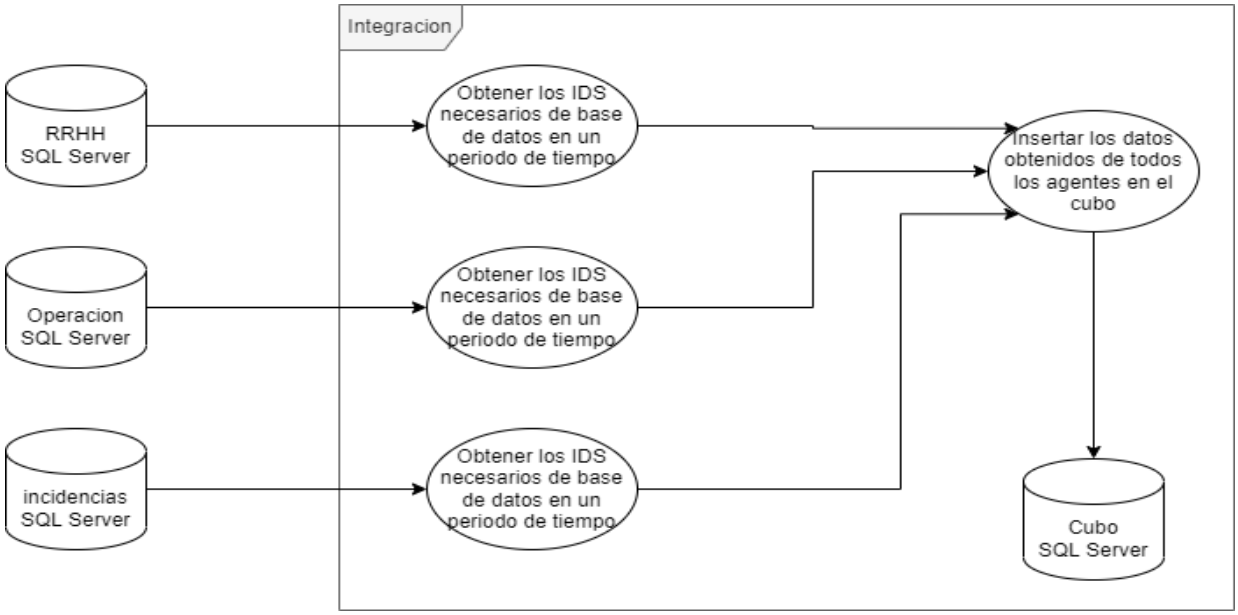


Figura 17 Diagrama de flujo de sistema de registro de incidencias.

Fuente: Elaboración propia

4.2 Plan de implementación

4.2.1 Objetivo

Establecer el plan de implementación de un sistema de información gerencial de georreferenciación, basado en inteligencia de negocios, para el personal ejecutivo de una empresa en el área de centro de atención telefónica.

4.2.2 Organización

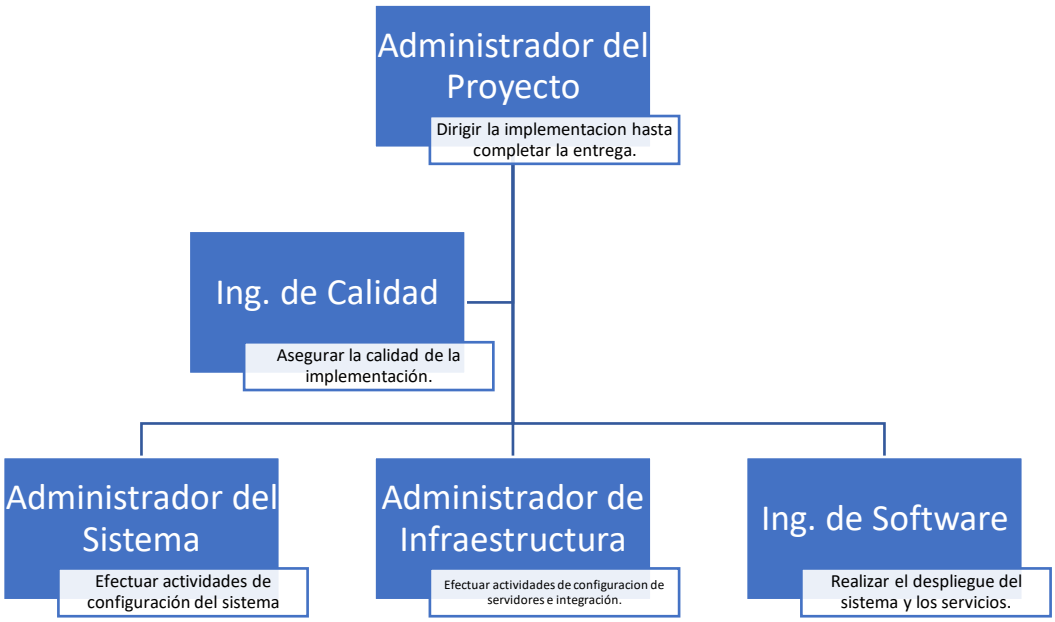


Figura 18. Organización.

Fuente: Elaboración propia

4.2.3 Actividades

Tabla 11. Actividades de implementación.

No	Actividad	Responsable
1	Configuración de servidores e integración	Administrador de Infraestructura
2	Despliegue del sistema	Ing. de Software
3	Configuración del sistema	Administrador del Sistema
4	Habilitación de los servicios de integración	Ing. de Software
5	Asegurar la calidad de la implementación	Ing. de Calidad

Fuente: Elaboración propia

4.2.4 Aplicación del modelo CMMI

Esta implementación persigue la meta de colaborar con la organización a aumentar la productividad de su personal operativo de atención de llamadas telefónicas y por diferentes medios electrónicos de comunicación, reduciendo de esta manera el riesgo de fallas en la operación, sin agregar esfuerzos innecesarios.

Maruthi (CMMI-DEV Implementation Simplified, 2019), denota que el modelo CMMI se considera un proyecto largo que podría llevar de 6 a 8 meses, y sus implementaciones son riesgosas para los negocios. Por lo tanto, para resolver los problemas antes mencionados, Maruthi (2019) recomienda el modelo en espiral para implementar CMMI para las

organizaciones que están dispuestas a implementar CMMI por su propia cuenta, tal como se muestra en la siguiente figura:

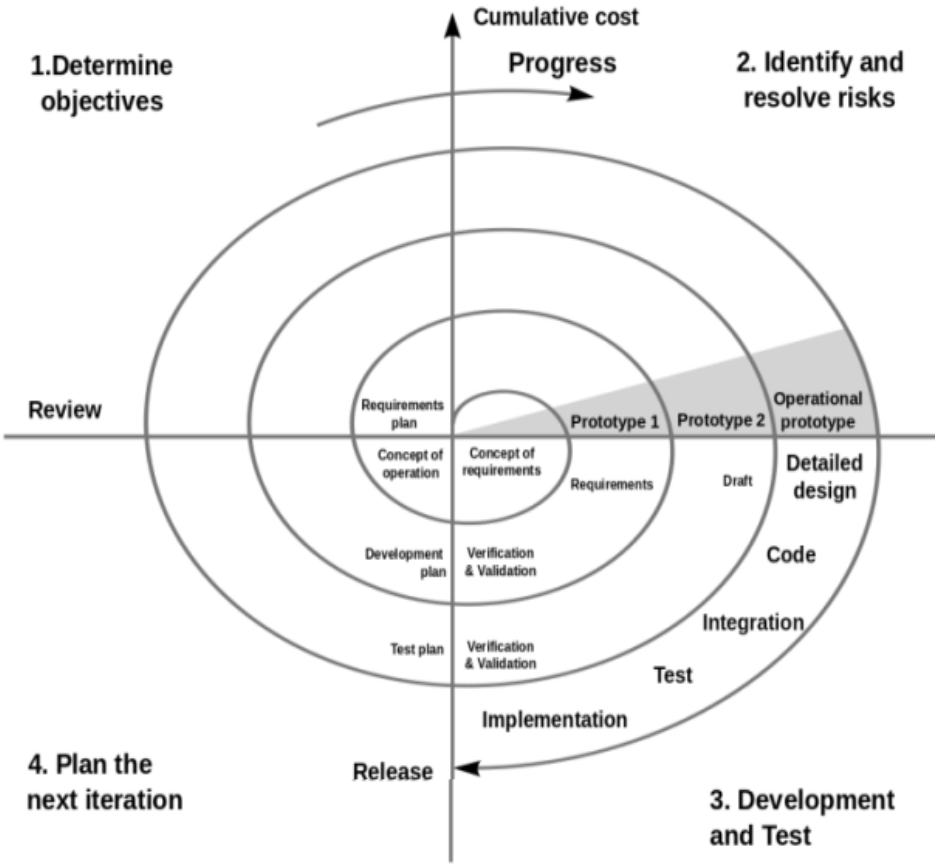


Figura 19 Bohem Spiral Software Model.

Fuente: Maruthi (2019) CMMI-DEV Implementation Simplified

A continuación, se brinda una explicación de cada uno de los elementos ilustrados en el modelo de software espiral, mostrados en la figura 19:

Review: Revisiones funcionales y técnicas.

1. Determine objetives: Se refiere a determinar los objetivos de la iteración.
    a. Requeriments plan: Plan de requerimientos sobre el cual se basa el desarrollo.

   Cumulative cost: Costo acumulado por todas las iteraciones.

   Progress: Progreso logrado el cual también es acumulativo.
2. Identify and resolve risks: Identificar y resolver los riesgos con el objeto de abordarlos progresivamente por cada iteración.
    a. Prototype 1. Prototipo 1, permite al usuario final tener una idea anticipada del software a desarrollar, con el fin de identificar y resolver riesgos.
    b. Prototype 2. Prototipo 2, es la evolución del prototipo 1.

c. Operational prototype: Prototipo operacional, es cuando se logra modelar la funcionalidad que estará en operación.

3. Development and Test: Desarrollo y prueba del componente de software de la respectiva iteración.
    a. Detailed design: Diseño detallado
    b. Code: Codificar el diseño
    c. Integration: Integración de funcionalidades
    d. Test: Pruebas del código desarrollado
    e. Implementation: Implementación de funcionalidades

    Draf: Borrador

    Verification & Validation; Verificación del equipo de desarrollo y validación del usuario final.

    Requeriments: Requerimientos del software.

    Concept of requeriments: Concepto de requerimientos.

    Release: Liberación a los ambientes productivos

4. Plan the next iteration: Planear la siguiente iteración, con el objeto de construir de manera gradual el producto en su totalidad.
    a. Test plan: Plan de pruebas para la siguiente iteración.
    b. Development plan: Plan de desarrollo de la siguiente iteración.
    c. Concept of operation: Concepto de operación de la siguiente iteración.

Del amplio conjunto de áreas de capacidad descritas por del modelo CMMI (CMMI Institute, 2018), a continuación, se muestran tres que contribuyen de forma positiva a que la fase de implementación sea un éxito para todo tipo de proyecto:

Garantizar la calidad (ENQ)

Esta área de capacidad incluye áreas de práctica importantes para el aseguramiento y el control de calidad. El área de capacidad garantizar la calidad incluye las siguientes áreas de prácticas:

1. Desarrollo y gestión de requisitos (RDM)

    Garantiza que se satisfagan las necesidades y expectativas de los clientes. Esta área de práctica consta de 3 niveles de maduración.
2. Aseguramiento de la calidad del proceso (PQA)

    Incrementa el uso consistente y la mejora de los procesos para maximizar el beneficio de negocio y la satisfacción del cliente. Esta área de práctica consta de 3 niveles de maduración.
3. Verificación y validación (VV)

La verificación y validación, a lo largo del proyecto, de las soluciones y componentes seleccionados al efecto, incrementará la probabilidad de que la solución satisfaga al cliente. Esta área de práctica consta de 3 niveles de maduración.

4. Revisiones por pares (PR)

   Reducir los costos y la reelaboración al descubrir temprano problemas o defectos. Esta área de práctica consta de 3 niveles de maduración.

Apoyar la implementación (SI)

Esta área de capacidad implica identificar y abordar las causas de los resultados seleccionados, la creación de una estructura y un enfoque de toma de decisiones, preservación de la integridad de los productos de trabajo y el fomento de la comunicación y coordinación entre los grupos involucrados.

1. Análisis de la causa y resolución (CAR)

   La solución de problemas de causa raíz elimina retrabajo y directamente mejora la calidad y la productividad. Esta área de práctica consta de 5 niveles de maduración.

2. Análisis de decisiones y resolución (DAR)

   Aumenta la objetividad de la toma de decisiones y la probabilidad de seleccionar la solución óptima. Esta área de práctica consta de 3 niveles de maduración.

3. Gestión de la configuración (CM)

   Reduce la pérdida de trabajo y aumenta la capacidad para ofrecer la versión correcta de la solución al cliente. Esta área de práctica consta de 2 niveles de maduración.

Sostener el hábito y la persistencia (SHP)

Esta área de capacidad garantiza que los procesos sean persistentes y se realicen y mantengan habitualmente en toda la organización y que estén contribuyendo efectivamente a cumplir con los objetivos de rendimiento del negocio.

1. Gobernanza (GOV)

   Minimiza el costo de implementación de proceso, aumenta la probabilidad de cumplir los objetivos y garantiza que los procesos implementados apoyen y contribuyan al éxito del negocio. Esta área de práctica consta de 4 niveles de maduración.

2. Infraestructura de implementación (II)

   Mantiene la capacidad de alcanzar consecuentemente metas y objetivos de manera eficiente y eficaz. Esta área de práctica consta de 3 niveles de maduración.

Para efectos de esta implementación, se presenta más en detalle la gestión de la configuración (CM), debido a que se considera el proceso más operativo y clave del control de cambios, para esta área existen las siguientes practicas especificas recomendadas:

1. Realizar el control de versiones.

   Los cambios a la configuración en los productos son comunes en la industria del software. Principalmente, porque, las empresas con el paso del tiempo comprenden de mejor manera su modelo de negocio. Por lo que, cambian sus requerimientos para adaptarse al contexto en que se desempeñan. Este hecho afecta de manera directa a la configuración.
2. Identificar los artículos a ser incluidos en la gestión de la configuración.

   Los requerimientos al ser aprobados, los cambios sobre ellos deben seguir un proceso definido de control de cambios, entonces, estos requerimientos se convierten en un elemento de la configuración; y pasan a ser parte de la línea base.

   Los elementos de la configuración identificados en la metodología SCRUM son:

   a) Las historias de usuario
   b) Los sprints (iteraciones de desarrollo)
   c) Las tareas
   d) Las incidencias
   e) Los códigos fuente
   f) Pruebas
   g) Retrospectivas
   h) Modelos

   Luego, los elementos de la configuración son organizados y planificados; la trazabilidad de los elementos de la configuración será mantenida por la estructura y jerarquía establecida por la empresa.

   Las organizaciones pueden contar con los elementos de la configuración definidos por la metodología SCRUM. Se debe tomar en cuenta la importancia de la automatización del proceso de gestión de la configuración, proporcionado por herramientas basadas, por ejemplo, en GIT, caso contrario entraría en conflicto con los valores propuestos por la metodología ágil.

3. Desarrollar, actualizar y usar un sistema de configuración y de Gestión del cambio.

   En este punto la posibilidad de integración con otros sistemas externos también es importante. Entre algunos de los sistemas de uso frecuente, que se recomienda en las organizaciones son: Jira y Travis.

4. Desarrollar o liberar líneas base para uso interno o para su entrega al cliente.

   En el desarrollo software, la comprensión de los conceptos de alcance, costo y tiempo es muy importante para la gestión de un proyecto. El Project Management Institute (2017) señala que las líneas base mantienen una relación directa a la planificación u hojas de ruta del proyecto. Por lo que, la falta de esta práctica

compromete el desarrollo de un proyecto de software exitoso. En consecuencia, tres factores evidencian el desarrollo de líneas base son sus productos de trabajo como son:

a) Su uso interno o externo.
b) Realización de hojas de ruta.
c) La estructura de desglose de trabajo (EDT).

5. Gestionar los cambios en los elementos de la gestión de la configuración.

   Los cambios en el software son frecuentes. Normalmente impactan en el desarrollo y operaciones de la organización. Esto implica que una coordinación y coreográfica de un proceso de cambio son necesarios para gestionar los cambios a los elementos de la configuración y solo los cambios establecidos sean cumplidos.

   Como está práctica es muy importante en cualquier organización, por esta razón, muchas de ellas, establecen algún tipo de proceso o mecanismo para atender los cambios a los requerimientos del software. Estos mecanismos incluyen: Uso de correos electrónicos, hasta sistemas de tiquetes.

6. Desarrollar, mantener actualizados y utilizar registros que describen los elementos de la gestión de la configuración.

   El estado de la configuración requiere la actualización y mantenimiento de registro que describan los elementos de la configuración, para revisar y reportar los activos bajo la gestión de la configuración del proyecto (IEEE 828, 2012). En la misma norma se recomienda que los elementos de la configuración deben contener la siguiente información (CMMI 2.0, 2018):

   - Fecha de origen
   - Localización
   - Versión aprobada
   - Estado
   - Trazabilidad

   De tal modo que, cuando las organizaciones logran el cumplimiento de esta práctica, son capaces de realizar reportes en base a la planificación y la información de los elementos de la configuración; por lo que, mantienen mecanismos y medios para la extracción de información, entre ellos:

   - Revisiones y registro de los elementos
   - Registro de solicitud de cambio
   - Estado de elementos de la configuración
   - Revisión de las líneas base

7. Realizar auditorías de configuración para mantener la integridad de las líneas base de configuración, cambios y contenido del sistema de gestión de la configuración.

Los contextos ágiles de gestión de proyectos de software han causado un gran impacto en el desarrollo de software.

El incremento de la frecuencia de liberaciones de software incluye mayor incremento de auditorías, revisiones o inspecciones por el equipo de desarrollo, que garanticen la calidad del software.

En la siguiente tabla, se muestra un ejemplo de la matriz de formato de evaluación del área de práctica "Gestión de la configuración (CM)", en la cual se debe marcar el cumplimiento de práctica específica acorde a las evidencias que lo comprueben (se debe indicar si: cumple / no cumple / cumple parcialmente)

Tabla 12. Ejemplo de evaluación de prácticas específicas "Gestión de la configuración (CM)"

Prácticas Específicas	Cumple	Parcial	No cumple	No Aplica	Documento de Trabajo
Realizar el control de versiones.	x				Team Foundation Version Control (TFVC)
Identificar los artículos a ser incluidos en la Gestión de la configuración.		x			Sprint
Desarrollar, actualizar y usar un sistema de configuración y de Gestión del cambio.	x				Azure DevOps
Desarrollar o liberar líneas base para uso interno o para su entrega al cliente.	x				Cronograma
Gestionar los cambios en los elementos de la gestión de la configuración.		x			Sistema de tickets
Desarrollar, mantener actualizados y utilizar registros que describen los elementos de la gestión de la configuración.		x			Solicitudes de cambio

Realizar auditorías de configuración para mantener la integridad de las líneas base de configuración, cambios y contenido del sistema de gestión de la configuración.	x		Informe de auditoria

Fuente: Elaboración propia

Se propone que en la etapa de implementación se logre alcanzar el nivel 2 de madurez para esta área de capacidad.

Criterios de asignación y argumentos para dicho grado de madurez o nivel de capacidad establecido al área de proceso.

Los procesos de estimación suelen ser los utilizados comúnmente en las empresas, pero tiene oportunidades de mejora.

A continuación, se presenta un ejemplo de cuadro resumen de la evaluación de las áreas de práctica, en el cual se denota el nivel actual de madurez, el nivel al que se quiere llegar, y el nivel máximo que permite cada área de práctica.

Tabla 13. Ejemplo de cuadro resumen de niveles de madurez según área de práctica

Área de Práctica	Nivel Actual	Nivel Propuesto	Nivel Máximo
Garantizar la calidad (ENQ)			
a. Desarrollo y gestión de requisitos (RDM)	1	3	3
b. Aseguramiento de la calidad del proceso (PQA)	1	3	3
c. Verificación y validación (VV)	1	3	3
d. Revisiones por pares (PR)	1	2	3
Apoyar la implementación (SI)			
a. Análisis de la causa y resolución (CAR)	1	3	5
b. Análisis de decisiones y resolución (DAR)	1	3	3
c. Gestión de la configuración (CM)	1	2	2

	Área de Práctica	Nivel Actual	Nivel Propuesto	Nivel Máximo
	Sostener el hábito y la persistencia (SHP)			
a.	**Gobernanza (GOV)**	1	3	4
b.	**Infraestructura de implementación (II)**	1	3	3

Fuente: Elaboración Propia

4.2.5 Gestión de modificación y errores

El control de cambios emergentes y errores se debe llevar a través de una bitácora de seguimiento, controlando el estado de los cambios y errores durante la implementación del proyecto, en caso de que algunos de ellos sean aplicables.

Para un eficiente manejo de cambios, es necesario tener un patrón sobre el cual basarse, este debe haber sido definido desde el análisis de requerimientos. Los cambios emergentes son una situación que es casi inevitable y es importante solventar este tipo de situaciones de manera óptima, para ello se deben considerar los siguientes aspectos generales:

- El administrador del proyecto y el dueño del producto deben analizar la solicitud de cambio brindada por los interesados, y con la ayuda del equipo técnico de operaciones, determinar el alcance y el tiempo que llevaría realizar el cambio, además de determinar los componentes que serían afectados por dicho cambio.
- El administrador del proyecto debe valorar el costo que el cambio tendría, ya que, dependiendo de la complejidad y el tiempo estimado, así sería de costoso implementar un cambio.
- Luego de recopilar la información, los analistas de sistemas deben revisar la modificación y documentar el cambio solicitado.
- Si el cambio es aprobado por el dueño del producto, se debe planificar un proceso de desarrollo para incluir este cambio en la implementación, luego se debe notificar a los desarrolladores de que un nuevo cambio emergente ha sido considerado en el proyecto y luego el equipo de aseguramiento de calidad debe validar que los cambios se han realizado y que no hay repercusiones negativas en el funcionamiento del sistema y sus interfaces. En caso el cambio sea aprobado o rechazado, esto se debe informar a los interesados una vez tomada la decisión y realizado el cambio si aplica.

Para determinar el cierre de la implementación, se debe haber concluido tanto con los cambios principales como los cambios emergentes solicitados y, además, se debe considerar un periodo de monitoreo sobre la implementación, en caso de contingencia por cualquier eventualidad, una vez entregado el proyecto y pasado el periodo de monitoreo, se puede realizar un acta de cierre y dar por terminado el proyecto.

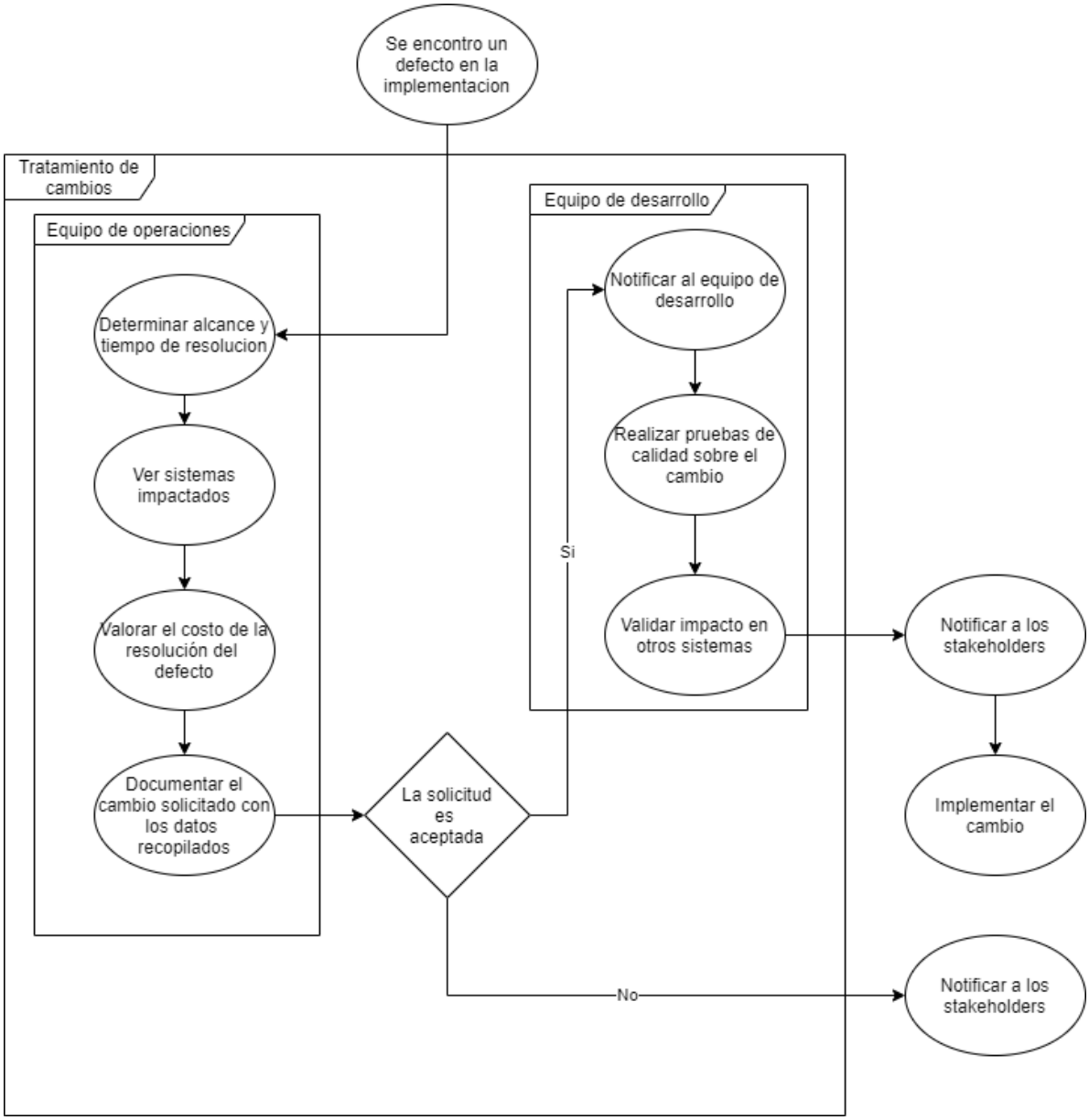


Figura 20 Figura 17. Tratamiento de Cambios.

Fuente: Elaboración Propia

4.2.6 Aceptación de entrega

Luego de haber completado la fase de pruebas y haber confirmado el trabajo realizado; se procede a realizar la presentación final del proyecto a los interesados, para que vean la demostración del producto finalizado. Luego de esto se procede a actualizar los documentos del proyecto, se redacta el informe final y se actualizan los activos de la empresa con el producto final.

Tabla 14. Plantilla aceptación de entregables

Proyecto: Nombre del proyecto.			
Usuario: Nombre de la unidad que recibe.			
Número de Entregable: 1			
Aceptación parcial del proyecto		Aceptación final del proyecto	x
Con el presente documento se certifica la finalización y aceptación de los siguientes entregables:			
Listado de funcionalidades entregadas: 1.			

	Director del proyecto	Dueño del producto
Firma		
Nombre		
Fecha		

Fuente: Elaboración propia

Las pruebas durante la implementación son de suma importancia con los usuarios que harán uso del sistema desarrollado. En este punto se debe validar que los requerimientos se cumplan en su totalidad al 100%.

Según los estándares establecidos por el ISTQB, las pruebas de aceptación de software son diseñadas a partir de:

a) Requerimientos del usuario.
b) Requerimientos de sistema.
c) Casos de uso.
d) Procesos de negocio.
e) Reportes de análisis de riesgo.

Tabla 15. Casos de pruebas de entregables.

No	Caso de prueba por funcionalidad	Estado	Comentarios
1		Pendiente/ En Ejecución/ Exitoso/ Fallido/	
2			
3			
4			
5			

Fuente: Elaboración propia

4.3 Desarrollo de prototipo

Base de datos

Con el fin de realizar el desarrollo del prototipo, se utilizó una base de datos libre de costos, la cual se creó para los propósitos académicos del presente trabajo.

Azure SQL Database es un servicio de base de datos relacional, inteligente y escalable creado para la nube. SQL Database es un motor de base de datos de plataforma como servicio (PaaS) totalmente administrada que se encarga de la mayoría de las funciones de administración de bases de datos, como actualizar, aplicar revisiones, hacer copias de seguridad y supervisar sin la intervención del usuario.

Con una cuenta gratuita de Azure, se puede probar Azure SQL Database sin costo durante 12 meses, con el límite mensual siguiente:

- 1 base de datos con 10 unidades de transacción de base de datos y almacenamiento de 250 GB

Para probar Azure SQL Database gratis, se necesita una cuenta gratuita de Azure (Azure product documentation, 2022).

Acorde a la referencia de Azure (The Azure SQL Workshop, 2022), dentro de la cobertura de la plataforma Azure SQL, existen muchas opciones de implementación y elecciones que se deben realizar para satisfacer las necesidades de desarrollo; estas opciones proveen la flexibilidad de obtener y pagar exactamente lo que necesita. Las opciones de implementación pueden incluir SQL Server en máquinas virtuales, Azure SQL Managed Instance, Azure SQL Database, grupos de Azure SQL Managed Instance y grupos de bases de datos elásticas de Azure SQL Database.

A continuación, se presenta una breve referencia de las tecnologías relacionadas a la base de datos Azure:

a) Máquinas virtuales Azure SQL. Permite utilizar versiones completas de SQL Server en la nube sin tener que administrar ningún hardware local
b) Base de datos Azure SQL. Servicio de base de datos SQL administrado (incluye bases de datos individuales y grupos elásticos)
c) Instancia administrada de Azure SQL. Servicio de SQL Server administrado (incluye instancia única y grupos de instancias)
d) Estudio administración de SQL Server (SSMS). Herramienta de gestión y consulta de la interfaz gráfica de usuario.
e) Azure Data Studio (ADS). Interfaz gráfica de usuario para ejecutar consultas T-SQL, cuadernos y administrar servicios de datos de Azure.

Integración de datos

Para la integración de los datos de la empresa en la base de datos intermedia, se diseñó un programa en C# consola basados en que la empresa utiliza productos de Microsoft. El programa lee la información relacionada a los empleados, su ubicación y los eventos que han registrado de las bases de la empresa, ordena la información y luego la inserta en la nueva base de datos. Este proceso tiene la finalidad de reducir el impacto de la carga de datos en la interfaz de usuario en los sistemas de la empresa ya que la constante lectura puede afectar el rendimiento de las bases de datos de la empresa.

Interfaz de usuario

En el desarrollo de la interfaz de usuario, se realizó a través de la herramienta Power BI, la cual permite conectarse a la base de datos intermedia que se encuentra en Azure SQL de forma integral sin mayor inconveniente o configuración para lograr dicha comunicación. Cabe mencionar que para este desarrollo, se utilizó Power BI Desktop que se puede descargar, usar y publicar de forma gratuita. Sobre el tipo de licencias se detalla en la tabla 16.

Tabla 16. Tipos de licencias.

Tipo de licencia	Funcionalidades cuando el área de trabajo está en capacidad compartida	Funcionalidades adicionales cuando el área de trabajo está en una capacidad Premium
Power BI (gratis)	Acceso al contenido de Mi área de trabajo	Consumo de contenido compartido con ellos
Power BI Pro	Publicación de contenido en otras áreas de trabajo, uso compartido paneles, suscripción a paneles e informes y uso compartido de contenido con usuarios que tienen una licencia Pro	Distribución de contenido a usuarios con licencias gratuitas
Power BI Premium por usuario	Publicación de contenido en otras áreas de trabajo, uso compartido de paneles, suscripción a paneles e informes y uso compartido de contenido con usuarios que tienen una licencia Premium por usuario	Distribución de contenido a usuarios con licencias Gratis y Pro

Fuente: Microsoft Documentatio (2022)

También, se detallan a continuación los tipos de planes y beneficios que otorga cada licencia.

Tabla 17. Características de cada plan de Power BI.

Características	Power BI Pro	Power BI Premium Por usuario	Power BI Premium Por capacidad
Acceso de aplicaciones móviles.	**x**	**x**	**x**
Publicación de informes para compartir y colaborar.	**x**	**x**	
Informes paginados		**x**	**x**
Consumo de contenido sin licencias por usuario.			**x**
Informes locales con Power BI Report Server			**x**
Límite de tamaño del modelo	**1 GB**	**100 GB**	**400 GB**
Frecuencia de actualización	**8/día**	**48/día**	**48/día**
Conexión con más de 100 orígenes de datos	**x**	**x**	**x**
Cree informes y visualizaciones con Power BI Desktop	**x**	**x**	**x**
Integración de API y controles	**x**	**x**	**x**
Seguridad y cifrado de datos	**x**	**x**	**x**
Métricas de creación, consumo y publicación de contenido	**x**	**x**	**x**
Almacenamiento máximo	**10 GB/usuario**	**100 TB**	**100 TB**

Fuente: Microsoft Power BI (2022)

A continuación, se presenta el costo de las licencias según cada plan que ofrece Microsoft:

Tabla 18. precios de licencias.

Power BI Pro Por usuario	Power BI Premium Por usuario	Power BI Premium Por capacidad
$ 9.99 por usuario/mes	**$ 20.00 por usuario/mes**	**$4,995.00 por capacidad/mes**
Licencia para usuarios individuales con análisis modernos de autoservicio para visualizar datos en paneles e informes interactivos y compartir conocimientos con toda la organización.	Licencia para usuarios individuales que les permita utilizar IA avanzada para preparar datos con funciones de autoservicio para macrodatos y simplificar la administración de datos y el acceso a ellos sea cual sea el tamaño de la empresa.	Licencia para usuarios individuales que les permita utilizar IA avanzada para preparar datos con funciones de autoservicio para macrodatos y simplificar la administración de datos y el acceso a ellos sea cual sea el tamaño de la empresa. Y todo sin necesidad de adquirir licencias de usuario para consumidores de contenido

Fuente: Microsoft Power BI (2022)

Luego de hacer la conexión con la base de datos, Power BI presenta la opción de mapear una o todas las tablas, según se requiera; para este desarrollo se seleccionó las tablas relacionadas a los eventos y se generó con la configuración “Import”. En Power BI se encuentra tres tipos de ambientes, el primero donde se mapea la información de la base y muestra las relaciones si hubiese, la segunda donde se realiza la transformación de los datos, donde se pueden crear columnas adicionales a la tabla o columnas de tipo calculadas que no se reflejan visualmente en la tabla, pero son muy útiles para la presentación de métricas o gráficas.

En la tercera sección, es la parte visual donde se va agregando cada gráfica, tabla, mapas, entre otros; y estos se van configurando con cada campo o columna que se requiera del modelo mapeado.

4.4 Implementación de prototipo

Base de datos

Para la implementación del prototipo, al igual que para el desarrollo de este, se procedió a utilizar una base de datos en Azure SQL Database, la cual se utilizó como repositorio de datos con el fin de realizar la carga de estos, y que de esta manera estuvieran disponibles en la nube para la visualización de la interfaz de usuario. En los anexos se presenta la guía de configuración de la base de datos y demás componentes del prototipo.

Citando la referencia de Azure (The Azure SQL Workshop, 2022), Azure SQL Database es una opción de implementación de PaaS de Azure SQL que abstrae tanto el sistema operativo como la instancia de SQL Server de los usuarios. Azure SQL Database tiene el SLA de disponibilidad más alto de la industria, junto con otras capacidades inteligentes relacionadas con la supervisión y el rendimiento, debido en parte al hecho de que Microsoft administra la instancia. Esta opción de implementación permite a los ingenieros simplemente 'obtener una base de datos' y comenzar a desarrollar aplicaciones. Azure SQL Database (Azure SQL DB) también es la única opción de implementación que actualmente admite escenarios relacionados con la necesidad de almacenamiento de base de datos ilimitado (hiperescala) y escalado automático para cargas de trabajo impredecibles (sin servidor).

La integración de Azure SQL Database con Power Platform permite facilitar la conexión a datos SQL con Power BI (por ejemplo, con DirectQuery). También facilita la creación de aplicaciones con poco o ningún código, aprovechando Power Apps e implementándolas para su

uso en cualquier dispositivo. Además, Power Automate contribuye a aprovechar Azure al permitir crear flujos de trabajo automatizados en minutos, en aplicaciones y servicios, como alertas, Dynamics 365, OneDrive, Office 365, SharePoint y más.

Power Platform es solo un ejemplo de cómo se puede ir más allá de Azure SQL. También hay integraciones y conectores en Azure Search (esto incluye Azure Cognitive Search) y Azure Stream Analytics. Incluso para los servicios en los que no hay integraciones específicas, todos los servicios de Azure se crean en la red troncal de Azure (principalmente Azure Service Fabric) y se pueden configurar para comunicarse entre sí de manera eficiente.

Los límites para Azure SQL Managed Instance y Azure SQL Database dependen de la elección de modelo de compra, nivel de servicio y número de núcleos virtuales (o DTU solo en Azure SQL Database). Dentro de una base de datos SQL de uso general de Azure, la elección de computación aprovisionada o sin servidor también afectará estos límites. Se debe revisar lo que está incluido en lo que se planea implementar, antes de realizar dicha actividad, para asegurarse de que se está comenzando con lo que realmente se necesita.

También es importante saber que los recursos de Azure SQL tienen límites generales de recursos por suscripción y por región. Si se necesita aumentar los límites, es posible solicitar un aumento de cuota en el portal de Azure.

Integración de datos

Para poder implementar el software de integración de datos, primero se debe configurar a nivel de código para colocar las conexiones a bases de datos y las consultas respectivas de las bases de datos que proveerán la información, la cual deberá ser almacenada en diccionarios, los cuales servirán para construir las consultas y las inserciones de datos en la base de datos intermedia. La cual es la fuente de alimentación de la interfaz de usuario.

Se debe considerar que el programa en C# tiene que ser compilado, ya que es un programa en consola y utiliza alguna herramienta integrada o de terceros en Windows para poder ejecutar este programa, ya sea bajo demanda o programado cada cierto tiempo.

Interfaz de usuario

Para la implementación del prototipo, al utilizar la licencia gratis, se presenta en el área de trabajo que designa Power BI, por lo tanto, luego de desarrollado y configurado el proyecto, se publica y se puede interactuar de forma web en dicha área de trabajo.

Capitulo 5. Presentación de resultados del prototipo

Sistema de Información Geográfico.

Se elaboró el dashboard en el que se presenta información sobre eventos o incidencias que se presentan en la empresa, se puede observar en la figura 21, donde se incluye filtros para visualizar información por año y mes, además de integrar la función del mapa que proporciona información de la localización donde ocurrieron dichos eventos.

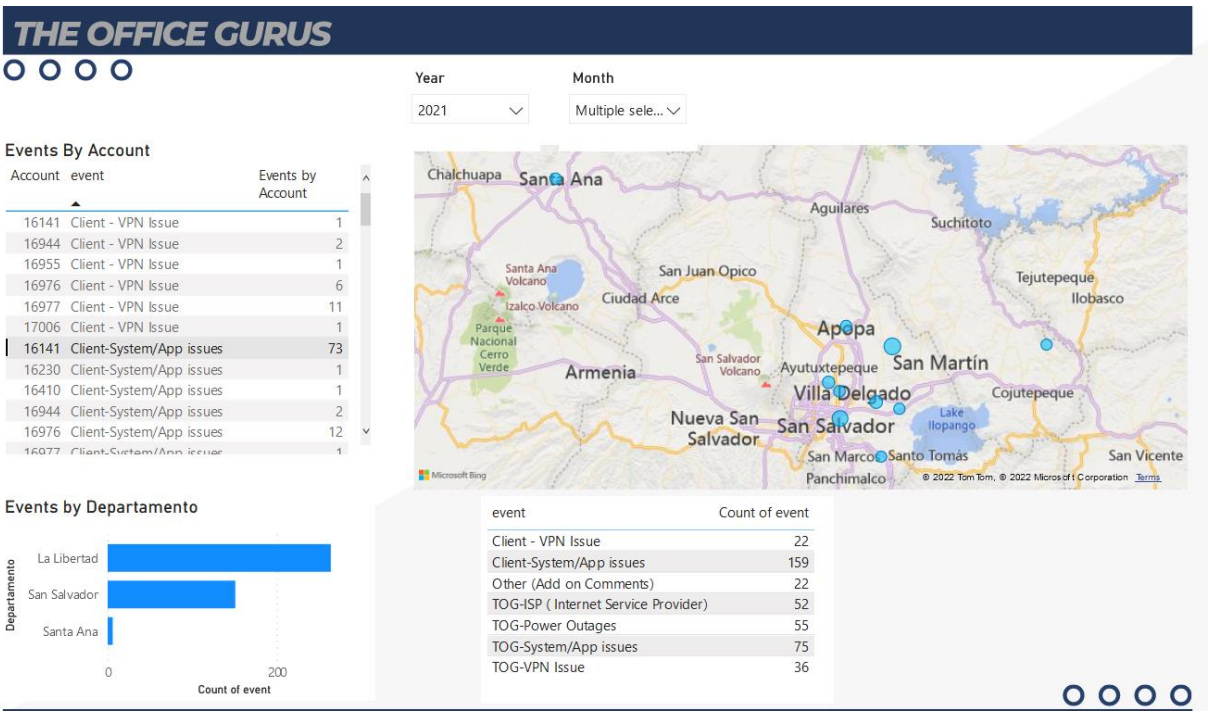


Figura 21. Dashboard prototipo.

Fuente: Elaboración propia.

Una de las características que brinda este tipo de reporte, es la forma dinámica y focalizada que muestra la información, en la figura 22, se puede observar un ejemplo de seleccionar un punto en el mapa y como la información que se visualiza, cambia según ese criterio para cada tabla y gráfica que está configurada en el dashboard.

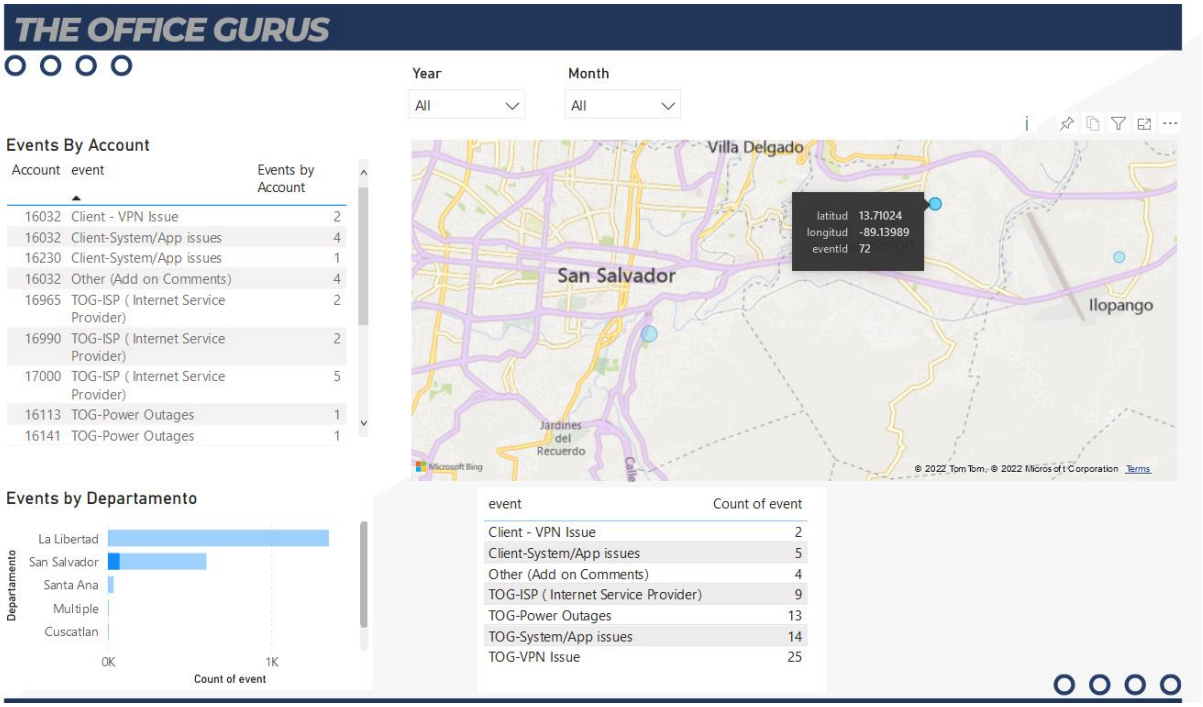


Figura 22. Información por locación.

Fuente: Elaboración propia.

Capitulo 6. Discusión

1. Evaluación del cumplimiento de la hipótesis

Tomando en cuenta la hipótesis: “La implementación de un prototipo de un módulo del sistema de información gerencial de localización brindará a la empresa la experiencia sobre el uso de la información de georreferenciación obtenida mediante inteligencia de negocios e integración de sistemas, con el fin de tomar decisiones relacionadas a los agentes y los eventos relacionados al teletrabajo”, se puede considerar como cumplida, esto debido a los comentarios provistos por los directivos de la empresa respecto a lo mostrado en el prototipo. Se mostraron motivados por cómo se desplegaba la información en la interfaz de usuario y también por la idea de usar una base de datos intermedia para poder reducir el impacto en producción, observando mucho potencial para futuras integraciones y planes a futuro para otro tipo de proyectos.

2. Impacto en las operaciones de la empresa

Luego de presentar el prototipo a los gerentes de la empresa, se procedió a realizar una dinámica de preguntas y respuestas, donde ellos brindaron sus valoraciones de más alto nivel con respecto a la información que se presenta en la interfaz de usuario.

1. ¿CÓMO CONSIDERA QUE ESTA HERRAMIENTA VA A MEJORAR LA EFICIENCIA EN LAS OPERACIONES DE LA EMPRESA?	
Se mencionaron muchos aspectos en los cuales esta herramienta va a mejorar las operaciones de la empresa, entre las cuales se destacan los siguientes puntos:	
	Auditoria. Se refiere a una confirmación de sistema acerca de donde está trabajando el empleado físicamente en tiempo real, acorde a su contrato de teletrabajo.
	Ubicación en el mapa geográfico donde están físicamente los dispositivos asignados a cada persona.
	Ubicar en el mapa a un empleado, con el fin de enviarle incentivos, por ejemplo, comida a domicilio, regalos, entre otros.
	Facilitar las tareas de envío de equipo nuevo o reemplazo de equipo averiado al domicilio del empleado.
	Ubicar geográficamente donde está la población de empleados, para efectos de buscar mejores ubicaciones de las oficinas de la empresa, ya que, por ejemplo, hubo deserción de talento humano cuando en una ocasión se cambió la ubicación física de las oficinas de la empresa, a un lugar donde no le convenia a una gran mayoría de personas.
	Promociones especificas a los empleados en un radio en particular, en donde haya cobertura en las zonas donde residen los empleados.
	Administrativamente, con el fin de mejorar las condiciones de trabajo de los empleados.
	Financieramente, para ahorrar costos por envío de incentivos a direcciones erróneas de domicilio.

	Debido a la Ley de Servicios Internacionales, regulado por el Ministerio de Economía, a la que se sujetan los Call Center, en el apartado de teletrabajo, al menos dos veces al año se debe conocer donde está ubicado geográficamente el equipo de cómputo asignado a cada empleado.
	Buscar las mejores ubicaciones de las oficinas de la empresa, acorde a la cercanía del domicilio de la mayoría de los empleados, cuando se presente la oportunidad de regresar a las oficinas operativas, ubicando de esta manera a los lugares más cercanos al domicilio del empleado.

2. ¿CONSIDERA QUE ESTÁ HERRAMIENTA LE BRINDARÁ A LA EMPRESA UNA VENTAJA COMPETITIVA EN EL MERCADO? ¿EN QUÉ MANERA?
Definitivamente que sí brindara una ventaja competitiva a la empresa, debido a que, a mayor conocimiento de datos, mayor es el fortalecimiento en el proceso de toma de decisiones, y sobre todo si estas decisiones se toman en lapsos cortos de tiempo, ya que de esta manera se tendrá ventaja sobre la competencia, al reaccionar, y si es posible anticiparse oportunamente ante eventos inesperados.

3. ¿CONSIDERA A ESTA HERRAMIENTA DE VALOR ESTRATÉGICO PARA LA EMPRESA?
Si, esta herramienta es de mucho valor estratégico para la empresa, ya que el conocimiento es poder, debido a que las ventajas obtenidas a partir del procesamiento y análisis de la información son al final estrategias que ayudan a que los objetivos del negocio se cumplan. Las necesidades del teletrabajo derivadas de la pandemia del covid-19, han obligado a cambiar los esquemas de trabajo, y a innovar la manera en que se hacen las cosas. Algunas ventajas se pueden listar de la siguiente manera:
Obtención de información en el momento oportuno.
Toma de decisiones más rápida, evitando contactar a otras áreas para obtener datos consolidados, ya que pedir consolidados toma más esfuerzo y tiempo, el cual mejor se puede utilizar en actividades de mejoramiento de atención al cliente.
La estrategia para manejar las operaciones del negocio se basa sobre los consolidados de datos y la generación de información.

3. Evaluación del fortalecimiento del proceso de toma de decisiones

Principalmente, el objetivo es lograr la creación oportuna de estrategias gerenciales, con el fin de generar la sostenibilidad del negocio a través de los ingresos generados por las cuentas de los clientes de la empresa.

1. ¿CÓMO CONSIDERA QUE ESTA HERRAMIENTA VA A AYUDAR EN EL FORTALECIMIENTO DEL PROCESO DE TOMA DE DECISIONES?	
	Esta es una solución muy deseada por muchos gerentes, ya que permite mejorar las estrategias del negocio y las operaciones, y contribuyen para aprovechar la creación de sistemas de integración de datos, que colecten, procesen y generen información de alto valor para el proceso de toma de decisiones.
	La publicación en sitios, con perfiles de seguridad específicos, permitiría ahorrar tiempo en la creación de estrategias. La sincronización eficiente de los sistemas periféricos con esta herramienta de reportería a nivel geográfico es de vital importancia para el éxito del proceso de toma de decisiones.

2. ¿A QUE CARGOS Y EN QUÉ TIPO DE DECISIONES SE TENDRÁ BENEFICIOS DIRECTOS?	
El tipo de decisiones depende del tipo de información que cada uno necesita para elaborar sus estrategias, sin embargo, los datos deben ser procesados en tiempo real para que la información que se genere sea de alto valor para el tomador de decisiones. Los principales cargos en los cuales se tendrán beneficios inmediatos son:	
	Presidencia.
	Director de recursos humanos.
	Altas gerencias.
	Mandos medios.

3. ¿CUÁLES ÁREAS DE LA EMPRESA SE VERÁN BENEFICIADAS Y EN QUÉ MANERA?	
Existen muchos campos de acción para explotar los beneficios de esta herramienta, entre los que se consideran en términos generales son:	
	Creación de planes estratégicos para prevenir paro de operaciones por eventos inesperados, por ejemplo, cortes de energía, internet, desastres naturales, entre muchos otros imprevistos.
	Brindar un soporte oportuno a los empleados que han sufrido algún tipo de evento inesperado.
	Ofrecer opciones de rápida movilidad en los casos que amerite.
	Brindar información en tiempo real para la ejecución de proyectos de bonificaciones por zona, basado en información geográfica.
	El mayor uso de herramientas que capturen coordenadas geográficas de los dispositivos asignados a los empleados.

Capitulo 7. Conclusiones

Con el desarrollo e implementación de este sistema, el principal objetivo que se busca es la eficiencia en las operaciones del negocio, a través del cumplimiento de indicadores de rendimiento, logrando de esta manera una ventaja competitiva en el mercado. El mantener la facturación de cuentas y aumentar los ingresos, es clave para el éxito de este tipo de negocios, que se basan en la excelencia operativa y además en la confianza de los clientes.

La creación oportuna de estrategias de mejora operativa es de gran ayuda para el negocio, utilizando como aliado clave el uso de las tecnologías de las comunicaciones y la información, alcanzado de esta forma los máximos beneficios posibles con el teletrabajo, y eventualmente el regreso a la oficina, o bien con esquemas de trabajo híbrido.

El tener disponible una herramienta de reportes que permite la ubicación por georreferenciación de los agentes, fortalece la detección de la veracidad de los reportes de incidentes o saber qué sector del mapa puede cubrir la operación en caso de incidentes.

La empresa, al llevar a cabo la implementación de este sistema, obtendrá una contribución de manera oportuna y positiva al establecimiento de las bases para un crecimiento rentable y sostenible del negocio, manteniéndose en el nivel estratégico el proceso de toma de decisiones, con el cual se puede garantizar el éxito del negocio y el beneficio de los empleados que laboran en la organización.

Al haber creado el prototipo acorde a las herramientas utilizadas en la empresa, se podrá acoplar a sus proyectos y necesidades sin ningún problema de compatibilidad entre distintos tipos de lenguajes de programación o de bases de datos, además de la capacidad de inserción que posee Power BI para sus reportes en otras aplicaciones

Capitulo 8. Recomendaciones

Se recomienda a la empresa realizar un análisis exhaustivo sobre las fuentes y los datos que se necesitan consolidar, con el fin de generar la información oportuna en tiempo real, que les permita tomar decisiones de forma ágil y confiable, que ayuden al negocio a ser más competitivo en el mercado de los centros de atención telefónica.

La empresa debería considerar la ejecución oportuna de los planes de desarrollo e implementación propuestos en este informe de investigación, ya que bajo la óptica de mejorar los procesos de atención al cliente, dichos planes promueven el análisis basado en la recolección, procesamiento, y presentación de información fidedigna, dichos datos permiten tanto a nivel de tiempo real como de históricos, el modelamiento de la realidad bajo diferentes vistas, las cuales contribuirán a mejorar el proceso toma de decisiones de los niveles ejecutivos, realizar la creación de estrategias sobre la base de la inteligencia de negocios, con una potente herramienta como lo es Power BI.

Se recomienda a la empresa tomar como base los detalles compartidos relacionados al prototipo mostrado en la demostración, sin embargo, no se debe tomar todo lo compartido al pie de la letra, se deben adaptar los diferentes elementos compartidos en el prototipo acorde a los lineamientos de la empresa, esto brindará un impulso a la hora de la creación de la aplicación principal y será una base bastante útil para el análisis de la información.

Una de las recomendaciones proporcionadas durante la demostración del prototipo, fue comprar una licencia de Power BI, la cual proveerá de acceso a múltiples herramientas de análisis de datos, así como a la herramienta de mapas integrada de ArcGIS que tiene Power BI, sin ninguna restricción de contenido.

Como parte del prototipo se creó una base de datos intermedia, se recomienda a la empresa seguir esta estrategia como un repositorio de la información que se necesita procesar en Power BI, dicha base de datos servirá para alimentar a los reportes y reducir el impacto en el desempeño de los ambientes productivos involucrados en la integración.

Capitulo 9. Referencias

Atencio Bravo, E. (2020). ¿El para qué y el cómo de la investigación? Dirección de Educación a Distancia, UDB Virtual

Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, P. (2014). Metodología de la investigación. McGraw Hill Interamericana.

García, N., Linares, J. (2020). Manual para la implementación de CMMI a nivel de capacidad 2 con SCRUM en el área de desarrollo de software. (Tesis de Maestría no publicada). Universidad Don Bosco, San Salvador, El Salvador, C.A.

Hiromoto, H. (2017, December 31). Product Vision Board. Scrumorganico. https://medium.com/scrumorganico/product-vision-board-81188795e712

CMMI Institute (2018). Guía de Adopción y Transición CMMI 2.0 https://docplayer.es/166278179-Guia-de-adopcion-y-transicion-de-cmmi.html

Maruthi Rohit Ayyagari, Issa Atoum (2019). CMMI-DEV Implementation Simplified https://thesai.org/Downloads/Volume10No4/Paper_55-CMMI_DEV_Implementation_Simplified.pdf

Olaya (2014). Sistemas de Información Geográfica, pag 7, disponible en: https://www.icog.es/TyT/files/Libro_SIG.pdf

Garcia, Fernandez, Galvez, Sanches y Grimaldi (2016), La geolocalización como herramienta para conocer a los consumidores: el caso de los centros de fitness. Pag. 265. Disponible en: https://www.researchgate.net/publication/319654852_La_Geolocalizacion_como_Herramienta_para_Conocer_a_los_Consumidores_El_Caso_de_los_Centros_de_Fitness

Micheli (2012), El sector de call centers: Estructura y tendencias. Apuntes sobre la situación de México, Pág. 147. Disponible en: http://www.scielo.org.mx/pdf/fn/v24n47/v24n47a6.pdf

Azure product documentation (2022). Quickstart: Create an Azure SQL Database single database. Disponible en: https://docs.microsoft.com/en-us/azure/azure-sql/database/single-database-create-quickstart?tabs=azure-portal&view=azuresql

Azure product documentation (2022). Prueba de Azure SQL Database gratis con la cuenta gratuita de Azure. Disponible en: https://docs.microsoft.com/es-es/azure/azure-sql/database/free-sql-db-free-account-how-to-deploy

The Azure SQL Workshop (2022). A Microsoft Course from the SQL team. Disponible en: https://github.com/microsoft/sqlworkshops-azuresqlworkshop

Microsoft Documentation (2022). Licencias de Power BI para los usuarios de la organización—Power BI, en https://docs.microsoft.com/es-es/power-bi/enterprise/service-admin-licensing-organization

Microsoft Power BI Comparación de productos y precios | Microsoft Power BI. (n.d.). en https://powerbi.microsoft.com/es-es/pricing/

Integración de datos: Concepto e importancia en la empresa actual, Powerdata. En https://www.powerdata.es/integracion-de-datos

Whitepaper - Que es inteligencia de Negocios, Oracle. En https://www.oracle.com/ocom/groups/public/@otn/documents/webcontent/317529_esa.pdf

Cascon,Ruiz y Alberich (2017), Usos y aplicaciones de georreferenciación y geolocalización en gestión documental cartográfica y fotográfica antiguas. En https://revista.profesionaldelainformacion.com/index.php/EPI/article/download/63288/38574/192210

Capitulo 10. Anexos

Anexo 1: Guía técnica de configuración y despliegue del sistema

Base de datos

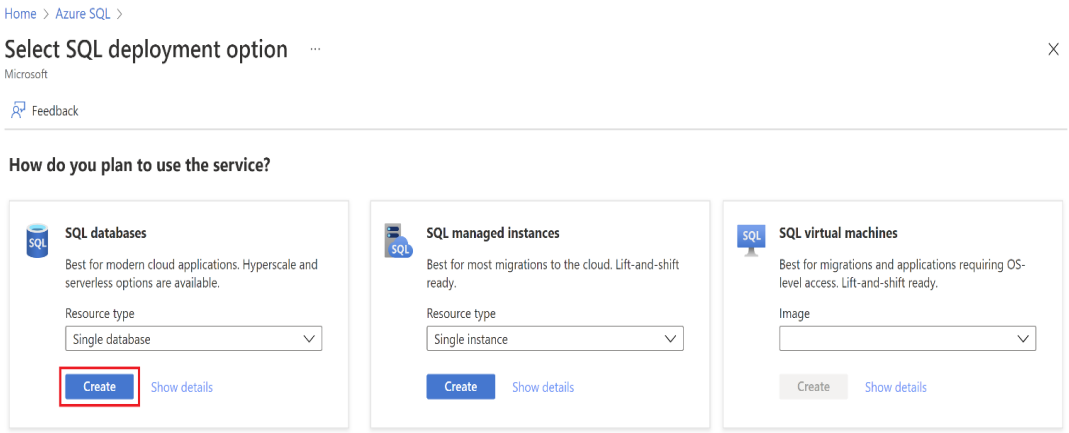
Acorde al Quickstart provisto por el sitio de Azure (Azure product documentation, 2022), se puede crear una base de datos única en Azure SQL Database mediante el Portal Azure, un script de PowerShell o un script de la CLI de Azure, es en ese momento, que ya se pueden realizar consultas a la base de datos mediante el editor de consultas en Azure Portal.

El tipo de recurso de base de datos único crea una base de datos en Azure SQL Database con su propio conjunto de recursos y se administra a través de un servidor. Con una sola base de datos, cada base de datos está aislada, utilizando un motor de base de datos dedicado. Cada uno tiene su propio nivel de servicio dentro del modelo de compra basado en DTU o modelo de compra basado en vCore y un tamaño de proceso que define los recursos asignados al motor de base de datos.

La base de datos única es un modelo de implementación para Azure SQL Database. El otro tipo es “elastic pools”.

Pasos para la configuración de la base de datos:

1. Ir a la página de opciones y seleccionar Implementación de SQL.
2. En Bases de datos SQL, deje Tipo de recurso establecido en Base de datos única y seleccione Crear.



3. En la pestaña Conceptos básicos del formulario Crear base de datos SQL, en Detalles del proyecto, seleccione la Suscripción de Azure deseada.

4. Para Grupo de recursos, seleccione Crear nuevo, ingrese myResourceGroup y seleccione Aceptar.

5. Para el nombre de la base de datos, ingrese mySampleDatabase.

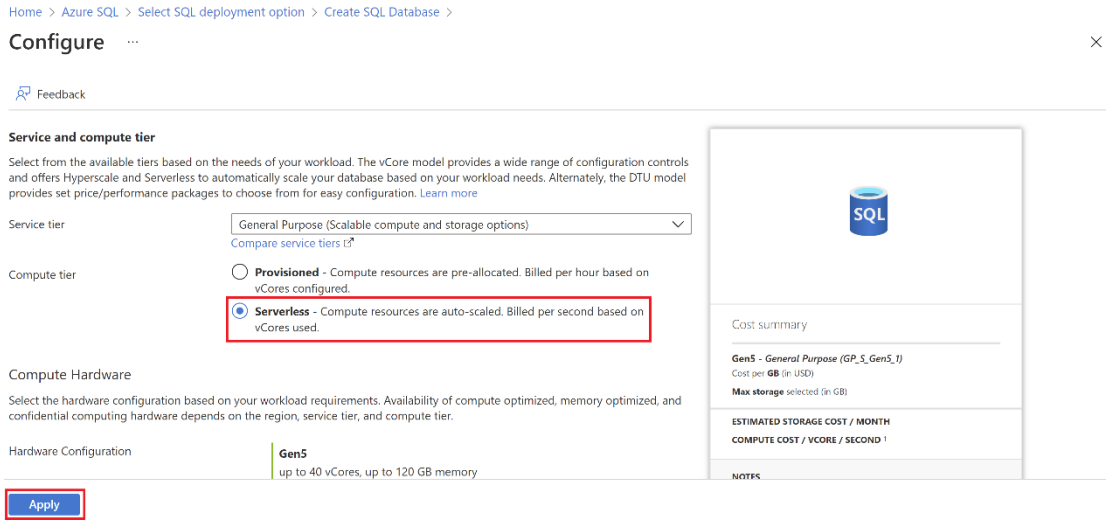
6. Para Servidor, seleccione Crear nuevo y complete el formulario Nuevo servidor con los siguientes valores:
    - Nombre del servidor: Ingrese mysqlserver y agregue algunos caracteres para que sea único. No se puede proporcionar un nombre de servidor exacto para usar porque los nombres de los servidores deben ser únicos globalmente para todos los servidores en Azure, no solo únicos dentro de una suscripción. Ingrese algo parecido a mysqlserver12345 y el portal le informará si está disponible o no.
    - Ubicación: Seleccione una ubicación de la lista desplegable.
    - Método de autenticación: Seleccione Usar autenticación SQL.
    - Inicio de sesión del administrador del servidor: Ingrese azureuser.
    - Contraseña: ingrese una contraseña que cumpla con los requisitos y vuelva a ingresarla en el campo Confirmar contraseña.

    Seleccione Aceptar.

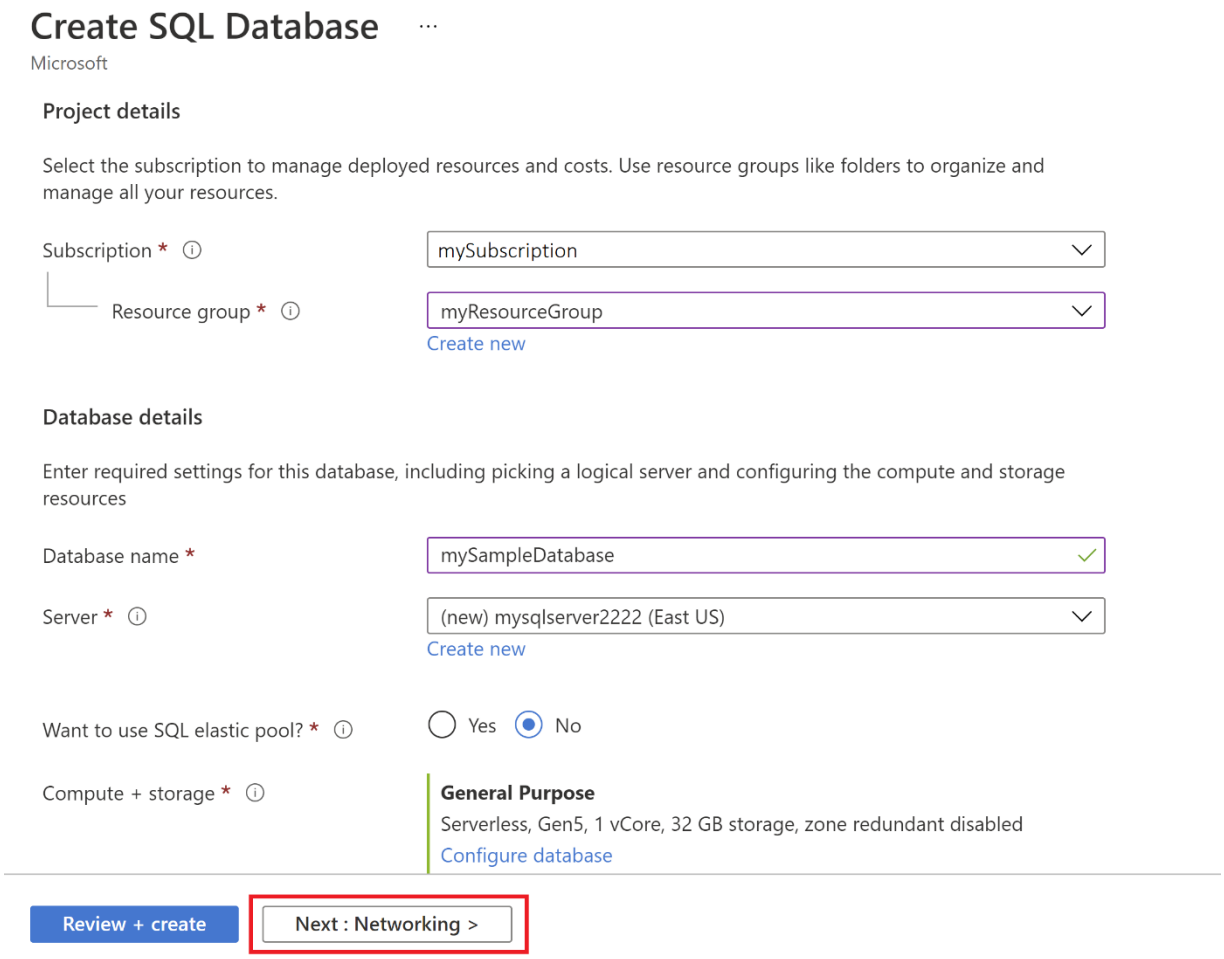
7. Deje Quiero usar el grupo elástico de SQL establecido en No.

8. En Cómputo + almacenamiento, seleccione Configurar base de datos.

9. Esta guía de inicio rápido utiliza una base de datos sin servidor, así que se puede dejar el Nivel de servicio establecido en Uso general (opciones de almacenamiento y computación escalables) y configure el Nivel de computación en Sin servidor. Seleccione Aplicar.


Home > Azure SQL > Select SQL deployment option > Create SQL Database >
Configure
Feedback
Service and compute tier
Select from the available tiers based on the needs of your workload. The vCore model provides a wide range of configuration controls and offers Hyperscale and Serverless to automatically scale your database based on your workload needs. Alternately, the DTU model provides set price/performance packages to choose from for easy configuration. Learn more
Service tier
General Purpose (Scalable compute and storage options)
Compare service tiers
Compute tier
Provisioned - Compute resources are pre-allocated. Billed per hour based on vCores configured.
Serverless - Compute resources are auto-scaled. Billed per second based on vCores used.
Compute Hardware
Select the hardware configuration based on your workload requirements. Availability of compute optimized, memory optimized, and confidential computing hardware depends on the region, service tier, and compute tier.
Hardware Configuration
Gen5
up to 40 vCores, up to 120 GB memory
Cost summary
Gen5 - General Purpose (GP_S_Gen5_1)
Cost per GB (in USD)
Max storage selected (in GB)
ESTIMATED STORAGE COST / MONTH
COMPUTE COST / VCORE / SECOND
NOTES
Apply

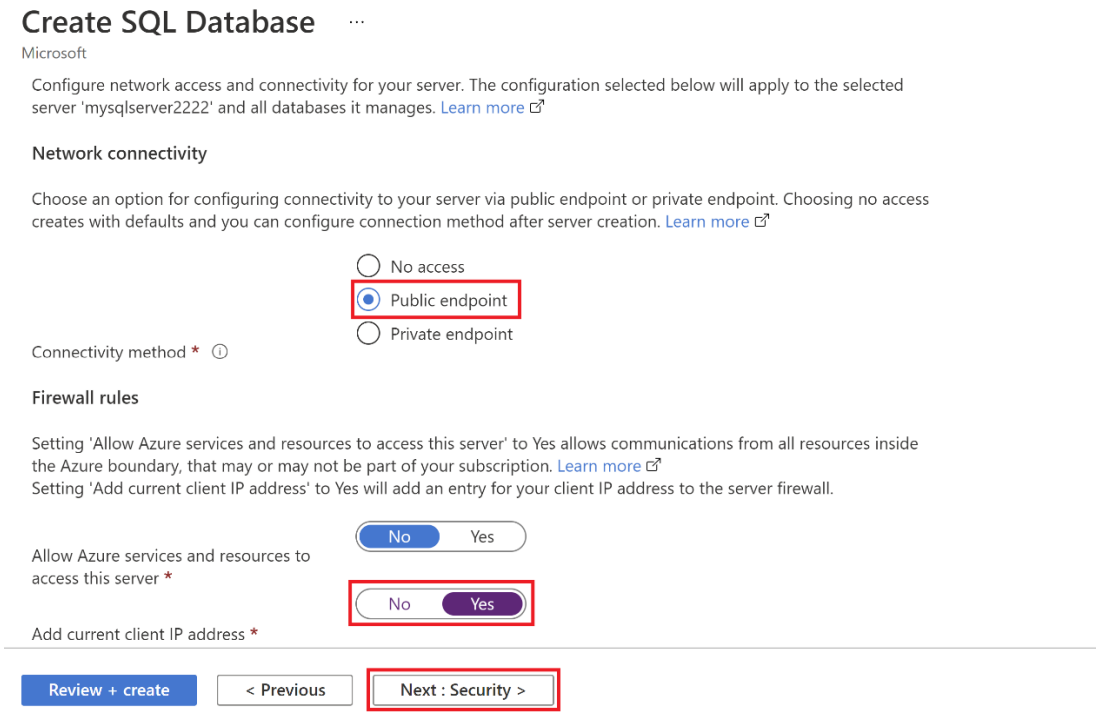
10. Seleccione Siguiente: Redes en la parte inferior de la página.



11. En la pestaña Redes, para Método de conectividad, seleccione Punto final público.

12. Para las reglas de Firewall, establezca Agregar la dirección IP del cliente actual en Sí. Deje Permitir que los servicios y recursos de Azure accedan a este servidor establecido en No.

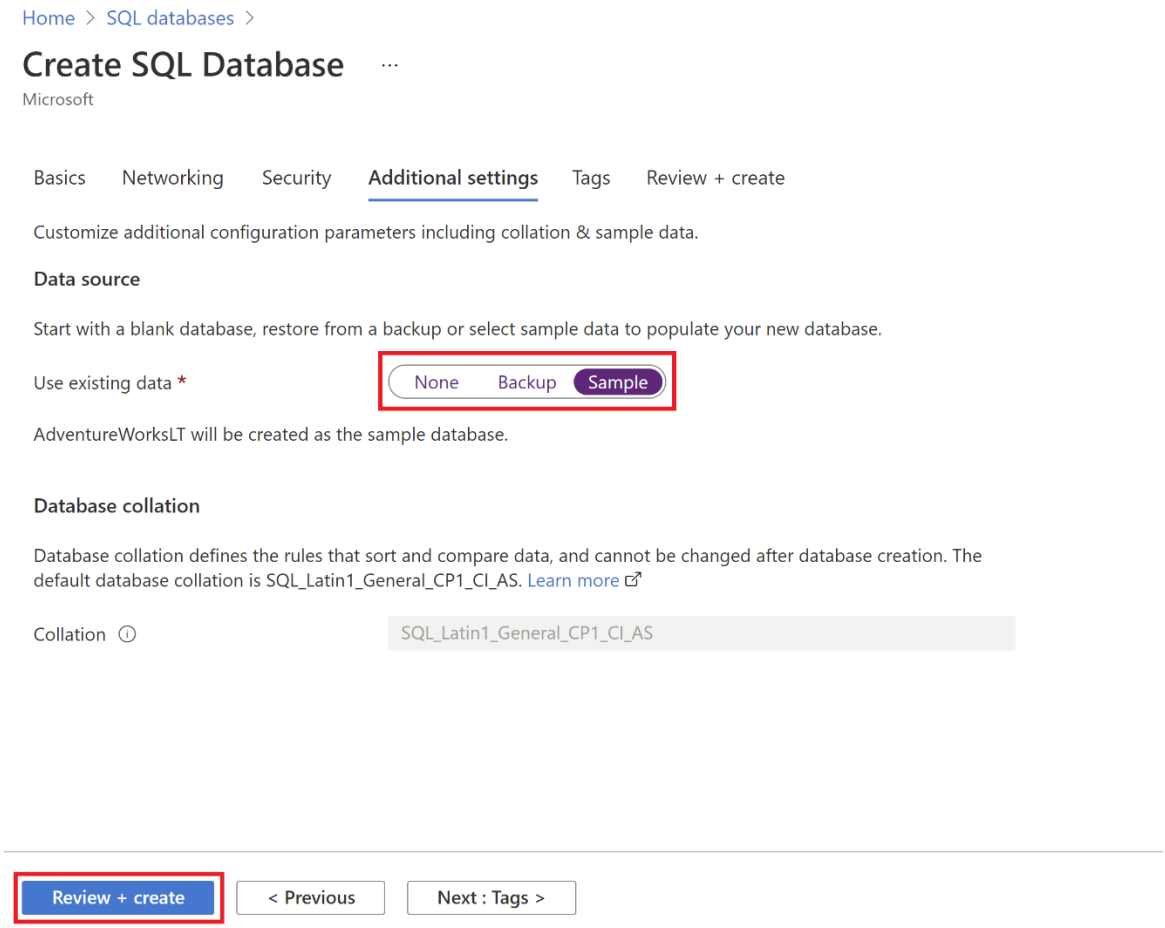
13. Seleccione Siguiente: Seguridad en la parte inferior de la página.



14. En la pestaña Seguridad, tiene la opción de habilitar Microsoft Defender para SQL. Seleccione Siguiente: Configuración adicional en la parte inferior de la página.

15. En la pestaña Configuración adicional, en la sección Fuente de datos, para Usar datos existentes, seleccione Muestra. Esto crea una base de datos de ejemplo AdventureWorksLT, por lo que hay algunas tablas y datos para consultar y experimentar, en lugar de una base de datos vacía.

16. Seleccione Revisar + crear en la parte inferior de la página:

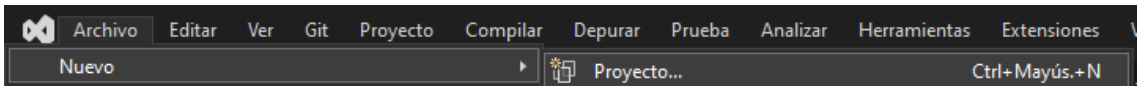


17. En la página Revisar y crear, después de revisar, seleccione Crear.

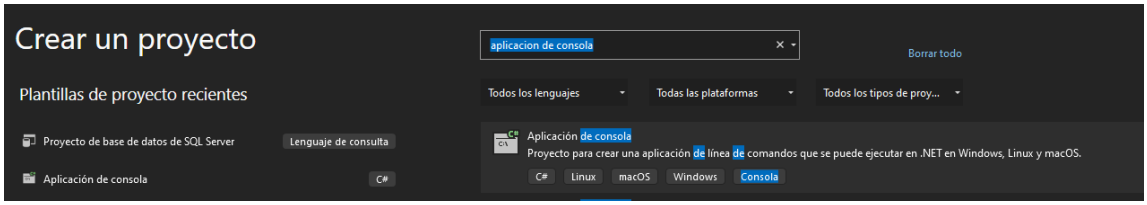
Integración de datos

Pasos para la creación de un proyecto de C# consola en Visual Studio 2022:

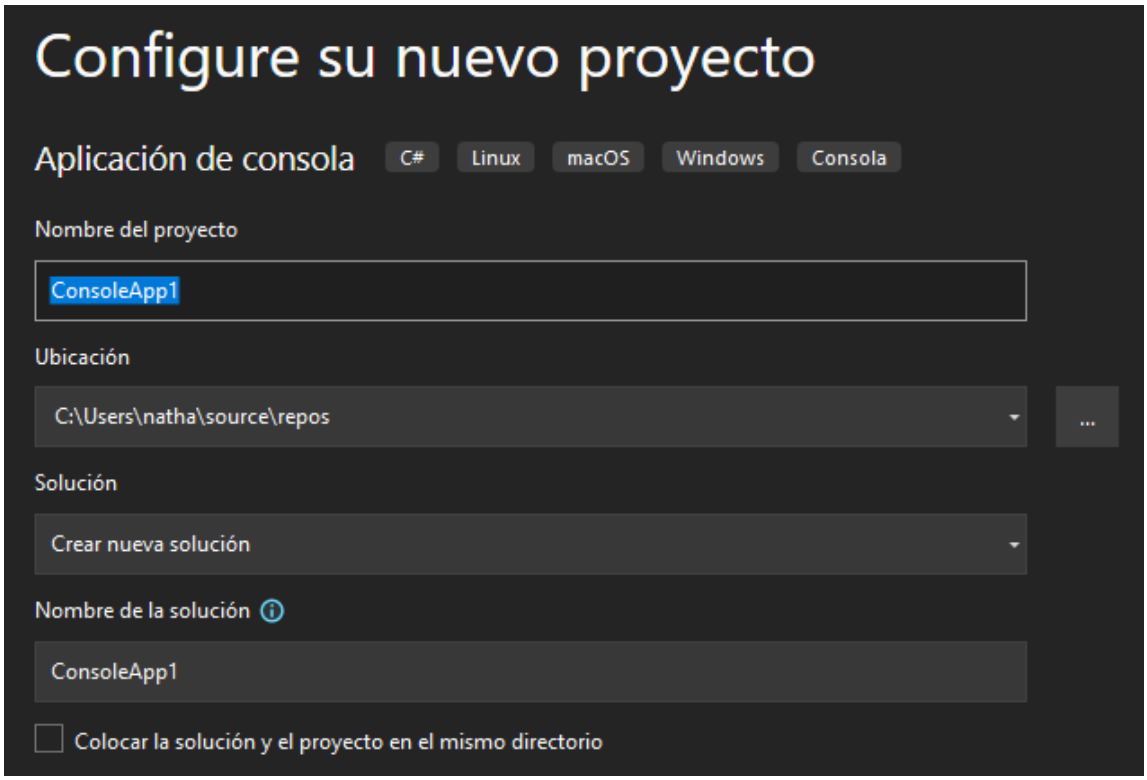
a. Ve a archivo y luego haz click en nuevo y elige la opción proyecto.



b. En la ventana de crear proyecto, escribi “aplicación de consola” y escoger la opción que sea de C#

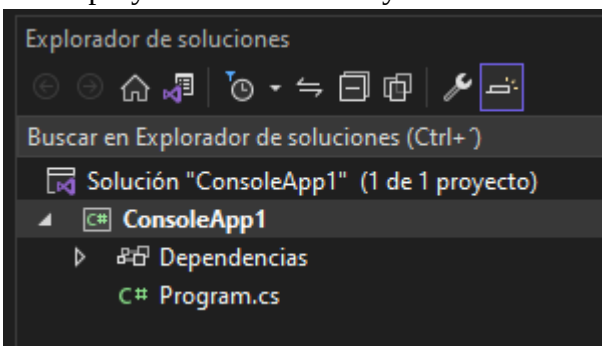


c. Colocar el nombre en el proyecto y hacer click en siguiente

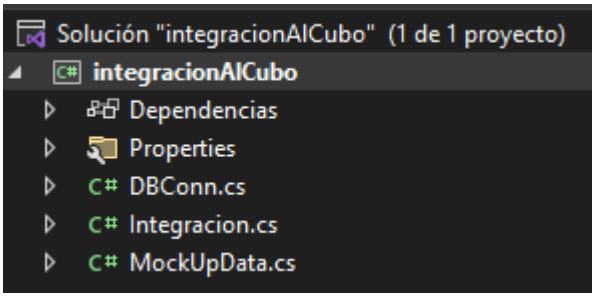


d. Elegir el framework y hacer click en crear

e. Esto creará un nuevo proyecto en consola cuyo estado inicial será el siguiente:



f. Para esta integración se crearon 3 archivos (este desarrollo es una sugerencia que puede variar acorde a las necesidades, creado con fines de prototipo y no representan el diseño ni la idea final): DBConn que contiene la información relacionada con conexión a bases de datos, integración que es la clase principal que ejecutara la aplicación de consola y MockUpData que se utilizara en caso se deseen hacer pruebas de carga de informacion.



g. Uno de los pasos más importantes a la hora de configurar este programa es tener en cuenta que la conexión se realiza por medio del siguiente código, el cual es bastante sencillo y soportado para una aplicación de consola de C#

```
var datasource = @"DESKTOP-PC\SQLEXPRESS";//your server
var database = "Students"; //your database name
var username = "sa"; //username of server to connect
var password = "password"; //password
```

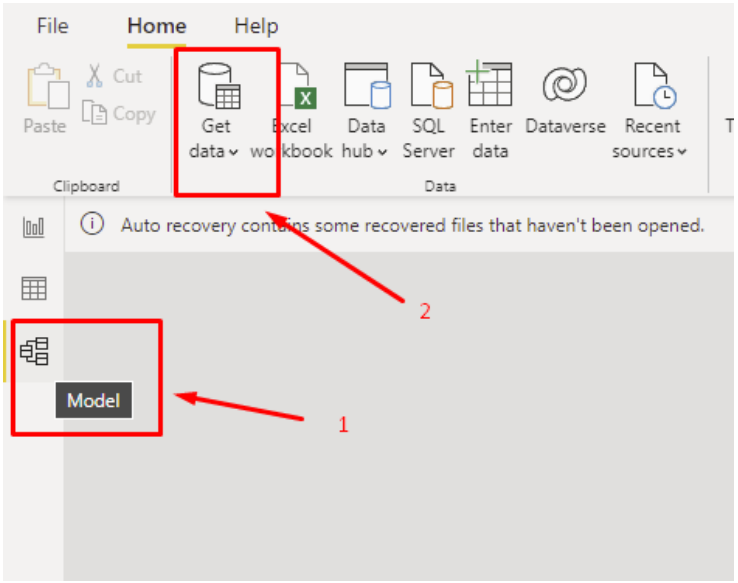
```
//your connection string
string connString = @"Data Source=" + datasource + ";Initial Catalog="
+ database + ";Persist Security Info=True;User ID=" + username + ";Password=" +
password;
//create instanace of database connection
SqlConnection conn = new SqlConnection(connString);
```

h. Teniendo configurado tanto las bases de datos que proveen la información como la base de datos para el cubo, solo queda hacer pruebas e insertar la información.

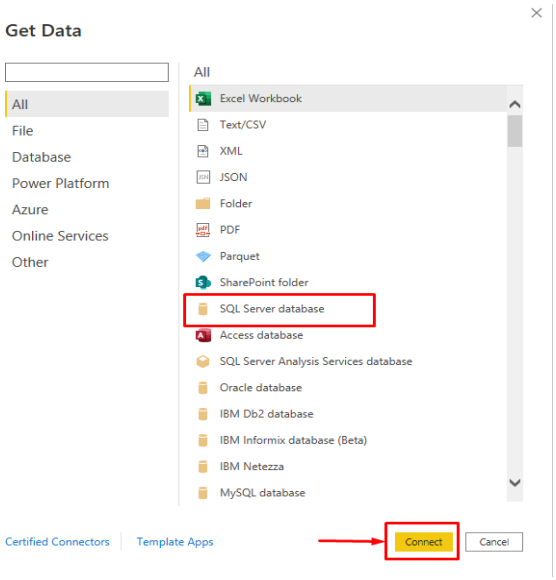
Interfaz de usuario

Pasos para la configuración del Power BI:

1. En el ambiente “Model”, en la pestaña del menú “Home” seleccionar la opción “Get data”.

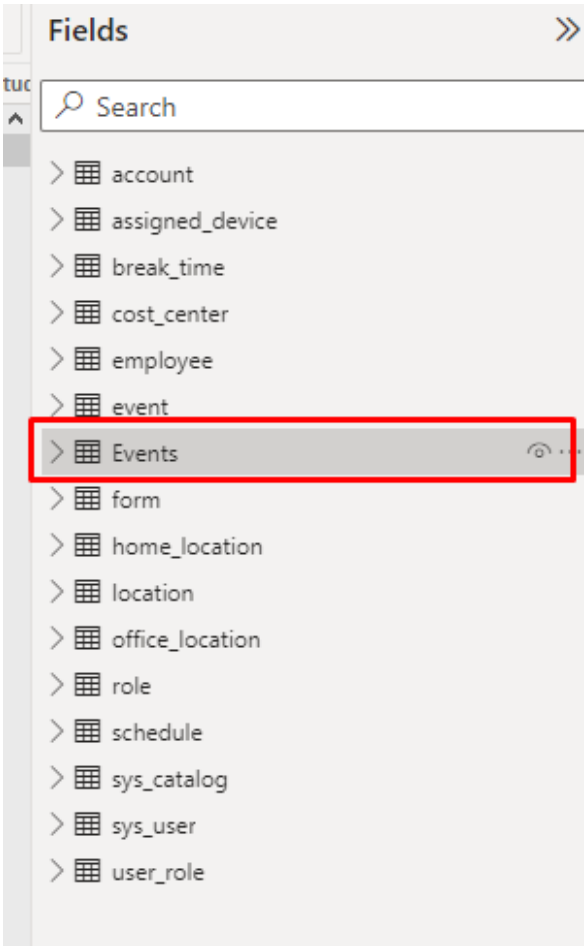


2. Se muestra un modal con diferentes tipos de conexiones que se pueden realizar desde Power BI, en este caso se selecciona SQL Server, y se presiona el botón “Connect”

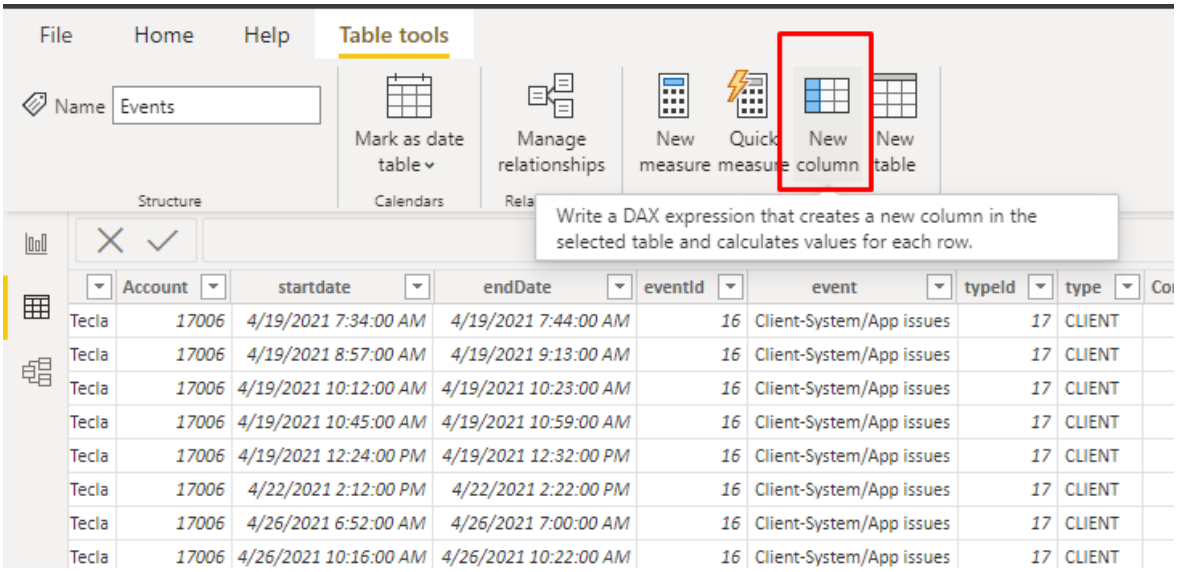


3. Al realizar el paso anterior, aparece un modal que pide ingresar la información para la conexión, se ingresa el nombre del servidor al cual se quiere conectar, la base de datos y el tipo de conexión. Si se elige “Import” se hará una toma de la información de ese momento y guardará en cache dicha información y en la sección “Data” permite usar todas las opciones para la transformación de datos, en caso de elegir “DirectQuery” la información se estará actualizando cada cierto tiempo con la limitante que no se puede utilizar todas las opciones para la transformación de datos.

4. Para el caso desarrollado, se seleccionó “Import” y se mapean las tablas seleccionadas. Luego pasamos a la sección “Data”, en este punto se mostrará un ejemplo de agregar una columna adicional física y otra calculada.
    - Columna física.
      Se selecciona la tabla donde se realizará la transformación de datos, para este ejemplo se hará en la tabla de eventos.

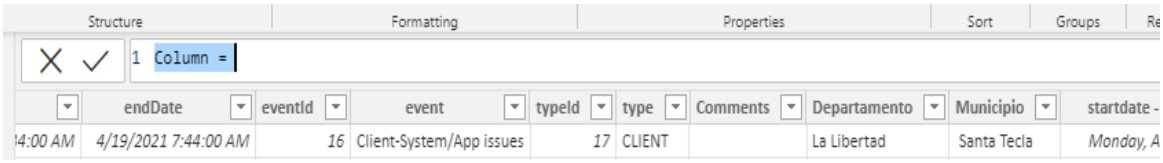


Luego en el menú, vamos a la opción de nueva columna, para este ejemplo se agregará una columna para filtrar por años.

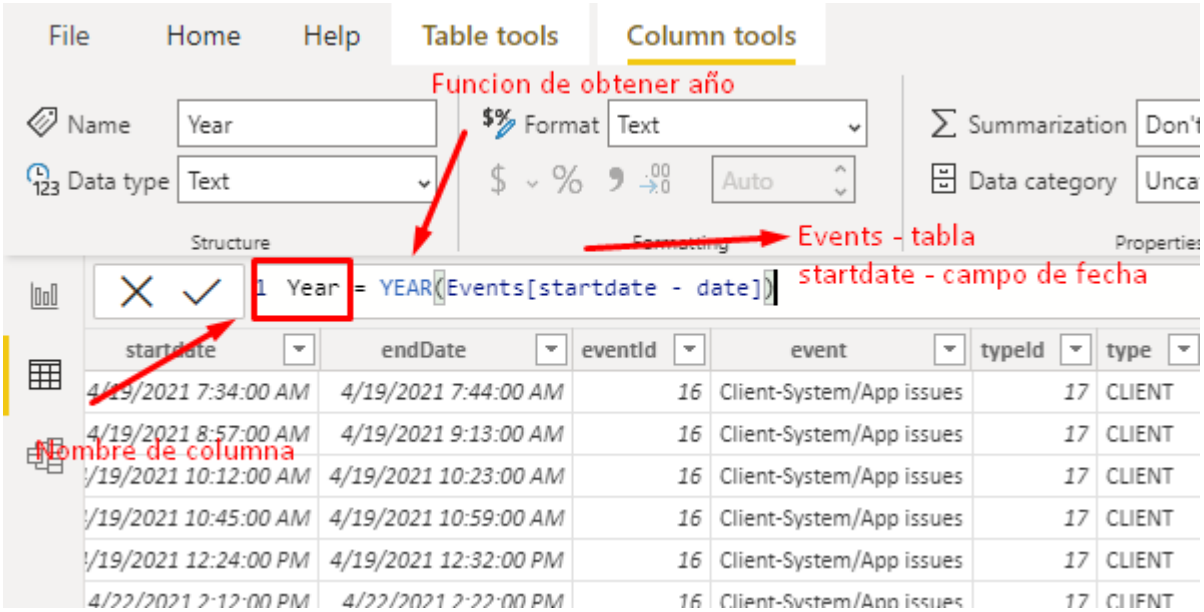


Account	startdate	endDate	eventId	event	typeId	type
Tecla	17006	4/19/2021 7:34:00 AM	4/19/2021 7:44:00 AM	16	Client-System/App issues	17	CLIENT
Tecla	17006	4/19/2021 8:57:00 AM	4/19/2021 9:13:00 AM	16	Client-System/App issues	17	CLIENT
Tecla	17006	4/19/2021 10:12:00 AM	4/19/2021 10:23:00 AM	16	Client-System/App issues	17	CLIENT
Tecla	17006	4/19/2021 10:45:00 AM	4/19/2021 10:59:00 AM	16	Client-System/App issues	17	CLIENT
Tecla	17006	4/19/2021 12:24:00 PM	4/19/2021 12:32:00 PM	16	Client-System/App issues	17	CLIENT
Tecla	17006	4/22/2021 2:12:00 PM	4/22/2021 2:22:00 PM	16	Client-System/App issues	17	CLIENT
Tecla	17006	4/26/2021 6:52:00 AM	4/26/2021 7:00:00 AM	16	Client-System/App issues	17	CLIENT
Tecla	17006	4/26/2021 10:16:00 AM	4/26/2021 10:22:00 AM	16	Client-System/App issues	17	CLIENT

Se habilitará la sección de fórmulas y mostrará el nombre “Column”, se puede cambiar ese nombre por el que llevará la columna.



Se utiliza la siguiente función DAX llamada Year y se asigna el campo de fecha del evento, se presiona el check para confirmar la configuración de la columna.

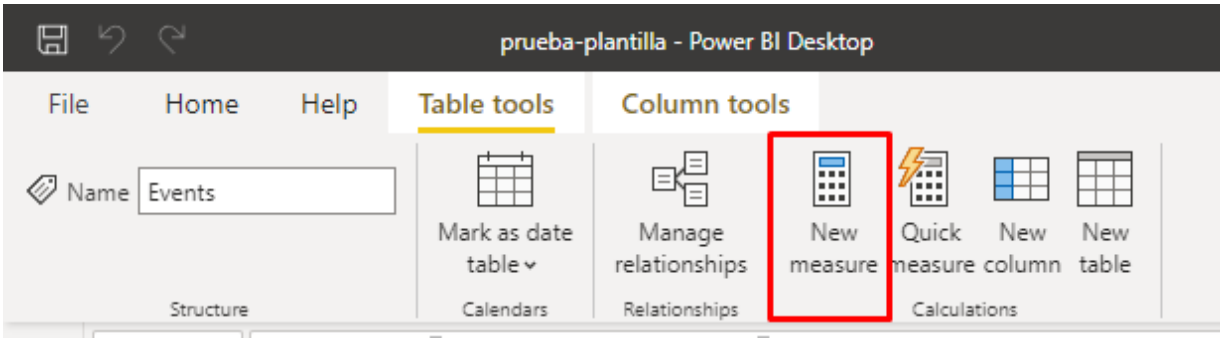


Esto nos creará una nueva columna donde se muestre solo el año de la fecha que se configuró.

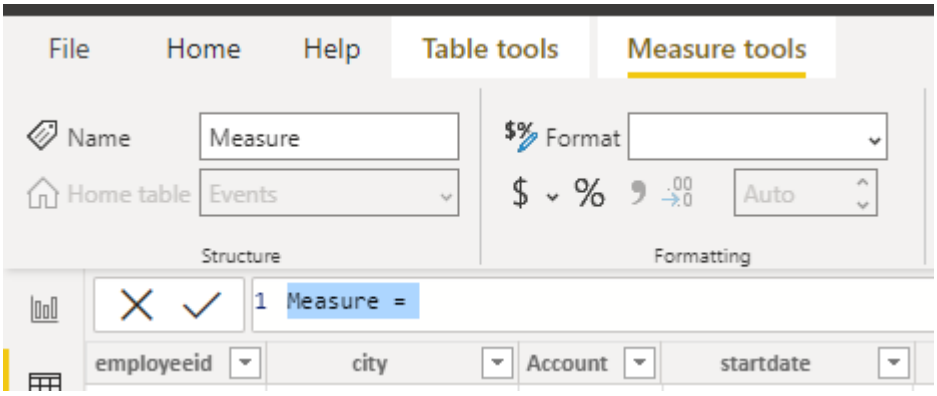
startdate - date	startdate - time	endDate - date	endDate - time	Year
Monday, April 19, 2021	7:34:00 AM	Monday, April 19, 2021	7:44:00 AM	2021
Monday, April 19, 2021	8:57:00 AM	Monday, April 19, 2021	9:13:00 AM	2021
Monday, April 19, 2021	10:12:00 AM	Monday, April 19, 2021	10:23:00 AM	2021
Monday, April 19, 2021	10:45:00 AM	Monday, April 19, 2021	10:59:00 AM	2021
Monday, April 19, 2021	12:24:00 PM	Monday, April 19, 2021	12:32:00 PM	2021
Thursday, April 22, 2021	2:12:00 PM	Thursday, April 22, 2021	2:22:00 PM	2021
Monday, April 26, 2021	6:52:00 AM	Monday, April 26, 2021	7:00:00 AM	2021
Monday, April 26, 2021	10:16:00 AM	Monday, April 26, 2021	10:22:00 AM	2021
Monday, April 26, 2021	12:00:00 PM	Monday, April 26, 2021	12:07:00 PM	2021
Tuesday, April 27, 2021	9:19:00 AM	Tuesday, April 27, 2021	9:32:00 AM	2021
Wednesday, April 28, 2021	9:59:00 AM	Wednesday, April 28, 2021	10:12:00 AM	2021
Wednesday, April 28, 2021	1:25:00 PM	Wednesday, April 28, 2021	1:35:00 PM	2021
Thursday, April 29, 2021	6:40:00 AM	Thursday, April 29, 2021	6:45:00 AM	2021
Monday, May 3, 2021	6:00:00 AM	Monday, May 3, 2021	6:05:00 AM	2021
Monday, May 3, 2021	11:55:00 AM	Monday, May 3, 2021	12:04:00 PM	2021
Tuesday, May 4, 2021	6:00:00 AM	Tuesday, May 4, 2021	6:10:00 AM	2021
Tuesday, May 4, 2021	10:23:00 AM	Tuesday, May 4, 2021	10:29:00 AM	2021
Tuesday, May 4, 2021	12:12:00 PM	Tuesday, May 4, 2021	12:22:00 PM	2021
Friday, April 30, 2021	9:42:00 AM	Friday, April 30, 2021	9:46:00 AM	2021
Friday, April 30, 2021	11:57:00 AM	Friday, April 30, 2021	12:01:00 PM	2021
Monday, May 10, 2021	8:12:00 AM	Monday, May 10, 2021	8:17:00 AM	2021
Monday, May 10, 2021	11:26:00 AM	Monday, May 10, 2021	11:33:00 AM	2021
Friday, May 7, 2021	12:04:00 PM	Friday, May 7, 2021	12:10:00 PM	2021
Wednesday, May 5, 2021	6:30:00 AM	Wednesday, May 5, 2021	6:37:00 AM	2021

- Columna Calculada o Medible

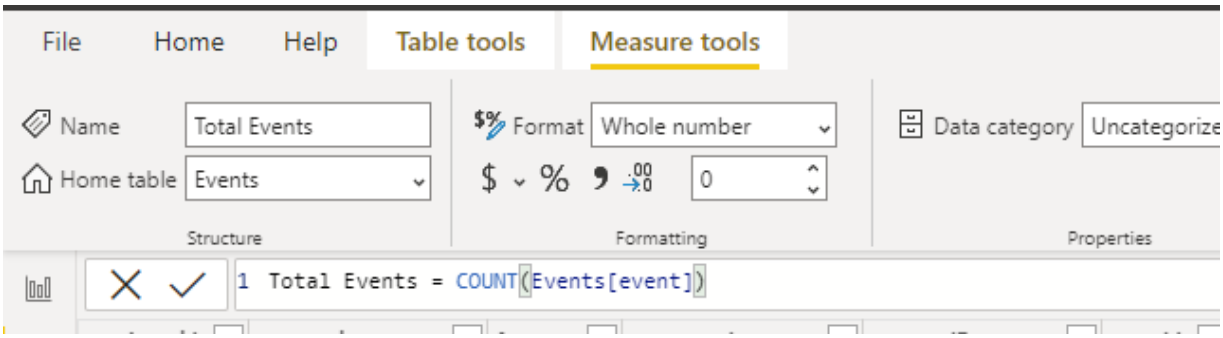
En la pestaña de herramientas de tabla, seleccionamos la opción nueva medición,



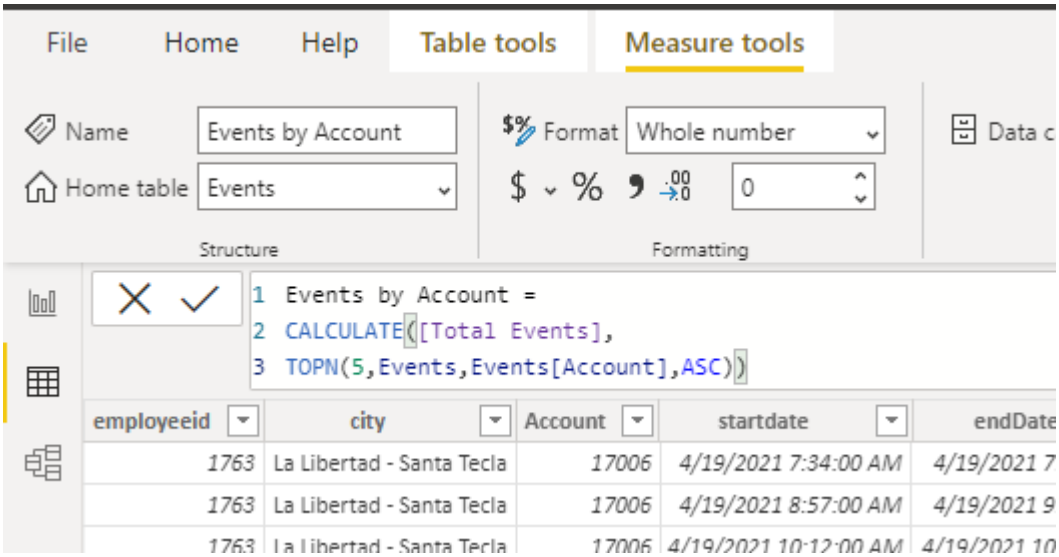
En la sección de formula aparece el nombre “Measure”, así como en el paso anterior ese nombre se puede editar y es el que corresponde a la nueva medición.



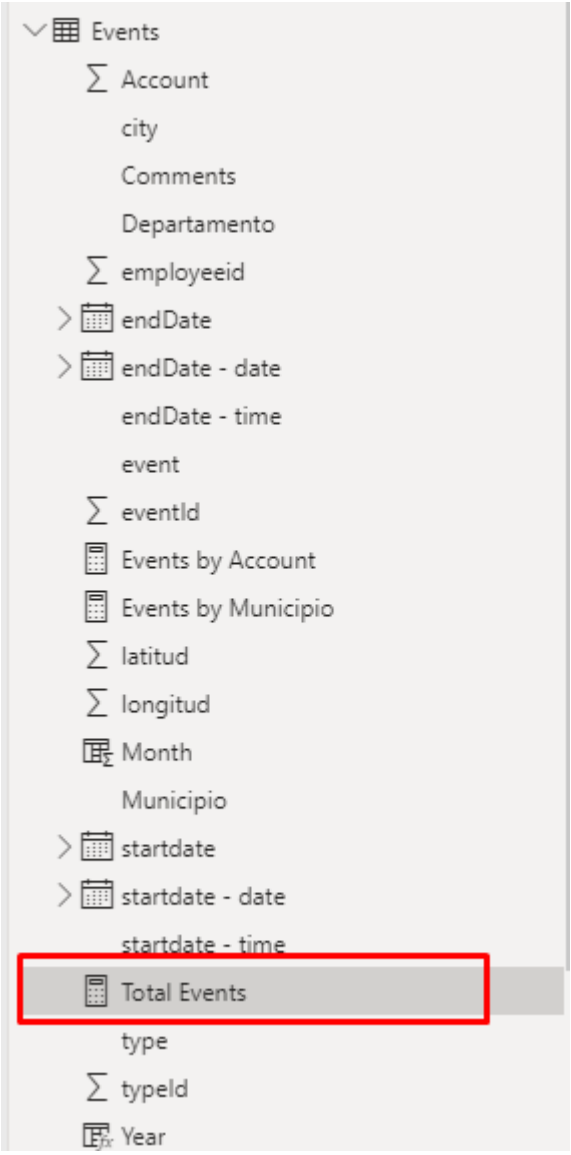
Se hará el ejemplo de crear una medición que cuente la cantidad de eventos de la siguiente forma.



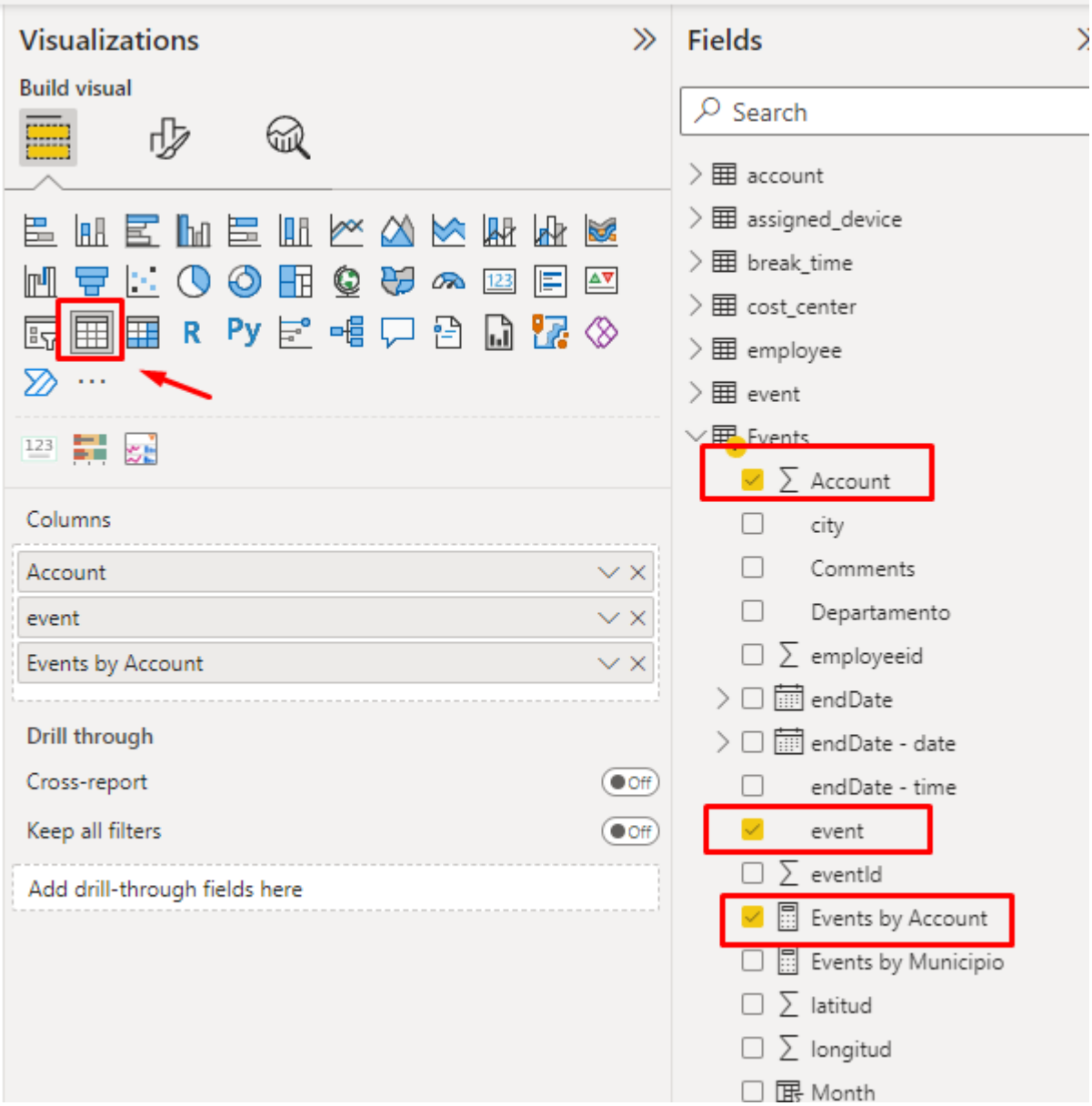
Luego vamos a usar la medición de total de ventos para calcular los eventos por cuentas, que se hará de la siguiente forma:



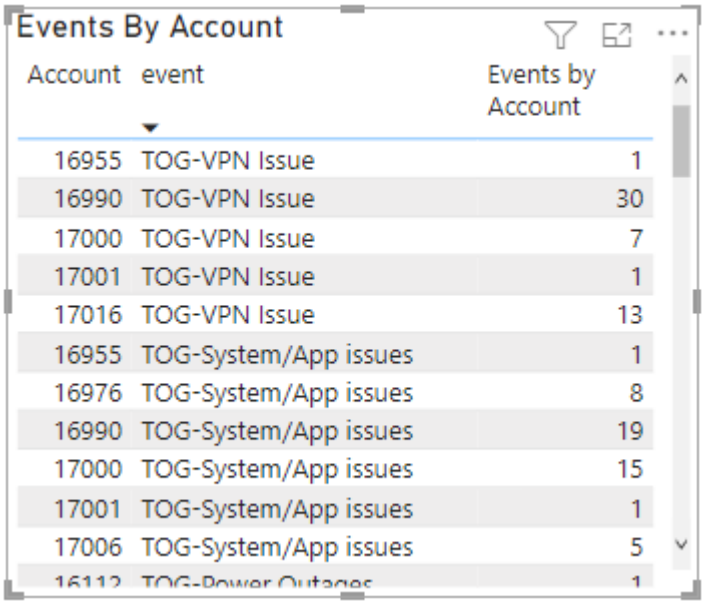
Al crear este tipo de mediciones no se visualizan en la tabla ni se crea un nuevo campo en ella, solo se reflejan al lado derecho bajo el nombre de la tabla y con un icono de calculadora.



5. En la sección de reporte, se presenta como usar estos campos para agregarlos a tablas, En el panel de visualización seleccionamos la opción de tabla, luego arrastramos los campos a mostrar, en este caso son los de cuentas, eventos, y eventos por cuenta.


Visualizations
Build visual
Columns
Account
event
Events by Account
Drill through
Cross-report
Off
Keep all filters
Off
Add drill-through fields here
Fields
Search
account
assigned_device
break_time
cost_center
employee
event
Events
Account
city
Comments
Departamento
employeeid
endDate
endDate - date
endDate - time
event
eventId
Events by Account
Events by Municipio
latitud
longitud
Month

La tabla presentará esta información.



Events By Account

Account	event	Events by Account
16955	TOG-VPN Issue	1
16990	TOG-VPN Issue	30
17000	TOG-VPN Issue	7
17001	TOG-VPN Issue	1
17016	TOG-VPN Issue	13
16955	TOG-System/App issues	1
16976	TOG-System/App issues	8
16990	TOG-System/App issues	19
17000	TOG-System/App issues	15
17001	TOG-System/App issues	1
17006	TOG-System/App issues	5
16112	TOG-Power Outages	1

Anexo 2: Script de creación de la base de datos

```
IF EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[user_role]') AND type in (N'U'))
DROP TABLE [dbo].[user_role]
GO
IF  EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[form]') AND type in (N'U'))
DROP TABLE [dbo].[form]
GO
IF EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[sys_user]') AND type in (N'U'))
DROP TABLE [dbo].[sys_user]
GO
IF  EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[event]') AND type in (N'U'))
DROP TABLE [dbo].[event]
GO
IF EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[break_time]') AND type in (N'U'))
DROP TABLE [dbo].[break_time]
GO
IF EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[schedule]') AND type in (N'U'))
DROP TABLE [dbo].[schedule]
GO
IF EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[assigned_device]') AND type in (N'U'))
DROP TABLE [dbo].[assigned_device]
GO
IF EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[employee]') AND type in (N'U'))
DROP TABLE [dbo].[employee]
GO
IF EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[cost_center]') AND type in (N'U'))
DROP TABLE [dbo].[cost_center]
GO
```

```
IF EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[home_location]') AND type in (N'U'))
DROP TABLE [dbo].[home_location]
GO
IF EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[office_location]') AND type in (N'U'))
DROP TABLE [dbo].[office_location]
GO
IF EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[location]') AND type in (N'U'))
DROP TABLE [dbo].[location]
GO
IF EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[account]') AND type in (N'U'))
DROP TABLE [dbo].[account]
GO
IF  EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[role]') AND type in (N'U'))
DROP TABLE [dbo].[role]
GO
IF EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[sys_catalog]') AND type in (N'U'))
DROP TABLE [dbo].[sys_catalog]
GO

CREATE TABLE [account]
(
        [account_id]        uniqueidentifier DEFAULT newid() NOT NULL ,
 [account_code]      varchar(50)  NULL ,
        [description]       varchar(100)  NULL ,
        [state]             char(1)  NULL ,
        [created_on]        datetime DEFAULT GETDATE() NULL ,
        [created_by]        char(18)  NULL ,
        [updated_on]        datetime DEFAULT GETDATE() NULL ,
        [updated_by]        char(18)  NULL
)
go
```

```
CREATE TABLE [assigned_device]
(
        [assigned_device_id] uniqueidentifier DEFAULT newid() NOT NULL ,
        [employee_id]       uniqueidentifier  NOT NULL ,
        [office_location_id] uniqueidentifier  NOT NULL ,
        [home_location_id]   uniqueidentifier  NOT NULL ,
        [account_id]        uniqueidentifier  NOT NULL ,
        [cost_center_id]    uniqueidentifier  NOT NULL ,
        [device_type_id]    uniqueidentifier  NULL ,
        [description]       varchar(500)  NULL ,
        [assigned_date_time] datetime  NULL ,
        [created_on]        datetime DEFAULT GETDATE() NULL ,
        [created_by]        char(18)  NULL ,
        [updated_on]        datetime DEFAULT GETDATE() NULL ,
        [updated_by]        char(18)  NULL
)
go

CREATE TABLE [break_time]
(
        [break_time_id]     uniqueidentifier DEFAULT newid() NOT NULL ,
        [employee_id]       uniqueidentifier  NOT NULL ,
        [office_location_id] uniqueidentifier  NOT NULL ,
        [home_location_id]   uniqueidentifier  NOT NULL ,
        [account_id]        uniqueidentifier  NOT NULL ,
        [cost_center_id]    uniqueidentifier  NOT NULL ,
        [break_start_time]   datetime  NULL ,
        [break_end_time]    datetime  NULL ,
        [break_type_id]     uniqueidentifier  NULL ,
        [created_on]        datetime DEFAULT GETDATE() NULL ,
        [created_by]        char(18)  NULL ,
        [updated_on]        datetime DEFAULT GETDATE() NULL ,
        [updated_by]        char(18)  NULL
)
go
```

```
CREATE TABLE [cost_center]
(
        [cost_center_id]     uniqueidentifier DEFAULT newid() NOT NULL ,
        [cost_center_code]   varchar(50)  NULL ,
        [state]              char(1)  NULL ,
        [description]        varchar(100)  NULL ,
        [created_on]         datetime DEFAULT GETDATE() NULL ,
        [created_by]         char(18)  NULL ,
        [updated_on]         datetime DEFAULT GETDATE() NULL ,
        [updated_by]         char(18)  NULL
)
go

CREATE TABLE [employee]
(
        [employee_id]        uniqueidentifier DEFAULT newid() NOT NULL ,
        [office_location_id] uniqueidentifier  NOT NULL ,
        [home_location_id]   uniqueidentifier  NOT NULL ,
        [account_id]         uniqueidentifier  NOT NULL ,
 [cost_center_id]     uniqueidentifier  NOT NULL ,
        [position_id]        uniqueidentifier  NULL ,
        [departament_id]     uniqueidentifier  NULL ,
        [employee_id_supervisor] uniqueidentifier  NULL ,
        [employee_id_manager] uniqueidentifier  NULL ,
 [employee_code]      varchar(50)  NULL ,
        [start_date]         datetime  NULL ,
        [end_date]           date  NULL ,
        [timezone_id]        uniqueidentifier  NULL ,
        [workplace_type_id]  uniqueidentifier  NULL ,
        [assigned_device_id] uniqueidentifier  NULL ,
        [internet_provider_id] uniqueidentifier  NULL ,
        [language_code]      varchar(10)  NULL ,
        [created_on]         datetime DEFAULT GETDATE() NULL ,
        [created_by]         char(18)  NULL ,
        [updated_on]         datetime DEFAULT GETDATE() NULL ,
        [updated_by]         char(18)  NULL
)
go
```

```
CREATE TABLE [event]
(
	[event_id]		uniqueidentifier DEFAULT newid() NOT NULL ,
	[office_location_id] uniqueidentifier  NOT NULL ,
	[home_location_id]	uniqueidentifier  NOT NULL ,
	[event_category_id]  uniqueidentifier  NULL ,
	[event_type_id]	uniqueidentifier  NULL ,
	[event_start_date_time] datetime  NULL ,
	[event_end_date_time] char(18)  NULL ,
	[employee_id_reporter] uniqueidentifier  NULL ,
	[employee_id_affected] uniqueidentifier  NULL ,
	[account_id_affected] uniqueidentifier  NULL ,
	[reason_id]		uniqueidentifier  NULL ,
	[reason_comments]	varchar(2000)  NULL ,
	[event_disposition_type_id] uniqueidentifier  NULL ,
	[event_disposition_comments] varchar(2000)  NULL ,
	[reception_type_id]  uniqueidentifier  NULL ,
	[opened_date_time]	datetime  NULL ,
	[resolution_date_time] datetime  NULL ,
	[calendar_duration_time] integer  NULL ,
	[business_duration_time] integer  NULL ,
	[created_on]		datetime DEFAULT GETDATE() NULL ,
	[created_by]		char(18)  NULL ,
	[updated_on]		datetime DEFAULT GETDATE() NULL ,
	[updated_by]		char(18)  NULL
)
go

CREATE TABLE [form]
(
	[form_id]		uniqueidentifier DEFAULT newid() NOT NULL ,
	[form_code]		varchar(50)  NULL ,
	[description]		varchar(100)  NULL ,
	[state]		char(1)  NULL ,
	[created_on]		datetime DEFAULT GETDATE() NULL ,
	[created_by]		char(18)  NULL ,
	[updated_on]		datetime DEFAULT GETDATE() NULL ,
	[updated_by]		char(18)  NULL
```

```
)
go

CREATE TABLE [home_location]
(
        [home_location_id]   uniqueidentifier DEFAULT newid() NOT NULL ,
        [neighborhood_id]    uniqueidentifier  NULL ,
        [town_id]            uniqueidentifier  NULL ,
        [city_id]           uniqueidentifier  NULL ,
        [zone_id]            uniqueidentifier  NULL ,
        [country_id]         uniqueidentifier  NULL ,
        [region_id]          uniqueidentifier  NULL ,
        [address_line1]      varchar(100)  NULL ,
        [address_line2]      varchar(100)  NULL ,
        [address_line3]      varchar(100)  NULL ,
        [created_on]         datetime DEFAULT GETDATE() NULL ,
        [created_by]         char(18)  NULL ,
        [updated_on]         datetime DEFAULT GETDATE() NULL ,
        [updated_by]         char(18)  NULL
)
go

CREATE TABLE [location]
(
        [location_id]        uniqueidentifier DEFAULT newid() NOT NULL ,
        [parent_location_id] uniqueidentifier  NULL ,
        [location_type]      varchar(50)  NULL ,
        [location_code]      varchar(150)  NULL ,
        [description]        varchar(500)  NULL ,
        [latitude]           decimal(25,10)  NULL ,
        [longitude]          decimal(25,10)  NULL ,
        [created_on]         datetime DEFAULT GETDATE() NULL ,
        [created_by]         char(18)  NULL ,
        [updated_on]         datetime DEFAULT GETDATE() NULL ,
        [updated_by]         char(18)  NULL
)
go
```

```
CREATE TABLE [office_location]
(
	[office_location_id] uniqueidentifier DEFAULT newid() NOT NULL ,
	[neighborhood_id]    uniqueidentifier  NULL ,
	[town_id]            uniqueidentifier  NULL ,
	[city_id]           uniqueidentifier  NULL ,
	[zone_id]           uniqueidentifier  NULL ,
	[country_id]         uniqueidentifier  NULL ,
	[region_id]         uniqueidentifier  NULL ,
	[address_line1]      varchar(100)  NULL ,
	[address_line2]      varchar(100)  NULL ,
	[address_line3]      char(18)  NULL ,
	[created_on]         datetime DEFAULT GETDATE() NULL ,
	[created_by]         char(18)  NULL ,
	[updated_on]         datetime DEFAULT GETDATE() NULL ,
	[updated_by]         char(18)  NULL
)
go

CREATE TABLE [role]
(
	[role_id]           uniqueidentifier DEFAULT newid() NOT NULL ,
	[description]       varchar(100)  NULL ,
	[state]            char(1)  NULL ,
	[created_on]         datetime DEFAULT GETDATE() NULL ,
	[created_by]         char(18)  NULL ,
	[updated_on]         datetime DEFAULT GETDATE() NULL ,
	[updated_by]         char(18)  NULL
)
go

CREATE TABLE [schedule]
(
	[schedule_id]        char(18) DEFAULT newid() NOT NULL ,
	[employee_id]        uniqueidentifier  NOT NULL ,
	[office_location_id] uniqueidentifier  NOT NULL ,
	[home_location_id]   uniqueidentifier  NOT NULL ,
	[account_id]         uniqueidentifier  NOT NULL ,
```

```
	[cost_center_id]	uniqueidentifier  NOT NULL ,
	[date]		date  NOT NULL ,
	[weekday]		varchar(10)  NULL ,
	[availability]		varchar(50)  NULL ,
	[time_in]		Time  NULL ,
	[time_out]		Time  NULL ,
	[created_on]		datetime DEFAULT GETDATE() NULL ,
	[created_by]		char(18)  NULL ,
	[updated_on]		datetime DEFAULT GETDATE() NULL ,
	[updated_by]		char(18)  NULL
)
go

CREATE TABLE [sys_catalog]
(
	[catalog_id]		uniqueidentifier DEFAULT newid() NOT NULL ,
	[form_code]		varchar(100)  NULL ,
	[catalog_type_code]	varchar(100)  NULL ,
	[language_code]		varchar(10)  NULL ,
	[value]		varchar(100)  NULL ,
	[label]		varchar(200)  NULL ,
	[parent_catalog_id]	uniqueidentifier  NULL ,
	[state]		char(1)  NULL ,
	[hint]		varchar(100)  NULL ,
	[sequence]		integer  NULL ,
	[created_on]		datetime DEFAULT GETDATE() NULL ,
	[created_by]		char(18)  NULL ,
	[updated_on]		datetime DEFAULT GETDATE() NULL ,
	[updated_by]		char(18)  NULL
)
go

CREATE TABLE [sys_user]
(
	[user_id]		uniqueidentifier DEFAULT newid() NOT NULL ,
	[user_name]		varchar(20)  NULL ,
	[user_password]		varchar(20)  NULL ,
	[name]		varchar(20)  NULL ,
```

```
	[last_name]		char(18)  NULL ,
	[state]		char(1)  NULL ,
	[created_on]		datetime DEFAULT GETDATE() NULL ,
	[created_by]		char(18)  NULL ,
	[updated_on]		datetime DEFAULT GETDATE() NULL ,
	[updated_by]		char(18)  NULL
)
go

CREATE TABLE [user_role]
(
	[user_id]		uniqueidentifier  NOT NULL ,
	[role_id]		uniqueidentifier  NOT NULL ,
	[form_id]		uniqueidentifier  NOT NULL ,
	[created_on]		datetime DEFAULT GETDATE() NULL ,
	[created_by]		char(18)  NULL ,
	[updated_on]		datetime DEFAULT GETDATE() NULL ,
	[updated_by]		char(18)  NULL
)
go

ALTER TABLE [account]
	ADD CONSTRAINT [XPKaccount] PRIMARY KEY  CLUSTERED ([account_id] ASC)
go

ALTER TABLE [assigned_device]
	ADD CONSTRAINT [XPKassigned_device] PRIMARY KEY  CLUSTERED ([assigned_device_id] ASC,[employee_id] ASC,[office_location_id] ASC,[home_location_id] ASC,[account_id] ASC,[cost_center_id] ASC)
go

ALTER TABLE [break_time]
	ADD CONSTRAINT [XPKbreak_time] PRIMARY KEY  CLUSTERED ([break_time_id] ASC,[employee_id] ASC,[office_location_id] ASC,[home_location_id] ASC,[account_id] ASC,[cost_center_id] ASC)
go
```

```
ALTER TABLE [cost_center]
	ADD CONSTRAINT [XPKcost_center] PRIMARY KEY CLUSTERED ([cost_center_id] ASC)
go

ALTER TABLE [employee]
	ADD CONSTRAINT [XPKemployee] PRIMARY KEY CLUSTERED ([employee_id] ASC,[office_location_id] ASC,[home_location_id] ASC,[account_id] ASC,[cost_center_id] ASC)
go

ALTER TABLE [event]
	ADD CONSTRAINT [XPKevent] PRIMARY KEY CLUSTERED ([event_id] ASC,[office_location_id] ASC,[home_location_id] ASC)
go

ALTER TABLE [form]
	ADD CONSTRAINT [XPKform] PRIMARY KEY CLUSTERED ([form_id] ASC)
go

ALTER TABLE [home_location]
	ADD CONSTRAINT [XPKhome_location] PRIMARY KEY CLUSTERED ([home_location_id] ASC)
go

ALTER TABLE [location]
	ADD CONSTRAINT [XPKlocation] PRIMARY KEY CLUSTERED ([location_id] ASC)
go

ALTER TABLE [office_location]
	ADD CONSTRAINT [XPKoffice_location] PRIMARY KEY CLUSTERED ([office_location_id] ASC)
go

ALTER TABLE [role]
	ADD CONSTRAINT [XPKrole] PRIMARY KEY CLUSTERED ([role_id] ASC)
go
```

```
ALTER TABLE [schedule]
	ADD CONSTRAINT [XPKschedule] PRIMARY KEY  CLUSTERED ([schedule_id] ASC,[employee_id] ASC,[office_location_id] ASC,[home_location_id] ASC,[account_id] ASC,[cost_center_id] ASC,[date] ASC)
go

ALTER TABLE [sys_catalog]
	ADD CONSTRAINT [XPKsys_catalog] PRIMARY KEY  CLUSTERED ([catalog_id] ASC)
go

ALTER TABLE [sys_user]
	ADD CONSTRAINT [XPKsys_user] PRIMARY KEY  CLUSTERED ([user_id] ASC)
go

ALTER TABLE [user_role]
	ADD CONSTRAINT [XPKuser_role] PRIMARY KEY  CLUSTERED ([user_id] ASC,[role_id] ASC,[form_id] ASC)
go
ALTER TABLE [assigned_device]
	ADD CONSTRAINT [R_12] FOREIGN KEY ([employee_id],[office_location_id],[home_location_id],[account_id],[cost_center_id])
REFERENCES
[employee]([employee_id],[office_location_id],[home_location_id],[account_id],[cost_center_id])
		ON DELETE NO ACTION
		ON UPDATE NO ACTION
go

ALTER TABLE [break_time]
	ADD CONSTRAINT [R_21] FOREIGN KEY ([employee_id],[office_location_id],[home_location_id],[account_id],[cost_center_id])
REFERENCES
[employee]([employee_id],[office_location_id],[home_location_id],[account_id],[cost_center_id])
		ON DELETE NO ACTION
		ON UPDATE NO ACTION
go
```

```
ALTER TABLE [employee]
	ADD CONSTRAINT [R_9] FOREIGN KEY ([office_location_id]) REFERENCES [office_location]([office_location_id])
		ON DELETE NO ACTION
		ON UPDATE NO ACTION
go

ALTER TABLE [employee]
	ADD CONSTRAINT [R_10] FOREIGN KEY ([home_location_id]) REFERENCES [home_location]([home_location_id])
		ON DELETE NO ACTION
		ON UPDATE NO ACTION
go

ALTER TABLE [employee]
	ADD CONSTRAINT [R_17] FOREIGN KEY ([office_location_id]) REFERENCES [office_location]([office_location_id])
		ON DELETE NO ACTION
		ON UPDATE NO ACTION
go

ALTER TABLE [employee]
	ADD CONSTRAINT [R_20] FOREIGN KEY ([home_location_id]) REFERENCES [home_location]([home_location_id])
		ON DELETE NO ACTION
		ON UPDATE NO ACTION
go

ALTER TABLE [employee]
	ADD CONSTRAINT [R_30] FOREIGN KEY ([account_id]) REFERENCES [account]([account_id])
		ON DELETE NO ACTION
		ON UPDATE NO ACTION
go

ALTER TABLE [employee]
	ADD CONSTRAINT [R_35] FOREIGN KEY ([cost_center_id]) REFERENCES [cost_center]([cost_center_id])
```

```
        ON DELETE NO ACTION
        ON UPDATE NO ACTION
go

ALTER TABLE [event]
    ADD CONSTRAINT [R_23] FOREIGN KEY ([office_location_id]) REFERENCES
[office_location]([office_location_id])
        ON DELETE NO ACTION
        ON UPDATE NO ACTION
go

ALTER TABLE [event]
    ADD CONSTRAINT [R_24] FOREIGN KEY ([home_location_id]) REFERENCES
[home_location]([home_location_id])
        ON DELETE NO ACTION
        ON UPDATE NO ACTION
go

ALTER TABLE [schedule]
    ADD CONSTRAINT [R_34] FOREIGN KEY
([employee_id],[office_location_id],[home_location_id],[account_id],[cost_center_id])
REFERENCES
[employee]([employee_id],[office_location_id],[home_location_id],[account_id],[cost_center
_id])
        ON DELETE NO ACTION
        ON UPDATE NO ACTION
go

ALTER TABLE [user_role]
    ADD CONSTRAINT [R_25] FOREIGN KEY ([user_id]) REFERENCES
[sys_user]([user_id])
        ON DELETE NO ACTION
        ON UPDATE NO ACTION
go

ALTER TABLE [user_role]
    ADD CONSTRAINT [R_26] FOREIGN KEY ([role_id]) REFERENCES
[role]([role_id])
```

```
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
go

ALTER TABLE [user_role]
        ADD CONSTRAINT [R_27] FOREIGN KEY ([form_id]) REFERENCES
[form]([form_id])
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
go

CREATE TRIGGER tD_account ON account FOR DELETE AS
/* erwin Builtin Trigger */
/* DELETE trigger on account */
BEGIN
  DECLARE  @errno   int,
           @severity int,
           @state    int,
           @errmsg  varchar(255)
    /* erwin Builtin Trigger */
    /* account  employee on parent delete no action */
    /*           ERWIN_RELATION:CHECKSUM="00011234",          PARENT_OWNER="",
PARENT_TABLE="account"
    CHILD_OWNER="", CHILD_TABLE="employee"
    P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
    FK_CONSTRAINT="R_30", FK_COLUMNS="account_id" */
    IF EXISTS (
      SELECT * FROM deleted,employee
      WHERE
        /*  %JoinFKPK(employee,deleted," = "," AND") */
        employee.account_id = deleted.account_id
    )
    BEGIN
      SELECT @errno  = 30001,
             @errmsg = 'Cannot delete account because employee exists.'
      GOTO error
    END
```

```
   /* erwin Builtin Trigger */
   RETURN
error:
  RAISERROR (@errmsg, -- Message text.
          @severity, -- Severity (0~25).
          @state) -- State (0~255).
   rollback transaction
END

go
CREATE TRIGGER tU_account ON account FOR UPDATE AS
/* erwin Builtin Trigger */
/* UPDATE trigger on account */
BEGIN
 DECLARE  @numrows int,
        @nullcnt int,
        @validcnt int,
        @insaccount_id uniqueidentifier,
        @errno   int,
        @severity int,
        @state    int,
        @errmsg  varchar(255)

 SELECT @numrows = @@rowcount
 /* erwin Builtin Trigger */
 /* account  employee on parent update no action */
 /*           ERWIN_RELATION:CHECKSUM="00012b72",          PARENT_OWNER="",
PARENT_TABLE="account"
   CHILD_OWNER="", CHILD_TABLE="employee"
   P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
   FK_CONSTRAINT="R_30", FK_COLUMNS="account_id" */
 IF
   /* %ParentPK(" OR",UPDATE) */
   UPDATE(account_id)
 BEGIN
   IF EXISTS (
     SELECT * FROM deleted,employee
     WHERE
```

```
      /*  %JoinFKPK(employee,deleted," = "," AND") */
      employee.account_id = deleted.account_id
   )
   BEGIN
    SELECT @errno  = 30005,
          @errmsg = 'Cannot update account because employee exists.'
    GOTO error
   END
  END

  /* erwin Builtin Trigger */
  RETURN
error:
   RAISERROR (@errmsg, -- Message text.
           @severity, -- Severity (0~25).
           @state) -- State (0~255).
   rollback transaction
END
go

CREATE TRIGGER tD_assigned_device ON assigned_device FOR DELETE AS
/* erwin Builtin Trigger */
/* DELETE trigger on assigned_device */
BEGIN
  DECLARE  @errno   int,
         @severity int,
         @state    int,
         @errmsg  varchar(255)
   /* erwin Builtin Trigger */
   /* employee  assigned_device on child delete no action */
   /*           ERWIN_RELATION:CHECKSUM="00027055",          PARENT_OWNER="",
PARENT_TABLE="employee"
   CHILD_OWNER="", CHILD_TABLE="assigned_device"
   P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
   FK_CONSTRAINT="R_12",
FK_COLUMNS="home_location_id""office_location_id""employee_id""account_id""
           cost_center_id" */
   IF EXISTS (SELECT * FROM deleted,employee
```

```
    WHERE
      /* %JoinFKPK(deleted,employee," = "," AND") */
      deleted.employee_id = employee.employee_id AND
      deleted.account_id = employee.account_id AND
      deleted.office_location_id = employee.office_location_id AND
      deleted.home_location_id = employee.home_location_id AND
      deleted.cost_center_id = employee.cost_center_id AND
      NOT EXISTS (
        SELECT * FROM assigned_device
        WHERE
          /* %JoinFKPK(assigned_device,employee," = "," AND") */
          assigned_device.employee_id = employee.employee_id AND
          assigned_device.account_id = employee.account_id AND
          assigned_device.office_location_id = employee.office_location_id AND
          assigned_device.home_location_id = employee.home_location_id AND
          assigned_device.cost_center_id = employee.cost_center_id
      )
  )
  BEGIN
    SELECT @errno  = 30010,
          @errmsg = 'Cannot delete last assigned_device because employee exists.'
    GOTO error
  END

  /* erwin Builtin Trigger */
  RETURN
error:
  RAISERROR (@errmsg, -- Message text.
          @severity, -- Severity (0~25).
          @state) -- State (0~255).
  rollback transaction
END
go
```

```
CREATE TRIGGER tU_assigned_device ON assigned_device FOR UPDATE AS
/* erwin Builtin Trigger */
/* UPDATE trigger on assigned_device */
BEGIN
  DECLARE  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insassigned_device_id uniqueidentifier,
           @insemployee_id uniqueidentifier,
           @insoffice_location_id uniqueidentifier,
           @inshome_location_id uniqueidentifier,
           @insaccount_id uniqueidentifier,
           @inscost_center_id uniqueidentifier,
           @errno   int,
           @severity int,
           @state    int,
           @errmsg  varchar(255)

  SELECT @numrows = @@rowcount
  /* erwin Builtin Trigger */
  /* employee  assigned_device on child update no action */
  /*            ERWIN_RELATION:CHECKSUM="0002366c",            PARENT_OWNER="",
PARENT_TABLE="employee"
    CHILD_OWNER="", CHILD_TABLE="assigned_device"
    P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
    FK_CONSTRAINT="R_12",
FK_COLUMNS="office_location_id""home_location_id""employee_id""account_id""
             cost_center_id" */
  IF
    /* %ChildFK(" OR",UPDATE) */
    UPDATE(employee_id) OR
    UPDATE(account_id) OR
    UPDATE(office_location_id) OR
    UPDATE(home_location_id) OR
    UPDATE(cost_center_id)
  BEGIN
    SELECT @nullcnt = 0
    SELECT @validcnt = count(*)
```

```
     FROM inserted,employee
      WHERE
        /* %JoinFKPK(inserted,employee) */
        inserted.employee_id = employee.employee_id and
        inserted.account_id = employee.account_id and
        inserted.office_location_id = employee.office_location_id and
        inserted.home_location_id = employee.home_location_id and
        inserted.cost_center_id = employee.cost_center_id
   /* %NotnullFK(inserted," IS NULL","select @nullcnt = count(*) from inserted where","
AND") */

   IF @validcnt + @nullcnt != @numrows
   BEGIN
     SELECT @errno  = 30007,
          @errmsg = 'Cannot update assigned_device because employee does not exist.'
     GOTO error
   END
  END

  /* erwin Builtin Trigger */
  RETURN
error:
   RAISERROR (@errmsg, -- Message text.
           @severity, -- Severity (0~25).
           @state) -- State (0~255).
    rollback transaction
END
go

CREATE TRIGGER tD_break_time ON break_time FOR DELETE AS
/* erwin Builtin Trigger */
/* DELETE trigger on break_time */
BEGIN
  DECLARE  @errno   int,
         @severity int,
         @state    int,
         @errmsg  varchar(255)
    /* erwin Builtin Trigger */
```

```
   /* employee  break_time on child delete no action */
   /*           ERWIN_RELATION:CHECKSUM="00023968",           PARENT_OWNER="",
PARENT_TABLE="employee"
   CHILD_OWNER="", CHILD_TABLE="break_time"
   P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
   FK_CONSTRAINT="R_21",
FK_COLUMNS="office_location_id""home_location_id""employee_id""account_id""
            cost_center_id" */
   IF EXISTS (SELECT * FROM deleted,employee
     WHERE
       /* %JoinFKPK(deleted,employee," = "," AND") */
       deleted.employee_id = employee.employee_id AND
       deleted.account_id = employee.account_id AND
       deleted.office_location_id = employee.office_location_id AND
       deleted.home_location_id = employee.home_location_id AND
       deleted.cost_center_id = employee.cost_center_id AND
       NOT EXISTS (
         SELECT * FROM break_time
         WHERE
           /* %JoinFKPK(break_time,employee," = "," AND") */
           break_time.employee_id = employee.employee_id AND
           break_time.account_id = employee.account_id AND
           break_time.office_location_id = employee.office_location_id AND
           break_time.home_location_id = employee.home_location_id AND
           break_time.cost_center_id = employee.cost_center_id
       )
   )

   BEGIN
     SELECT @errno  = 30010,
          @errmsg = 'Cannot delete last break_time because employee exists.'
     GOTO error
   END

   /* erwin Builtin Trigger */
   RETURN
error:
```

```
RAISERROR (@errmsg, -- Message text.
           @severity, -- Severity (0~25).
           @state) -- State (0~255).
   rollback transaction
END
go

CREATE TRIGGER tU_break_time ON break_time FOR UPDATE AS
/* erwin Builtin Trigger */
/* UPDATE trigger on break_time */
BEGIN
  DECLARE  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insbreak_time_id uniqueidentifier,
           @insemployee_id uniqueidentifier,
           @insoffice_location_id uniqueidentifier,
           @inshome_location_id uniqueidentifier,
           @insaccount_id uniqueidentifier,
           @inscost_center_id uniqueidentifier,
           @errno   int,
           @severity int,
           @state    int,
           @errmsg  varchar(255)
```

```
SELECT @numrows = @@rowcount
  /* erwin Builtin Trigger */
  /* employee  break_time on child update no action */
  /*            ERWIN_RELATION:CHECKSUM="00021fdc",            PARENT_OWNER="",
PARENT_TABLE="employee"
    CHILD_OWNER="", CHILD_TABLE="break_time"
    P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
    FK_CONSTRAINT="R_21",
FK_COLUMNS="home_location_id""office_location_id""employee_id""account_id""
            cost_center_id" */
  IF
    /* %ChildFK(" OR",UPDATE) */
    UPDATE(employee_id) OR
    UPDATE(account_id) OR
    UPDATE(office_location_id) OR
    UPDATE(home_location_id) OR
    UPDATE(cost_center_id)
  BEGIN
    SELECT @nullcnt = 0
    SELECT @validcnt = count(*)
      FROM inserted,employee
        WHERE
          /* %JoinFKPK(inserted,employee) */
          inserted.employee_id = employee.employee_id and
          inserted.account_id = employee.account_id and
          inserted.office_location_id = employee.office_location_id and
          inserted.home_location_id = employee.home_location_id and
          inserted.cost_center_id = employee.cost_center_id
    /* %NotnullFK(inserted," IS NULL","select @nullcnt = count(*) from inserted where","
AND") */

    IF @validcnt + @nullcnt != @numrows
    BEGIN
      SELECT @errno  = 30007,
            @errmsg = 'Cannot update break_time because employee does not exist.'
      GOTO error
    END
  END
```

```
 /* erwin Builtin Trigger */
 RETURN
error:
  RAISERROR (@errmsg, -- Message text.
          @severity, -- Severity (0~25).
          @state) -- State (0~255).
   rollback transaction
END
go

CREATE TRIGGER tD_cost_center ON cost_center FOR DELETE AS
/* erwin Builtin Trigger */
/* DELETE trigger on cost_center */
BEGIN
 DECLARE  @errno   int,
        @severity int,
        @state    int,
        @errmsg  varchar(255)
   /* erwin Builtin Trigger */
   /* cost_center  employee on parent delete no action */
   /*           ERWIN_RELATION:CHECKSUM="00010fe5",           PARENT_OWNER="",
PARENT_TABLE="cost_center"
   CHILD_OWNER="", CHILD_TABLE="employee"
   P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
   FK_CONSTRAINT="R_35", FK_COLUMNS="cost_center_id" */
   IF EXISTS (
     SELECT * FROM deleted,employee
     WHERE
       /*  %JoinFKPK(employee,deleted," = "," AND") */
       employee.cost_center_id = deleted.cost_center_id
   )
   BEGIN
     SELECT @errno  = 30001,
          @errmsg = 'Cannot delete cost_center because employee exists.'
     GOTO error
   END
```

```
   /* erwin Builtin Trigger */
   RETURN
error:
  RAISERROR (@errmsg, -- Message text.
          @severity, -- Severity (0~25).
          @state) -- State (0~255).
   rollback transaction
END
go

CREATE TRIGGER tU_cost_center ON cost_center FOR UPDATE AS
/* erwin Builtin Trigger */
/* UPDATE trigger on cost_center */
BEGIN
 DECLARE  @numrows int,
        @nullcnt int,
        @validcnt int,
        @inscost_center_id uniqueidentifier,
        @errno   int,
        @severity int,
        @state    int,
        @errmsg  varchar(255)

 SELECT @numrows = @@rowcount
 /* erwin Builtin Trigger */
 /* cost_center  employee on parent update no action */
 /*            ERWIN_RELATION:CHECKSUM="00013efd",            PARENT_OWNER="",
PARENT_TABLE="cost_center"
   CHILD_OWNER="", CHILD_TABLE="employee"
   P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
   FK_CONSTRAINT="R_35", FK_COLUMNS="cost_center_id" */
 IF
   /* %ParentPK(" OR",UPDATE) */
   UPDATE(cost_center_id)
 BEGIN
   IF EXISTS (
     SELECT * FROM deleted,employee
     WHERE
```

```
      /*  %JoinFKPK(employee,deleted," = "," AND") */
      employee.cost_center_id = deleted.cost_center_id
   )
   BEGIN
    SELECT @errno  = 30005,
          @errmsg = 'Cannot update cost_center because employee exists.'
    GOTO error
   END
  END

  /* erwin Builtin Trigger */
  RETURN
error:
   RAISERROR (@errmsg, -- Message text.
           @severity, -- Severity (0~25).
           @state) -- State (0~255).
   rollback transaction
END
go

CREATE TRIGGER tD_employee ON employee FOR DELETE AS
/* erwin Builtin Trigger */
/* DELETE trigger on employee */
BEGIN
  DECLARE  @errno   int,
         @severity int,
         @state    int,
         @errmsg  varchar(255)
   /* erwin Builtin Trigger */
   /* employee  schedule on parent delete no action */
   /*           ERWIN_RELATION:CHECKSUM="000c3192",          PARENT_OWNER="",
PARENT_TABLE="employee"
   CHILD_OWNER="", CHILD_TABLE="schedule"
   P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
   FK_CONSTRAINT="R_34",
FK_COLUMNS="home_location_id""office_location_id""employee_id""account_id""
           cost_center_id" */
   IF EXISTS (
```

```
    SELECT * FROM deleted,schedule
    WHERE
      /*  %JoinFKPK(schedule,deleted," = "," AND") */
      schedule.employee_id = deleted.employee_id AND
      schedule.account_id = deleted.account_id AND
      schedule.office_location_id = deleted.office_location_id AND
      schedule.home_location_id = deleted.home_location_id AND
      schedule.cost_center_id = deleted.cost_center_id
   )
   BEGIN
    SELECT @errno  = 30001,
          @errmsg = 'Cannot delete employee because schedule exists.'
    GOTO error
   END

   /* erwin Builtin Trigger */
   /* employee  break_time on parent delete no action */
   /*           ERWIN_RELATION:CHECKSUM="00000000",          PARENT_OWNER="",
PARENT_TABLE="employee"
   CHILD_OWNER="", CHILD_TABLE="break_time"
   P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
   FK_CONSTRAINT="R_21",
FK_COLUMNS="home_location_id""office_location_id""employee_id""account_id""
           cost_center_id" */
   IF EXISTS (
    SELECT * FROM deleted,break_time
    WHERE
      /*  %JoinFKPK(break_time,deleted," = "," AND") */
      break_time.employee_id = deleted.employee_id AND
      break_time.account_id = deleted.account_id AND
      break_time.office_location_id = deleted.office_location_id AND
      break_time.home_location_id = deleted.home_location_id AND
      break_time.cost_center_id = deleted.cost_center_id
   )
   BEGIN
    SELECT @errno  = 30001,
          @errmsg = 'Cannot delete employee because break_time exists.'
    GOTO error
```

```
   END

   /* erwin Builtin Trigger */
   /* employee  assigned_device on parent delete no action */
   /*           ERWIN_RELATION:CHECKSUM="00000000",          PARENT_OWNER="",
PARENT_TABLE="employee"
   CHILD_OWNER="", CHILD_TABLE="assigned_device"
   P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
   FK_CONSTRAINT="R_12",
FK_COLUMNS="home_location_id""office_location_id""employee_id""account_id""
           cost_center_id" */
   IF EXISTS (
    SELECT * FROM deleted,assigned_device
    WHERE
     /*  %JoinFKPK(assigned_device,deleted," = "," AND") */
     assigned_device.employee_id = deleted.employee_id AND
     assigned_device.account_id = deleted.account_id AND
     assigned_device.office_location_id = deleted.office_location_id AND
     assigned_device.home_location_id = deleted.home_location_id AND
     assigned_device.cost_center_id = deleted.cost_center_id
   )
   BEGIN
    SELECT @errno  = 30001,
         @errmsg = 'Cannot delete employee because assigned_device exists.'
    GOTO error
   END

   /* erwin Builtin Trigger */
   /* cost_center  employee on child delete no action */
   /*           ERWIN_RELATION:CHECKSUM="00000000",          PARENT_OWNER="",
PARENT_TABLE="cost_center"
   CHILD_OWNER="", CHILD_TABLE="employee"
   P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
   FK_CONSTRAINT="R_35", FK_COLUMNS="cost_center_id" */
   IF EXISTS (SELECT * FROM deleted,cost_center
    WHERE
     /* %JoinFKPK(deleted,cost_center," = "," AND") */
     deleted.cost_center_id = cost_center.cost_center_id AND
```

```
    NOT EXISTS (
      SELECT * FROM employee
      WHERE
        /* %JoinFKPK(employee,cost_center," = "," AND") */
        employee.cost_center_id = cost_center.cost_center_id
    )
  )
  BEGIN
    SELECT @errno  = 30010,
         @errmsg = 'Cannot delete last employee because cost_center exists.'
    GOTO error
  END

  /* erwin Builtin Trigger */
  /* account  employee on child delete no action */
  /*           ERWIN_RELATION:CHECKSUM="00000000",           PARENT_OWNER="",
PARENT_TABLE="account"
  CHILD_OWNER="", CHILD_TABLE="employee"
  P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
  FK_CONSTRAINT="R_30", FK_COLUMNS="account_id" */
  IF EXISTS (SELECT * FROM deleted,account
    WHERE
      /* %JoinFKPK(deleted,account," = "," AND") */
      deleted.account_id = account.account_id AND
      NOT EXISTS (
        SELECT * FROM employee
        WHERE
          /* %JoinFKPK(employee,account," = "," AND") */
          employee.account_id = account.account_id
      )
  )
  BEGIN
    SELECT @errno  = 30010,
         @errmsg = 'Cannot delete last employee because account exists.'
    GOTO error
  END
```

```
 /* erwin Builtin Trigger */
   /* home_location  employee on child delete no action */
   /*           ERWIN_RELATION:CHECKSUM="00000000",           PARENT_OWNER="",
PARENT_TABLE="home_location"
   CHILD_OWNER="", CHILD_TABLE="employee"
   P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
   FK_CONSTRAINT="R_20", FK_COLUMNS="home_location_id" */
   IF EXISTS (SELECT * FROM deleted,home_location
    WHERE
     /* %JoinFKPK(deleted,home_location," = "," AND") */
     deleted.home_location_id = home_location.home_location_id AND
     NOT EXISTS (
       SELECT * FROM employee
       WHERE
        /* %JoinFKPK(employee,home_location," = "," AND") */
        employee.home_location_id = home_location.home_location_id
     )
   )
   BEGIN
    SELECT @errno  = 30010,
         @errmsg = 'Cannot delete last employee because home_location exists.'
    GOTO error
   END

   /* erwin Builtin Trigger */
   /* office_location  employee on child delete no action */
   /*           ERWIN_RELATION:CHECKSUM="00000000",           PARENT_OWNER="",
PARENT_TABLE="office_location"
   CHILD_OWNER="", CHILD_TABLE="employee"
   P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
   FK_CONSTRAINT="R_17", FK_COLUMNS="office_location_id" */
   IF EXISTS (SELECT * FROM deleted,office_location
    WHERE
     /* %JoinFKPK(deleted,office_location," = "," AND") */
     deleted.office_location_id = office_location.office_location_id AND
     NOT EXISTS (
       SELECT * FROM employee
       WHERE
```

```
        /* %JoinFKPK(employee,office_location," = "," AND") */
        employee.office_location_id = office_location.office_location_id
      )
    )
    BEGIN
     SELECT @errno  = 30010,
          @errmsg = 'Cannot delete last employee because office_location exists.'
     GOTO error
    END

    /* erwin Builtin Trigger */
    /* home_location  employee on child delete no action */
    /*           ERWIN_RELATION:CHECKSUM="00000000",          PARENT_OWNER="",
PARENT_TABLE="home_location"
    CHILD_OWNER="", CHILD_TABLE="employee"
    P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
    FK_CONSTRAINT="R_10", FK_COLUMNS="home_location_id" */
    IF EXISTS (SELECT * FROM deleted,home_location
     WHERE
      /* %JoinFKPK(deleted,home_location," = "," AND") */
      deleted.home_location_id = home_location.home_location_id AND
      NOT EXISTS (
       SELECT * FROM employee
       WHERE
        /* %JoinFKPK(employee,home_location," = "," AND") */
        employee.home_location_id = home_location.home_location_id
      )
    )
    BEGIN
     SELECT @errno  = 30010,
          @errmsg = 'Cannot delete last employee because home_location exists.'
     GOTO error
    END

    /* erwin Builtin Trigger */
    /* office_location  employee on child delete no action */
    /*           ERWIN_RELATION:CHECKSUM="00000000",          PARENT_OWNER="",
PARENT_TABLE="office_location"
```

```
   CHILD_OWNER="", CHILD_TABLE="employee"
   P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
   FK_CONSTRAINT="R_9", FK_COLUMNS="office_location_id" */
   IF EXISTS (SELECT * FROM deleted,office_location
    WHERE
     /* %JoinFKPK(deleted,office_location," = "," AND") */
     deleted.office_location_id = office_location.office_location_id AND
     NOT EXISTS (
      SELECT * FROM employee
      WHERE
       /* %JoinFKPK(employee,office_location," = "," AND") */
       employee.office_location_id = office_location.office_location_id
     )
   )
   BEGIN
    SELECT @errno  = 30010,
         @errmsg = 'Cannot delete last employee because office_location exists.'
    GOTO error
   END

   /* erwin Builtin Trigger */
   RETURN
error:
  RAISERROR (@errmsg, -- Message text.
          @severity, -- Severity (0~25).
          @state) -- State (0~255).
   rollback transaction
END
go

CREATE TRIGGER tU_employee ON employee FOR UPDATE AS
/* erwin Builtin Trigger */
/* UPDATE trigger on employee */
BEGIN
 DECLARE  @numrows int,
         @nullcnt int,
         @validcnt int,
         @insemployee_id uniqueidentifier,
```

```
        @insaccount_id uniqueidentifier,
        @insoffice_location_id uniqueidentifier,
        @inshome_location_id uniqueidentifier,
        @inscost_center_id uniqueidentifier,
        @errno   int,
        @severity int,
        @state    int,
        @errmsg  varchar(255)

 SELECT @numrows = @@rowcount
 /* erwin Builtin Trigger */
 /* employee  schedule on parent update no action */
 /*            ERWIN_RELATION:CHECKSUM="000db756",           PARENT_OWNER="",
PARENT_TABLE="employee"
   CHILD_OWNER="", CHILD_TABLE="schedule"
   P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
   FK_CONSTRAINT="R_34",
FK_COLUMNS="office_location_id""home_location_id""employee_id""account_id""
           cost_center_id" */
 IF
   /* %ParentPK(" OR",UPDATE) */
   UPDATE(employee_id) OR
   UPDATE(account_id) OR
   UPDATE(office_location_id) OR
   UPDATE(home_location_id) OR
   UPDATE(cost_center_id)
 BEGIN
   IF EXISTS (
     SELECT * FROM deleted,schedule
     WHERE
       /*  %JoinFKPK(schedule,deleted," = "," AND") */
       schedule.employee_id = deleted.employee_id AND
       schedule.account_id = deleted.account_id AND
       schedule.office_location_id = deleted.office_location_id AND
       schedule.home_location_id = deleted.home_location_id AND
       schedule.cost_center_id = deleted.cost_center_id
   )
```

```
  BEGIN
    SELECT @errno  = 30005,
          @errmsg = 'Cannot update employee because schedule exists.'
    GOTO error
  END
 END

 /* erwin Builtin Trigger */
 /* employee  break_time on parent update no action */
 /*            ERWIN_RELATION:CHECKSUM="00000000",            PARENT_OWNER="",
PARENT_TABLE="employee"
   CHILD_OWNER="", CHILD_TABLE="break_time"
   P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
   FK_CONSTRAINT="R_21",
FK_COLUMNS="office_location_id""home_location_id""employee_id""account_id""
           cost_center_id" */
 IF
   /* %ParentPK(" OR",UPDATE) */
   UPDATE(employee_id) OR
   UPDATE(account_id) OR
   UPDATE(office_location_id) OR
   UPDATE(home_location_id) OR
   UPDATE(cost_center_id)
 BEGIN
   IF EXISTS (
     SELECT * FROM deleted,break_time
     WHERE
       /*  %JoinFKPK(break_time,deleted," = "," AND") */
       break_time.employee_id = deleted.employee_id AND
       break_time.account_id = deleted.account_id AND
       break_time.office_location_id = deleted.office_location_id AND
       break_time.home_location_id = deleted.home_location_id AND
       break_time.cost_center_id = deleted.cost_center_id
   )
   BEGIN
     SELECT @errno  = 30005,
          @errmsg = 'Cannot update employee because break_time exists.'
     GOTO error
```

```
  END
 END

 /* erwin Builtin Trigger */
 /* employee  assigned_device on parent update no action */
 /*            ERWIN_RELATION:CHECKSUM="00000000",            PARENT_OWNER="",
PARENT_TABLE="employee"
   CHILD_OWNER="", CHILD_TABLE="assigned_device"
   P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
   FK_CONSTRAINT="R_12",
FK_COLUMNS="office_location_id""home_location_id""employee_id""account_id""
           cost_center_id" */
 IF
   /* %ParentPK(" OR",UPDATE) */
   UPDATE(employee_id) OR
   UPDATE(account_id) OR
   UPDATE(office_location_id) OR
   UPDATE(home_location_id) OR
   UPDATE(cost_center_id)
 BEGIN
   IF EXISTS (
     SELECT * FROM deleted,assigned_device
     WHERE
       /*  %JoinFKPK(assigned_device,deleted," = "," AND") */
       assigned_device.employee_id = deleted.employee_id AND
       assigned_device.account_id = deleted.account_id AND
       assigned_device.office_location_id = deleted.office_location_id AND
       assigned_device.home_location_id = deleted.home_location_id AND
       assigned_device.cost_center_id = deleted.cost_center_id
   )
   BEGIN
     SELECT @errno  = 30005,
           @errmsg = 'Cannot update employee because assigned_device exists.'
     GOTO error
   END
 END
```

```
/* erwin Builtin Trigger */
  /* cost_center  employee on child update no action */
  /*            ERWIN_RELATION:CHECKSUM="00000000",            PARENT_OWNER="",
PARENT_TABLE="cost_center"
    CHILD_OWNER="", CHILD_TABLE="employee"
    P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
    FK_CONSTRAINT="R_35", FK_COLUMNS="cost_center_id" */
  IF
    /* %ChildFK(" OR",UPDATE) */
    UPDATE(cost_center_id)
  BEGIN
    SELECT @nullcnt = 0
    SELECT @validcnt = count(*)
      FROM inserted,cost_center
        WHERE
          /* %JoinFKPK(inserted,cost_center) */
          inserted.cost_center_id = cost_center.cost_center_id
    /* %NotnullFK(inserted," IS NULL","select @nullcnt = count(*) from inserted where"," AND") */

    IF @validcnt + @nullcnt != @numrows
    BEGIN
      SELECT @errno  = 30007,
            @errmsg = 'Cannot update employee because cost_center does not exist.'
      GOTO error
    END
  END

  /* erwin Builtin Trigger */
  /* account  employee on child update no action */
  /*            ERWIN_RELATION:CHECKSUM="00000000",            PARENT_OWNER="",
PARENT_TABLE="account"
    CHILD_OWNER="", CHILD_TABLE="employee"
    P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
    FK_CONSTRAINT="R_30", FK_COLUMNS="account_id" */
  IF
    /* %ChildFK(" OR",UPDATE) */
    UPDATE(account_id)
```

```
 BEGIN
  SELECT @nullcnt = 0
  SELECT @validcnt = count(*)
   FROM inserted,account
    WHERE
     /* %JoinFKPK(inserted,account) */
     inserted.account_id = account.account_id
  /* %NotnullFK(inserted," IS NULL","select @nullcnt = count(*) from inserted where","
AND") */

  IF @validcnt + @nullcnt != @numrows
  BEGIN
   SELECT @errno  = 30007,
        @errmsg = 'Cannot update employee because account does not exist.'
   GOTO error
  END
 END

 /* erwin Builtin Trigger */
 /* home_location  employee on child update no action */
 /*           ERWIN_RELATION:CHECKSUM="00000000",           PARENT_OWNER="",
PARENT_TABLE="home_location"
  CHILD_OWNER="", CHILD_TABLE="employee"
  P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
  FK_CONSTRAINT="R_20", FK_COLUMNS="home_location_id" */
 IF
  /* %ChildFK(" OR",UPDATE) */
  UPDATE(home_location_id)
 BEGIN
  SELECT @nullcnt = 0
  SELECT @validcnt = count(*)
   FROM inserted,home_location
    WHERE
     /* %JoinFKPK(inserted,home_location) */
     inserted.home_location_id = home_location.home_location_id
  /* %NotnullFK(inserted," IS NULL","select @nullcnt = count(*) from inserted where","
AND") */
```

```
  IF @validcnt + @nullcnt != @numrows
  BEGIN
    SELECT @errno  = 30007,
          @errmsg = 'Cannot update employee because home_location does not exist.'
    GOTO error
  END
 END

 /* erwin Builtin Trigger */
 /* office_location  employee on child update no action */
 /*            ERWIN_RELATION:CHECKSUM="00000000",            PARENT_OWNER="",
PARENT_TABLE="office_location"
   CHILD_OWNER="", CHILD_TABLE="employee"
   P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
   FK_CONSTRAINT="R_17", FK_COLUMNS="office_location_id" */
 IF
   /* %ChildFK(" OR",UPDATE) */
   UPDATE(office_location_id)
 BEGIN
   SELECT @nullcnt = 0
   SELECT @validcnt = count(*)
     FROM inserted,office_location
       WHERE
         /* %JoinFKPK(inserted,office_location) */
         inserted.office_location_id = office_location.office_location_id
   /* %NotnullFK(inserted," IS NULL","select @nullcnt = count(*) from inserted where","
AND") */

   IF @validcnt + @nullcnt != @numrows
   BEGIN
     SELECT @errno  = 30007,
           @errmsg = 'Cannot update employee because office_location does not exist.'
     GOTO error
   END
 END

 /* erwin Builtin Trigger */
 /* home_location  employee on child update no action */
```

```
 /*          ERWIN_RELATION:CHECKSUM="00000000",          PARENT_OWNER="",
PARENT_TABLE="home_location"
   CHILD_OWNER="", CHILD_TABLE="employee"
   P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
   FK_CONSTRAINT="R_10", FK_COLUMNS="home_location_id" */
 IF
   /* %ChildFK(" OR",UPDATE) */
   UPDATE(home_location_id)
 BEGIN
   SELECT @nullcnt = 0
   SELECT @validcnt = count(*)
    FROM inserted,home_location
      WHERE
       /* %JoinFKPK(inserted,home_location) */
       inserted.home_location_id = home_location.home_location_id
   /* %NotnullFK(inserted," IS NULL","select @nullcnt = count(*) from inserted where","
AND") */

   IF @validcnt + @nullcnt != @numrows
   BEGIN
    SELECT @errno  = 30007,
          @errmsg = 'Cannot update employee because home_location does not exist.'
    GOTO error
   END
 END

 /* erwin Builtin Trigger */
 /* office_location  employee on child update no action */
 /*          ERWIN_RELATION:CHECKSUM="00000000",          PARENT_OWNER="",
PARENT_TABLE="office_location"
   CHILD_OWNER="", CHILD_TABLE="employee"
   P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
   FK_CONSTRAINT="R_9", FK_COLUMNS="office_location_id" */
 IF
   /* %ChildFK(" OR",UPDATE) */
   UPDATE(office_location_id)
 BEGIN
   SELECT @nullcnt = 0
```

```
   SELECT @validcnt = count(*)
     FROM inserted,office_location
       WHERE
         /* %JoinFKPK(inserted,office_location) */
         inserted.office_location_id = office_location.office_location_id
   /* %NotnullFK(inserted," IS NULL","select @nullcnt = count(*) from inserted where","
AND") */

   IF @validcnt + @nullcnt != @numrows
   BEGIN
     SELECT @errno  = 30007,
           @errmsg = 'Cannot update employee because office_location does not exist.'
     GOTO error
   END
 END

  /* erwin Builtin Trigger */
  RETURN
error:
   RAISERROR (@errmsg, -- Message text.
            @severity, -- Severity (0~25).
            @state) -- State (0~255).
    rollback transaction
END
go

CREATE TRIGGER tD_event ON event FOR DELETE AS
/* erwin Builtin Trigger */
/* DELETE trigger on event */
BEGIN
  DECLARE  @errno   int,
           @severity int,
           @state    int,
           @errmsg  varchar(255)
    /* erwin Builtin Trigger */
    /* home_location  event on child delete no action */
    /*           ERWIN_RELATION:CHECKSUM="0002bf8e",           PARENT_OWNER="",
PARENT_TABLE="home_location"
```

```
  CHILD_OWNER="", CHILD_TABLE="event"
  P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
  FK_CONSTRAINT="R_24", FK_COLUMNS="home_location_id" */
  IF EXISTS (SELECT * FROM deleted,home_location
   WHERE
     /* %JoinFKPK(deleted,home_location," = "," AND") */
     deleted.home_location_id = home_location.home_location_id AND
     NOT EXISTS (
       SELECT * FROM event
       WHERE
         /* %JoinFKPK(event,home_location," = "," AND") */
         event.home_location_id = home_location.home_location_id
     )
  )
  BEGIN
   SELECT @errno  = 30010,
        @errmsg = 'Cannot delete last event because home_location exists.'
   GOTO error
  END

  /* erwin Builtin Trigger */
  /* office_location  event on child delete no action */
  /*           ERWIN_RELATION:CHECKSUM="00000000",           PARENT_OWNER="",
PARENT_TABLE="office_location"
  CHILD_OWNER="", CHILD_TABLE="event"
  P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
  FK_CONSTRAINT="R_23", FK_COLUMNS="office_location_id" */
  IF EXISTS (SELECT * FROM deleted,office_location
   WHERE
     /* %JoinFKPK(deleted,office_location," = "," AND") */
     deleted.office_location_id = office_location.office_location_id AND
     NOT EXISTS (
       SELECT * FROM event
       WHERE
         /* %JoinFKPK(event,office_location," = "," AND") */
         event.office_location_id = office_location.office_location_id
     )
  )
```

```
   BEGIN
    SELECT @errno  = 30010,
          @errmsg = 'Cannot delete last event because office_location exists.'
    GOTO error
   END

   /* erwin Builtin Trigger */
   RETURN
error:
  RAISERROR (@errmsg, -- Message text.
           @severity, -- Severity (0~25).
           @state) -- State (0~255).
   rollback transaction
END
go

CREATE TRIGGER tU_event ON event FOR UPDATE AS
/* erwin Builtin Trigger */
/* UPDATE trigger on event */
BEGIN
 DECLARE  @numrows int,
         @nullcnt int,
         @validcnt int,
         @insevent_id uniqueidentifier,
         @insoffice_location_id uniqueidentifier,
         @inshome_location_id uniqueidentifier,
         @errno   int,
         @severity int,
         @state    int,
         @errmsg  varchar(255)

 SELECT @numrows = @@rowcount
 /* erwin Builtin Trigger */
 /* home_location  event on child update no action */
 /*            ERWIN_RELATION:CHECKSUM="0002eb46",            PARENT_OWNER="",
PARENT_TABLE="home_location"
   CHILD_OWNER="", CHILD_TABLE="event"
   P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
```

```
  FK_CONSTRAINT="R_24", FK_COLUMNS="home_location_id" */
 IF
  /* %ChildFK(" OR",UPDATE) */
  UPDATE(home_location_id)
 BEGIN
  SELECT @nullcnt = 0
  SELECT @validcnt = count(*)
   FROM inserted,home_location
    WHERE
     /* %JoinFKPK(inserted,home_location) */
     inserted.home_location_id = home_location.home_location_id
  /* %NotnullFK(inserted," IS NULL","select @nullcnt = count(*) from inserted where","
AND") */

  IF @validcnt + @nullcnt != @numrows
  BEGIN
   SELECT @errno  = 30007,
        @errmsg = 'Cannot update event because home_location does not exist.'
   GOTO error
  END
 END

 /* erwin Builtin Trigger */
 /* office_location  event on child update no action */
 /*            ERWIN_RELATION:CHECKSUM="00000000",            PARENT_OWNER="",
PARENT_TABLE="office_location"
  CHILD_OWNER="", CHILD_TABLE="event"
  P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
  FK_CONSTRAINT="R_23", FK_COLUMNS="office_location_id" */
 IF
  /* %ChildFK(" OR",UPDATE) */
  UPDATE(office_location_id)
 BEGIN
  SELECT @nullcnt = 0
  SELECT @validcnt = count(*)
   FROM inserted,office_location
    WHERE
     /* %JoinFKPK(inserted,office_location) */
```

```
        inserted.office_location_id = office_location.office_location_id
   /* %NotnullFK(inserted," IS NULL","select @nullcnt = count(*) from inserted where","
AND") */

   IF @validcnt + @nullcnt != @numrows
   BEGIN
    SELECT @errno  = 30007,
          @errmsg = 'Cannot update event because office_location does not exist.'
    GOTO error
   END
  END

  /* erwin Builtin Trigger */
  RETURN
error:
   RAISERROR (@errmsg, -- Message text.
           @severity, -- Severity (0~25).
           @state) -- State (0~255).
   rollback transaction
END
go

CREATE TRIGGER tD_form ON form FOR DELETE AS
/* erwin Builtin Trigger */
/* DELETE trigger on form */
BEGIN
  DECLARE  @errno   int,
         @severity int,
         @state    int,
         @errmsg  varchar(255)
   /* erwin Builtin Trigger */
   /* form  user_role on parent delete no action */
   /*            ERWIN_RELATION:CHECKSUM="00010fa5",            PARENT_OWNER="",
PARENT_TABLE="form"
   CHILD_OWNER="", CHILD_TABLE="user_role"
   P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
   FK_CONSTRAINT="R_27", FK_COLUMNS="form_id" */
   IF EXISTS (
```

```
    SELECT * FROM deleted,user_role
    WHERE
      /*  %JoinFKPK(user_role,deleted," = "," AND") */
      user_role.form_id = deleted.form_id
  )
  BEGIN
    SELECT @errno  = 30001,
         @errmsg = 'Cannot delete form because user_role exists.'
    GOTO error
  END

  /* erwin Builtin Trigger */
  RETURN
error:
  RAISERROR (@errmsg, -- Message text.
          @severity, -- Severity (0~25).
          @state) -- State (0~255).
  rollback transaction
END
go

CREATE TRIGGER tU_form ON form FOR UPDATE AS
/* erwin Builtin Trigger */
/* UPDATE trigger on form */
BEGIN
  DECLARE  @numrows int,
         @nullcnt int,
         @validcnt int,
         @insform_id uniqueidentifier,
         @errno   int,
         @severity int,
         @state    int,
         @errmsg  varchar(255)

  SELECT @numrows = @@rowcount
  /* erwin Builtin Trigger */
  /* form  user_role on parent update no action */
```

```
 /*            ERWIN_RELATION:CHECKSUM="0001291f",            PARENT_OWNER="",
PARENT_TABLE="form"
   CHILD_OWNER="", CHILD_TABLE="user_role"
   P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
   FK_CONSTRAINT="R_27", FK_COLUMNS="form_id" */
 IF
   /* %ParentPK(" OR",UPDATE) */
   UPDATE(form_id)
 BEGIN
   IF EXISTS (
     SELECT * FROM deleted,user_role
     WHERE
       /*  %JoinFKPK(user_role,deleted," = "," AND") */
       user_role.form_id = deleted.form_id
   )
   BEGIN
     SELECT @errno  = 30005,
           @errmsg = 'Cannot update form because user_role exists.'
     GOTO error
   END
 END

 /* erwin Builtin Trigger */
 RETURN
error:
  RAISERROR (@errmsg, -- Message text.
           @severity, -- Severity (0~25).
           @state) -- State (0~255).
   rollback transaction
END
go


CREATE TRIGGER tD_home_location ON home_location FOR DELETE AS
/* erwin Builtin Trigger */
/* DELETE trigger on home_location */
BEGIN
 DECLARE  @errno   int,
```

```
        @severity int,
        @state    int,
        @errmsg  varchar(255)
    /* erwin Builtin Trigger */
    /* home_location  event on parent delete no action */
    /*           ERWIN_RELATION:CHECKSUM="00030d58",          PARENT_OWNER="",
PARENT_TABLE="home_location"
    CHILD_OWNER="", CHILD_TABLE="event"
    P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
    FK_CONSTRAINT="R_24", FK_COLUMNS="home_location_id" */
    IF EXISTS (
      SELECT * FROM deleted,event
      WHERE
        /*  %JoinFKPK(event,deleted," = "," AND") */
        event.home_location_id = deleted.home_location_id
    )
    BEGIN
      SELECT @errno  = 30001,
           @errmsg = 'Cannot delete home_location because event exists.'
      GOTO error
    END

    /* erwin Builtin Trigger */
    /* home_location  employee on parent delete no action */
    /*           ERWIN_RELATION:CHECKSUM="00000000",          PARENT_OWNER="",
PARENT_TABLE="home_location"
    CHILD_OWNER="", CHILD_TABLE="employee"
    P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
    FK_CONSTRAINT="R_20", FK_COLUMNS="home_location_id" */
    IF EXISTS (
      SELECT * FROM deleted,employee
      WHERE
        /*  %JoinFKPK(employee,deleted," = "," AND") */
        employee.home_location_id = deleted.home_location_id
    )
    BEGIN
      SELECT @errno  = 30001,
           @errmsg = 'Cannot delete home_location because employee exists.'
```

```
    GOTO error
   END

   /* erwin Builtin Trigger */
   /* home_location  employee on parent delete no action */
   /*           ERWIN_RELATION:CHECKSUM="00000000",          PARENT_OWNER="",
PARENT_TABLE="home_location"
   CHILD_OWNER="", CHILD_TABLE="employee"
   P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
   FK_CONSTRAINT="R_10", FK_COLUMNS="home_location_id" */
   IF EXISTS (
    SELECT * FROM deleted,employee
    WHERE
     /*  %JoinFKPK(employee,deleted," = "," AND") */
     employee.home_location_id = deleted.home_location_id
   )
   BEGIN
    SELECT @errno  = 30001,
         @errmsg = 'Cannot delete home_location because employee exists.'
    GOTO error
   END

   /* erwin Builtin Trigger */
   RETURN
error:
  RAISERROR (@errmsg, -- Message text.
          @severity, -- Severity (0~25).
          @state) -- State (0~255).
   rollback transaction
END
go
```

```
CREATE TRIGGER tU_home_location ON home_location FOR UPDATE AS
/* erwin Builtin Trigger */
/* UPDATE trigger on home_location */
BEGIN
  DECLARE  @numrows int,
           @nullcnt int,
           @validcnt int,
           @inshome_location_id uniqueidentifier,
           @errno   int,
           @severity int,
           @state    int,
           @errmsg  varchar(255)

  SELECT @numrows = @@rowcount
  /* erwin Builtin Trigger */
  /* home_location  event on parent update no action */
  /*              ERWIN_RELATION:CHECKSUM="000369b2",            PARENT_OWNER="",
PARENT_TABLE="home_location"
    CHILD_OWNER="", CHILD_TABLE="event"
    P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
    FK_CONSTRAINT="R_24", FK_COLUMNS="home_location_id" */
  IF
    /* %ParentPK(" OR",UPDATE) */
    UPDATE(home_location_id)
  BEGIN
    IF EXISTS (
      SELECT * FROM deleted,event
      WHERE
        /*  %JoinFKPK(event,deleted," = "," AND") */
        event.home_location_id = deleted.home_location_id
    )
    BEGIN
      SELECT @errno  = 30005,
             @errmsg = 'Cannot update home_location because event exists.'
      GOTO error
    END
  END
```

```
  /* erwin Builtin Trigger */
  /* home_location  employee on parent update no action */
  /*            ERWIN_RELATION:CHECKSUM="00000000",            PARENT_OWNER="",
PARENT_TABLE="home_location"
    CHILD_OWNER="", CHILD_TABLE="employee"
    P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
    FK_CONSTRAINT="R_20", FK_COLUMNS="home_location_id" */
  IF
    /* %ParentPK(" OR",UPDATE) */
    UPDATE(home_location_id)
  BEGIN
    IF EXISTS (
      SELECT * FROM deleted,employee
      WHERE
        /*  %JoinFKPK(employee,deleted," = "," AND") */
        employee.home_location_id = deleted.home_location_id
    )
    BEGIN
      SELECT @errno  = 30005,
            @errmsg = 'Cannot update home_location because employee exists.'
      GOTO error
    END
  END

  /* erwin Builtin Trigger */
  /* home_location  employee on parent update no action */
  /*            ERWIN_RELATION:CHECKSUM="00000000",            PARENT_OWNER="",
PARENT_TABLE="home_location"
    CHILD_OWNER="", CHILD_TABLE="employee"
    P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
    FK_CONSTRAINT="R_10", FK_COLUMNS="home_location_id" */
  IF
    /* %ParentPK(" OR",UPDATE) */
    UPDATE(home_location_id)
  BEGIN
    IF EXISTS (
      SELECT * FROM deleted,employee
      WHERE
```

```
      /*  %JoinFKPK(employee,deleted," = "," AND") */
      employee.home_location_id = deleted.home_location_id
   )
   BEGIN
    SELECT @errno  = 30005,
          @errmsg = 'Cannot update home_location because employee exists.'
    GOTO error
   END
  END

  /* erwin Builtin Trigger */
  RETURN
error:
   RAISERROR (@errmsg, -- Message text.
           @severity, -- Severity (0~25).
           @state) -- State (0~255).
   rollback transaction
END
go

CREATE TRIGGER tD_office_location ON office_location FOR DELETE AS
/* erwin Builtin Trigger */
/* DELETE trigger on office_location */
BEGIN
  DECLARE  @errno   int,
         @severity int,
         @state    int,
         @errmsg  varchar(255)
   /* erwin Builtin Trigger */
   /* office_location  event on parent delete no action */
   /*           ERWIN_RELATION:CHECKSUM="000318ee",           PARENT_OWNER="",
PARENT_TABLE="office_location"
   CHILD_OWNER="", CHILD_TABLE="event"
   P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
   FK_CONSTRAINT="R_23", FK_COLUMNS="office_location_id" */
   IF EXISTS (
    SELECT * FROM deleted,event
    WHERE
```

```
      /*  %JoinFKPK(event,deleted," = "," AND") */
      event.office_location_id = deleted.office_location_id
    )
    BEGIN
     SELECT @errno  = 30001,
          @errmsg = 'Cannot delete office_location because event exists.'
     GOTO error
    END

    /* erwin Builtin Trigger */
    /* office_location  employee on parent delete no action */
    /*           ERWIN_RELATION:CHECKSUM="00000000",           PARENT_OWNER="",
PARENT_TABLE="office_location"
    CHILD_OWNER="", CHILD_TABLE="employee"
    P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
    FK_CONSTRAINT="R_17", FK_COLUMNS="office_location_id" */
    IF EXISTS (
     SELECT * FROM deleted,employee
     WHERE
      /*  %JoinFKPK(employee,deleted," = "," AND") */
      employee.office_location_id = deleted.office_location_id
    )
    BEGIN
     SELECT @errno  = 30001,
          @errmsg = 'Cannot delete office_location because employee exists.'
     GOTO error
    END

    /* erwin Builtin Trigger */
    /* office_location  employee on parent delete no action */
    /*           ERWIN_RELATION:CHECKSUM="00000000",           PARENT_OWNER="",
PARENT_TABLE="office_location"
    CHILD_OWNER="", CHILD_TABLE="employee"
    P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
    FK_CONSTRAINT="R_9", FK_COLUMNS="office_location_id" */
    IF EXISTS (
     SELECT * FROM deleted,employee
     WHERE
```

```
      /*  %JoinFKPK(employee,deleted," = "," AND") */
      employee.office_location_id = deleted.office_location_id
   )
   BEGIN
    SELECT @errno  = 30001,
          @errmsg = 'Cannot delete office_location because employee exists.'
    GOTO error
   END

   /* erwin Builtin Trigger */
   RETURN
error:
  RAISERROR (@errmsg, -- Message text.
           @severity, -- Severity (0~25).
           @state) -- State (0~255).
   rollback transaction
END
go

CREATE TRIGGER tU_office_location ON office_location FOR UPDATE AS
/* erwin Builtin Trigger */
/* UPDATE trigger on office_location */
BEGIN
 DECLARE  @numrows int,
         @nullcnt int,
         @validcnt int,
         @insoffice_location_id uniqueidentifier,
         @errno   int,
         @severity int,
         @state    int,
         @errmsg  varchar(255)

 SELECT @numrows = @@rowcount
 /* erwin Builtin Trigger */
 /* office_location  event on parent update no action */
 /*            ERWIN_RELATION:CHECKSUM="00036d5a",            PARENT_OWNER="",
PARENT_TABLE="office_location"
   CHILD_OWNER="", CHILD_TABLE="event"
```

```
  P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
  FK_CONSTRAINT="R_23", FK_COLUMNS="office_location_id" */
 IF
  /* %ParentPK(" OR",UPDATE) */
  UPDATE(office_location_id)
 BEGIN
  IF EXISTS (
   SELECT * FROM deleted,event
   WHERE
    /*  %JoinFKPK(event,deleted," = "," AND") */
    event.office_location_id = deleted.office_location_id
  )
  BEGIN
   SELECT @errno  = 30005,
        @errmsg = 'Cannot update office_location because event exists.'
   GOTO error
  END
 END

 /* erwin Builtin Trigger */
 /* office_location  employee on parent update no action */
 /*            ERWIN_RELATION:CHECKSUM="00000000",            PARENT_OWNER="",
PARENT_TABLE="office_location"
  CHILD_OWNER="", CHILD_TABLE="employee"
  P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
  FK_CONSTRAINT="R_17", FK_COLUMNS="office_location_id" */
 IF
  /* %ParentPK(" OR",UPDATE) */
  UPDATE(office_location_id)
 BEGIN
  IF EXISTS (
   SELECT * FROM deleted,employee
   WHERE
    /*  %JoinFKPK(employee,deleted," = "," AND") */
    employee.office_location_id = deleted.office_location_id
  )
  BEGIN
   SELECT @errno  = 30005,
```

```
        @errmsg = 'Cannot update office_location because employee exists.'
    GOTO error
   END
  END

  /* erwin Builtin Trigger */
  /* office_location  employee on parent update no action */
  /*            ERWIN_RELATION:CHECKSUM="00000000",            PARENT_OWNER="",
PARENT_TABLE="office_location"
    CHILD_OWNER="", CHILD_TABLE="employee"
    P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
    FK_CONSTRAINT="R_9", FK_COLUMNS="office_location_id" */
  IF
    /* %ParentPK(" OR",UPDATE) */
    UPDATE(office_location_id)
  BEGIN
    IF EXISTS (
      SELECT * FROM deleted,employee
      WHERE
        /*  %JoinFKPK(employee,deleted," = "," AND") */
        employee.office_location_id = deleted.office_location_id
    )
    BEGIN
      SELECT @errno  = 30005,
        @errmsg = 'Cannot update office_location because employee exists.'
      GOTO error
    END
  END

  /* erwin Builtin Trigger */
  RETURN
error:
   RAISERROR (@errmsg, -- Message text.
          @severity, -- Severity (0~25).
          @state) -- State (0~255).
    rollback transaction
END
go
```

```
CREATE TRIGGER tD_role ON role FOR DELETE AS
/* erwin Builtin Trigger */
/* DELETE trigger on role */
BEGIN
  DECLARE  @errno   int,
           @severity int,
           @state    int,
           @errmsg  varchar(255)
    /* erwin Builtin Trigger */
    /* role  user_role on parent delete no action */
    /*            ERWIN_RELATION:CHECKSUM="000107f7",            PARENT_OWNER="",
PARENT_TABLE="role"
    CHILD_OWNER="", CHILD_TABLE="user_role"
    P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
    FK_CONSTRAINT="R_26", FK_COLUMNS="role_id" */
    IF EXISTS (
      SELECT * FROM deleted,user_role
      WHERE
        /*  %JoinFKPK(user_role,deleted," = "," AND") */
        user_role.role_id = deleted.role_id
    )
    BEGIN
      SELECT @errno  = 30001,
             @errmsg = 'Cannot delete role because user_role exists.'
      GOTO error
    END

    /* erwin Builtin Trigger */
    RETURN
error:
   RAISERROR (@errmsg, -- Message text.
              @severity, -- Severity (0~25).
              @state) -- State (0~255).
    rollback transaction
END
go
```

```
CREATE TRIGGER tU_role ON role FOR UPDATE AS
/* erwin Builtin Trigger */
/* UPDATE trigger on role */
BEGIN
  DECLARE  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insrole_id uniqueidentifier,
           @errno   int,
           @severity int,
           @state    int,
           @errmsg  varchar(255)

  SELECT @numrows = @@rowcount
  /* erwin Builtin Trigger */
  /* role  user_role on parent update no action */
  /*              ERWIN_RELATION:CHECKSUM="00012ba4",            PARENT_OWNER="",
PARENT_TABLE="role"
    CHILD_OWNER="", CHILD_TABLE="user_role"
    P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
    FK_CONSTRAINT="R_26", FK_COLUMNS="role_id" */
  IF
    /* %ParentPK(" OR",UPDATE) */
    UPDATE(role_id)
  BEGIN
    IF EXISTS (
      SELECT * FROM deleted,user_role
      WHERE
        /*  %JoinFKPK(user_role,deleted," = "," AND") */
        user_role.role_id = deleted.role_id
    )
    BEGIN
      SELECT @errno  = 30005,
            @errmsg = 'Cannot update role because user_role exists.'
      GOTO error
    END
  END
```

```
 /* erwin Builtin Trigger */
 RETURN
error:
  RAISERROR (@errmsg, -- Message text.
          @severity, -- Severity (0~25).
          @state) -- State (0~255).
   rollback transaction
END
go

CREATE TRIGGER tD_schedule ON schedule FOR DELETE AS
/* erwin Builtin Trigger */
/* DELETE trigger on schedule */
BEGIN
 DECLARE  @errno   int,
         @severity int,
         @state    int,
         @errmsg  varchar(255)
   /* erwin Builtin Trigger */
   /* employee  schedule on child delete no action */
   /*           ERWIN_RELATION:CHECKSUM="00023ef5",           PARENT_OWNER="",
PARENT_TABLE="employee"
   CHILD_OWNER="", CHILD_TABLE="schedule"
   P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
   FK_CONSTRAINT="R_34",
FK_COLUMNS="office_location_id""home_location_id""employee_id""account_id""
           cost_center_id" */
   IF EXISTS (SELECT * FROM deleted,employee
     WHERE
       /* %JoinFKPK(deleted,employee," = "," AND") */
       deleted.employee_id = employee.employee_id AND
       deleted.account_id = employee.account_id AND
       deleted.office_location_id = employee.office_location_id AND
       deleted.home_location_id = employee.home_location_id AND
       deleted.cost_center_id = employee.cost_center_id AND
       NOT EXISTS (
         SELECT * FROM schedule
         WHERE
```

```
        /* %JoinFKPK(schedule,employee," = "," AND") */
        schedule.employee_id = employee.employee_id AND
        schedule.account_id = employee.account_id AND
        schedule.office_location_id = employee.office_location_id AND
        schedule.home_location_id = employee.home_location_id AND
        schedule.cost_center_id = employee.cost_center_id
      )
  )
  BEGIN
    SELECT @errno  = 30010,
        @errmsg = 'Cannot delete last schedule because employee exists.'
    GOTO error
  END

  /* erwin Builtin Trigger */
  RETURN
error:
  RAISERROR (@errmsg, -- Message text.
          @severity, -- Severity (0~25).
          @state) -- State (0~255).
  rollback transaction
END
go

CREATE TRIGGER tU_schedule ON schedule FOR UPDATE AS
/* erwin Builtin Trigger */
/* UPDATE trigger on schedule */
BEGIN
  DECLARE  @numrows int,
        @nullcnt int,
        @validcnt int,
        @insschedule_id char(18),
        @insdate date,
        @insemployee_id uniqueidentifier,
        @insoffice_location_id uniqueidentifier,
        @inshome_location_id uniqueidentifier,
        @insaccount_id uniqueidentifier,
        @inscost_center_id uniqueidentifier,
```

```
        @errno   int,
        @severity int,
        @state    int,
        @errmsg  varchar(255)

  SELECT @numrows = @@rowcount
  /* erwin Builtin Trigger */
  /* employee  schedule on child update no action */
  /*            ERWIN_RELATION:CHECKSUM="000223f0",           PARENT_OWNER="",
PARENT_TABLE="employee"
    CHILD_OWNER="", CHILD_TABLE="schedule"
    P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
    FK_CONSTRAINT="R_34",
FK_COLUMNS="home_location_id""office_location_id""employee_id""account_id""
           cost_center_id" */
  IF
    /* %ChildFK(" OR",UPDATE) */
    UPDATE(employee_id) OR
    UPDATE(account_id) OR
    UPDATE(office_location_id) OR
    UPDATE(home_location_id) OR
    UPDATE(cost_center_id)
  BEGIN
    SELECT @nullcnt = 0
    SELECT @validcnt = count(*)
     FROM inserted,employee
       WHERE
        /* %JoinFKPK(inserted,employee) */
        inserted.employee_id = employee.employee_id and
        inserted.account_id = employee.account_id and
        inserted.office_location_id = employee.office_location_id and
        inserted.home_location_id = employee.home_location_id and
        inserted.cost_center_id = employee.cost_center_id
    /* %NotnullFK(inserted," IS NULL","select @nullcnt = count(*) from inserted where","
AND") */

    IF @validcnt + @nullcnt != @numrows
    BEGIN
```

```
    SELECT @errno  = 30007,
          @errmsg = 'Cannot update schedule because employee does not exist.'
    GOTO error
   END
  END

  /* erwin Builtin Trigger */
  RETURN
error:
   RAISERROR (@errmsg, -- Message text.
           @severity, -- Severity (0~25).
           @state) -- State (0~255).
   rollback transaction
END
go

CREATE TRIGGER tD_sys_user ON sys_user FOR DELETE AS
/* erwin Builtin Trigger */
/* DELETE trigger on sys_user */
BEGIN
  DECLARE  @errno   int,
         @severity int,
         @state    int,
         @errmsg  varchar(255)
   /* erwin Builtin Trigger */
   /* sys_user  user_role on parent delete no action */
   /*           ERWIN_RELATION:CHECKSUM="00010a74",           PARENT_OWNER="",
PARENT_TABLE="sys_user"
   CHILD_OWNER="", CHILD_TABLE="user_role"
   P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
   FK_CONSTRAINT="R_25", FK_COLUMNS="user_id" */
   IF EXISTS (
    SELECT * FROM deleted,user_role
    WHERE
      /*  %JoinFKPK(user_role,deleted," = "," AND") */
      user_role.user_id = deleted.user_id
   )
```

```
   BEGIN
    SELECT @errno  = 30001,
          @errmsg = 'Cannot delete sys_user because user_role exists.'
    GOTO error
   END

   /* erwin Builtin Trigger */
   RETURN
error:
  RAISERROR (@errmsg, -- Message text.
           @severity, -- Severity (0~25).
           @state) -- State (0~255).
   rollback transaction
END
go

CREATE TRIGGER tU_sys_user ON sys_user FOR UPDATE AS
/* erwin Builtin Trigger */
/* UPDATE trigger on sys_user */
BEGIN
 DECLARE  @numrows int,
         @nullcnt int,
         @validcnt int,
         @insuser_id uniqueidentifier,
         @errno   int,
         @severity int,
         @state    int,
         @errmsg  varchar(255)

 SELECT @numrows = @@rowcount
 /* erwin Builtin Trigger */
 /* sys_user  user_role on parent update no action */
 /*            ERWIN_RELATION:CHECKSUM="000123b3",            PARENT_OWNER="",
PARENT_TABLE="sys_user"
   CHILD_OWNER="", CHILD_TABLE="user_role"
   P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
   FK_CONSTRAINT="R_25", FK_COLUMNS="user_id" */
```

```
 IF
  /* %ParentPK(" OR",UPDATE) */
  UPDATE(user_id)
 BEGIN
  IF EXISTS (
   SELECT * FROM deleted,user_role
   WHERE
    /*  %JoinFKPK(user_role,deleted," = "," AND") */
    user_role.user_id = deleted.user_id
  )
  BEGIN
   SELECT @errno  = 30005,
        @errmsg = 'Cannot update sys_user because user_role exists.'
   GOTO error
  END
 END

 /* erwin Builtin Trigger */
 RETURN
error:
  RAISERROR (@errmsg, -- Message text.
          @severity, -- Severity (0~25).
          @state) -- State (0~255).
  rollback transaction
END
go

CREATE TRIGGER tD_user_role ON user_role FOR DELETE AS
/* erwin Builtin Trigger */
/* DELETE trigger on user_role */
BEGIN
 DECLARE  @errno   int,
        @severity int,
        @state    int,
        @errmsg  varchar(255)
  /* erwin Builtin Trigger */
  /* form  user_role on child delete no action */
```

```
  /*          ERWIN_RELATION:CHECKSUM="00038cf6",          PARENT_OWNER="",
PARENT_TABLE="form"
  CHILD_OWNER="", CHILD_TABLE="user_role"
  P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
  FK_CONSTRAINT="R_27", FK_COLUMNS="form_id" */
  IF EXISTS (SELECT * FROM deleted,form
   WHERE
    /* %JoinFKPK(deleted,form," = "," AND") */
    deleted.form_id = form.form_id AND
    NOT EXISTS (
     SELECT * FROM user_role
     WHERE
      /* %JoinFKPK(user_role,form," = "," AND") */
      user_role.form_id = form.form_id
    )
  )
  BEGIN
   SELECT @errno  = 30010,
        @errmsg = 'Cannot delete last user_role because form exists.'
   GOTO error
  END

  /* erwin Builtin Trigger */
  /* role  user_role on child delete no action */
  /*          ERWIN_RELATION:CHECKSUM="00000000",          PARENT_OWNER="",
PARENT_TABLE="role"
  CHILD_OWNER="", CHILD_TABLE="user_role"
  P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
  FK_CONSTRAINT="R_26", FK_COLUMNS="role_id" */
  IF EXISTS (SELECT * FROM deleted,role
   WHERE
    /* %JoinFKPK(deleted,role," = "," AND") */
    deleted.role_id = role.role_id AND
    NOT EXISTS (
     SELECT * FROM user_role
     WHERE
      /* %JoinFKPK(user_role,role," = "," AND") */
      user_role.role_id = role.role_id
```

```
      )
   )
   BEGIN
    SELECT @errno  = 30010,
         @errmsg = 'Cannot delete last user_role because role exists.'
    GOTO error
   END

   /* erwin Builtin Trigger */
   /* sys_user  user_role on child delete no action */
   /*           ERWIN_RELATION:CHECKSUM="00000000",           PARENT_OWNER="",
PARENT_TABLE="sys_user"
   CHILD_OWNER="", CHILD_TABLE="user_role"
   P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
   FK_CONSTRAINT="R_25", FK_COLUMNS="user_id" */
   IF EXISTS (SELECT * FROM deleted,sys_user
    WHERE
     /* %JoinFKPK(deleted,sys_user," = "," AND") */
     deleted.user_id = sys_user.user_id AND
     NOT EXISTS (
      SELECT * FROM user_role
      WHERE
       /* %JoinFKPK(user_role,sys_user," = "," AND") */
       user_role.user_id = sys_user.user_id
     )
   )
   BEGIN
    SELECT @errno  = 30010,
         @errmsg = 'Cannot delete last user_role because sys_user exists.'
    GOTO error
   END

   /* erwin Builtin Trigger */
   RETURN
error:
  RAISERROR (@errmsg, -- Message text.
          @severity, -- Severity (0~25).
          @state) -- State (0~255).
```

```
   rollback transaction
END
go

CREATE TRIGGER tU_user_role ON user_role FOR UPDATE AS
/* erwin Builtin Trigger */
/* UPDATE trigger on user_role */
BEGIN
  DECLARE  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insuser_id uniqueidentifier,
           @insrole_id uniqueidentifier,
           @insform_id uniqueidentifier,
           @errno   int,
           @severity int,
           @state    int,
           @errmsg  varchar(255)

  SELECT @numrows = @@rowcount
  /* erwin Builtin Trigger */
  /* form  user_role on child update no action */
  /*              ERWIN_RELATION:CHECKSUM="0003f8c5",             PARENT_OWNER="",
PARENT_TABLE="form"
    CHILD_OWNER="", CHILD_TABLE="user_role"
    P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
    FK_CONSTRAINT="R_27", FK_COLUMNS="form_id" */
  IF
    /* %ChildFK(" OR",UPDATE) */
    UPDATE(form_id)
  BEGIN
    SELECT @nullcnt = 0
    SELECT @validcnt = count(*)
      FROM inserted,form
        WHERE
          /* %JoinFKPK(inserted,form) */
          inserted.form_id = form.form_id
```

```
   /* %NotnullFK(inserted," IS NULL","select @nullcnt = count(*) from inserted where","
AND") */

   IF @validcnt + @nullcnt != @numrows
   BEGIN
     SELECT @errno  = 30007,
          @errmsg = 'Cannot update user_role because form does not exist.'
     GOTO error
   END
  END

  /* erwin Builtin Trigger */
  /* role  user_role on child update no action */
  /*            ERWIN_RELATION:CHECKSUM="00000000",           PARENT_OWNER="",
PARENT_TABLE="role"
   CHILD_OWNER="", CHILD_TABLE="user_role"
   P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
   FK_CONSTRAINT="R_26", FK_COLUMNS="role_id" */
  IF
   /* %ChildFK(" OR",UPDATE) */
   UPDATE(role_id)
  BEGIN
   SELECT @nullcnt = 0
   SELECT @validcnt = count(*)
    FROM inserted,role
      WHERE
       /* %JoinFKPK(inserted,role) */
       inserted.role_id = role.role_id
   /* %NotnullFK(inserted," IS NULL","select @nullcnt = count(*) from inserted where","
AND") */

   IF @validcnt + @nullcnt != @numrows
   BEGIN
     SELECT @errno  = 30007,
          @errmsg = 'Cannot update user_role because role does not exist.'
     GOTO error
   END
  END
```

```
  /* erwin Builtin Trigger */
  /* sys_user  user_role on child update no action */
  /*            ERWIN_RELATION:CHECKSUM="00000000",            PARENT_OWNER="",
PARENT_TABLE="sys_user"
    CHILD_OWNER="", CHILD_TABLE="user_role"
    P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
    FK_CONSTRAINT="R_25", FK_COLUMNS="user_id" */
  IF
    /* %ChildFK(" OR",UPDATE) */
    UPDATE(user_id)
  BEGIN
    SELECT @nullcnt = 0
    SELECT @validcnt = count(*)
      FROM inserted,sys_user
        WHERE
          /* %JoinFKPK(inserted,sys_user) */
          inserted.user_id = sys_user.user_id
    /* %NotnullFK(inserted," IS NULL","select @nullcnt = count(*) from inserted where","
AND") */

    IF @validcnt + @nullcnt != @numrows
    BEGIN
      SELECT @errno  = 30007,
            @errmsg = 'Cannot update user_role because sys_user does not exist.'
      GOTO error
    END
  END

  /* erwin Builtin Trigger */
  RETURN
error:
   RAISERROR (@errmsg, -- Message text.
             @severity, -- Severity (0~25).
             @state) -- State (0~255).
    rollback transaction
END
go
```

CONFIGURACIONES INCIALES DE LA BASE DE DATOS

Insert Into location(parent_location_id, location_type, location_code, description, latitude, longitude, created_by, updated_by)
values(NULL, 'departamento', 'SS', 'San Salvador', 13.68935, -89.18718 , 'admin', 'admin');

Insert Into location(parent_location_id, location_type, location_code, description, latitude, longitude, created_by, updated_by)
values('0e641013-434c-4a5b-b9c9-9d464add3661', 'municipio', 'SM', 'San Marcos', 13.66108, -89.18252 , 'admin', 'admin');

Insert Into form(form_code, description, state, created_by, updated_by) values('idioma', 'Idioma', 'A' , 'admin', 'admin');
Insert Into form(form_code, description, state, created_by, updated_by) values('tipo_provision', 'Tipo de Provision', 'A' , 'admin', 'admin');
Insert Into form(form_code, description, state, created_by, updated_by) values('evento', 'Evento', 'A', 'admin', 'admin');
Insert Into form(form_code, description, state, created_by, updated_by) values('perspectiva_evento', 'Perspectiva Evento', 'A', 'admin', 'admin');

Insert Into sys_catalog(form_code, catalog_type_code, language_code, value, label, parent_catalog_id, state, hint, sequence, created_by, updated_by)
values ('idioma', 'idioma_id', 'sp', 'Spanish', 'Spanish', null, 'A', null, 0, 'admin', 'admin');
Insert Into sys_catalog(form_code, catalog_type_code, language_code, value, label, parent_catalog_id, state, hint, sequence, created_by, updated_by)
values ('idioma', 'idioma_id', 'en', 'English', 'English', null, 'A', null, 1, 'admin', 'admin');

Insert Into sys_catalog(form_code, catalog_type_code, language_code, value, label, parent_catalog_id, state, hint, sequence, created_by, updated_by)
values ('evento', 'evento_id', 'sp', 'incidente', 'Incidente', null, 'A', null, 0, 'admin', 'admin');
Insert Into sys_catalog(form_code, catalog_type_code, language_code, value, label, parent_catalog_id, state, hint, sequence, created_by, updated_by)
values ('evento', 'evento_id', 'sp', 'gestion', 'Gestion', null, 'A', null, 1, 'admin', 'admin');

Insert Into sys_catalog(form_code, catalog_type_code, language_code, value, label, parent_catalog_id, state, hint, sequence, created_by, updated_by)
values ('tipo_provision', 'language_code', 'sp', 'cuenta', 'Por cuenta', null, 'A', null, 0, 'admin', 'admin');

Insert Into sys_catalog(form_code, catalog_type_code, language_code, value, label, parent_catalog_id, state, hint, sequence, created_by, updated_by)
values ('tipo_provision', 'language_code', 'sp', 'zona', 'Por zona', null, 'A', null, 1, 'admin', 'admin');
Insert Into sys_catalog(form_code, catalog_type_code, language_code, value, label, parent_catalog_id, state, hint, sequence, created_by, updated_by)
values ('tipo_provision', 'language_code', 'sp', 'disposicion', 'Por disposicion', null, 'A', null, 2, 'admin', 'admin');
Insert Into sys_catalog(form_code, catalog_type_code, language_code, value, label, parent_catalog_id, state, hint, sequence, created_by, updated_by)
values ('tipo_provision', 'language_code', 'sp', 'fecha', 'Por fecha', null, 'A', null, 3, 'admin', 'admin');
Insert Into sys_catalog(form_code, catalog_type_code, language_code, value, label, parent_catalog_id, state, hint, sequence, created_by, updated_by)
values ('tipo_provision', 'language_code', 'sp', 'empleado', 'Por empleado', null, 'A', null, 4, 'admin', 'admin');

Insert Into sys_catalog(form_code, catalog_type_code, language_code, value, label, parent_catalog_id, state, hint, sequence, created_by, updated_by)
values ('perspectiva_evento', 'perspectiva_id', 'sp', 'fecha', 'Por fecha', null, 'A', null, 0, 'admin', 'admin');
Insert Into sys_catalog(form_code, catalog_type_code, language_code, value, label, parent_catalog_id, state, hint, sequence, created_by, updated_by)
values ('perspectiva_evento', 'perspectiva_id', 'sp', 'lugar_movil', 'Por lugar movil', null, 'A', null, 1, 'admin', 'admin');
Insert Into sys_catalog(form_code, catalog_type_code, language_code, value, label, parent_catalog_id, state, hint, sequence, created_by, updated_by)
values ('perspectiva_evento', 'perspectiva_id', 'sp', 'tipo_evento', 'Por tipo evento', null, 'A', null, 2, 'admin', 'admin');
Insert Into sys_catalog(form_code, catalog_type_code, language_code, value, label, parent_catalog_id, state, hint, sequence, created_by, updated_by)
values ('perspectiva_evento', 'perspectiva_id', 'sp', 'forma_recepcion', 'Por forma de recepcion', null, 'A', null, 3, 'admin', 'admin');
Insert Into sys_catalog(form_code, catalog_type_code, language_code, value, label, parent_catalog_id, state, hint, sequence, created_by, updated_by)
values ('perspectiva_evento', 'perspectiva_id', 'sp', 'cuenta', 'Por cuenta', null, 'A', null, 4, 'admin', 'admin');
Insert Into sys_catalog(form_code, catalog_type_code, language_code, value, label, parent_catalog_id, state, hint, sequence, created_by, updated_by)

values ('perspectiva_evento', 'perspectiva_id', 'sp', 'clasificacion', 'Clasificacion (Sector y Categoría)', null, 'A', null, 5, 'admin', 'admin');

Insert Into sys_catalog(form_code, catalog_type_code, language_code, value, label, parent_catalog_id, state, hint, sequence, created_by, updated_by)

values ('perspectiva_evento', 'perspectiva_id', 'sp', 'motivo', 'Motivo', null, 'A', null, 6, 'admin', 'admin');

Insert Into sys_catalog(form_code, catalog_type_code, language_code, value, label, parent_catalog_id, state, hint, sequence, created_by, updated_by)

values ('perspectiva_evento', 'perspectiva_id', 'sp', 'zona', 'Por zona', null, 'A', null, 7, 'admin', 'admin');

Insert Into sys_catalog(form_code, catalog_type_code, language_code, value, label, parent_catalog_id, state, hint, sequence, created_by, updated_by)

values ('perspectiva_evento', 'perspectiva_id', 'sp', 'ubicacion_empleado', 'Por ubicacion del empleado', null, 'A', null, 8, 'admin', 'admin');

Insert Into sys_catalog(form_code, catalog_type_code, language_code, value, label, parent_catalog_id, state, hint, sequence, created_by, updated_by)

values ('perspectiva_evento', 'perspectiva_id', 'sp', 'empleado', 'Por empleado', null, 'A', null, 9, 'admin', 'admin');

Insert Into sys_catalog(form_code, catalog_type_code, language_code, value, label, parent_catalog_id, state, hint, sequence, created_by, updated_by)

values ('perspectiva_evento', 'perspectiva_id', 'sp', 'cargo', 'Por cargo', null, 'A', null, 10, 'admin', 'admin');

Insert Into sys_catalog(form_code, catalog_type_code, language_code, value, label, parent_catalog_id, state, hint, sequence, created_by, updated_by)

values ('perspectiva_evento', 'perspectiva_id', 'sp', 'tipo_disposicion', 'Por tipo de disposicion', null, 'A', null, 11, 'admin', 'admin');

Anexo 3: Código fuente de programa de integración de datos

Integración.cs

```
using System;
using System.Data.SqlClient;
using System.Collections.Generic;
using System.Collections;
using integracionAlCubo;

namespace Integracion
{
   class Integracion
   {
      static void Main(string[] args)
      {
         Console.WriteLine("Leyendo informacion de agentes");
         Console.WriteLine("Leyendo informacion de operacion");
         ArrayList list = new MockUpData.GetMockupEmployee();

         for (int i = 0; i < list.Count; i++)
         {
            IDictionary<string, string> row = (IDictionary<string, string>)list[i];
            //Paso 1: agregar y obtener id_location region
            String queryRegion = "select location_id from dbo.location where location_type='region' and
description='Centroamerica';";
            var readerRegion = DBConn.ReadDBIntegracion(queryRegion);
            string regionId = readerRegion;
            //Paso 2: agregar y obtener id_location pais
            String queryPais = "select location_id from dbo.location where location_type='pais' and
description='El Salvador';";
            var readerPais = DBConn.ReadDBIntegracion(queryPais);
            String paisId = readerPais;
            //Paso 3: buscar departamemto y si no existe agregar y obtener id_location departamento
            String queryDepto = "select location_id from dbo.location where location_type='departamento'
and description='" + row["resDepartamento"] +"';";
            var readerDepto = DBConn.ReadDBIntegracion(queryDepto);
            string deptoId = readerDepto;
            //Paso 4: buscar municipio y si no existe agregar y obtener id_location municipio
            String queryMuni = "select location_id from dbo.location where location_type='municipio' and
description='" + row["resMunicipio"] + "';";
            var readerMuni = DBConn.ReadDBIntegracion(queryMuni);
            string muniId = readerMuni;
            //Paso 5: crear y obtener office location
            String queryOffice = "select office_location_id FROM dbo.office_location where
town_id='803ba0a0-7d68-4911-9f3f-35bfda125bc9' and city_id='bd1b1234-3247-44ad-9425-68a185b0a068'
and address_line1='Santa elena'";
            var readerOffice = DBConn.ReadDBIntegracion(queryOffice);
            string offiId = readerOffice;
            //Paso 6: crear y obtener home location
```

```
            String queryHome = "select home_location_id FROM dbo.home_location where town_id='" +
muniId + "' and city_id='" + deptoId + "' and address_line1='" + row["Address"] + "'";
            var readerHome = DBConn.ReadDBIntegracion(queryHome);
            string homeId = readerHome;
            if (homeId == "")
            {
                Console.WriteLine("insertando informacion en base de integracion");
                String addHome = "insert into
home_location(town_id,city_id,country_id,region_id,address_line1,created_by) values('" + muniId + "','" +
deptoId + "','" + paisId + "','" + regionId + "','" + row["Address"] + "','Integracion');";
                DBConn.ConnectDBIntegracion(addHome);
                queryHome = "select home_location_id FROM dbo.home_location where town_id='" + muniId +
"' and city_id='" + deptoId + "' and address_line1='" + row["Address"] + "'";
                readerHome = DBConn.ReadDBIntegracion(queryHome);
                homeId = readerHome;
            }
            //Paso 7: buscar centro de costo y si no existe agregar y obtener centro de costo
            String queryCentroCosto = "select * from cost_center where cost_center_code='" +
row["cost_center"] + "'";
            var readerCentroCosto = DBConn.ReadDBIntegracion(queryCentroCosto);
            string centroCostoId = readerCentroCosto;
            if (centroCostoId == "")
            {
                Console.WriteLine("insertando informacion en base de integracion");
                String addCentroCost = "insert into cost_center(cost_center_code,state,description) values('" +
row["cost_center"] + "','A','BTO');";
                DBConn.ConnectDBIntegracion(addCentroCost);
                queryCentroCosto = "select * from cost_center where cost_center_code='" + row["cost_center"]
+ "'";
                readerCentroCosto = DBConn.ReadDBIntegracion(queryCentroCosto);
                centroCostoId = readerCentroCosto;
            }
            String queryAccount = "select * from account where account_code='" + row["account_id"] + "'";
            string readerAccount = DBConn.ReadDBIntegracion(queryAccount);
            String accountId = readerAccount;
            if(accountId == "")
            {
                Console.WriteLine("insertando informacion en base de integracion");
                String addAccount = "insert into account(account_code,description,state,created_by) VALUES('"
+ row["account_id"] + "','BTO','A','Integracion');";
                DBConn.ConnectDBIntegracion(addAccount);
                queryAccount = "select * from account where account_code='" + row["account_id"] + "'";
                readerAccount = DBConn.ReadDBIntegracion(queryAccount);
                accountId = readerAccount;
            }
            Console.WriteLine("insertando informacion en base de integracion");
            //Paso 8: agregar empleado
            String addEmp = "INSERT into dbo.employee(employee_code, office_location_id,
home_location_id, start_date,account_id,cost_center_id,created_by) " +
                "values('" + row["employee_id"] + "','" + offiId + "','" + homeId + "','" + row["start_date"] + "','"+
accountId + "','" + centroCostoId + "','Integracion');";
```

```
            DBConn.ConnectDBIntegracion(addEmp);
        }
        Console.WriteLine("Leyendo informacion de incidentes");
        ArrayList listEvt = MockUpData.GetMockupEvent();

        for (int i = 0; i < listEvt.Count; i++)
        {
            IDictionary<string, string> row = (IDictionary<string, string>)listEvt[i];
            //Paso 9: agregar evento al catalogo si no existe
            String queryEvtCat = "select catalog_id from sys_catalog where form_code='evento' and value='"+
row["eventId"] + "'";
            String readerEvtCat = DBConn.ReadDBIntegracion(queryEvtCat);
            String evtCatId = readerEvtCat;
            if (evtCatId == "")
            {
                String addEvtCat = "insert into
sys_catalog(form_code,catalog_type_code,language_code,value,label,state,created_by)
VALUES('evento','evento_id','sp','"+ row["eventId"] + "','"+ row["event"] + "','A','Integracion');";
                DBConn.ConnectDBIntegracion(addEvtCat);
                queryEvtCat = "select catalog_id from sys_catalog where form_code='evento' and value='" +
row["eventId"] + "'";
                readerEvtCat = DBConn.ReadDBIntegracion(queryEvtCat);
                evtCatId = readerEvtCat;
            }
            //Paso 10: Agregar evento ligado al empleado
            String queryOffice = "select office_location_id from employee where employee_code='" +
row["employeeid"] + "';";
            String readerOffice = DBConn.ReadDBIntegracion(queryOffice);
            String officeId = readerOffice;

            String queryHome = "select home_location_id from employee where employee_code='" +
row["employeeid"] + "';";
            String readerHome = DBConn.ReadDBIntegracion(queryHome);
            String homeId = readerHome;

            String queryEmp = "select employee_id from employee where employee_code='" +
row["employeeid"] + "';";
            String readerEmp = DBConn.ReadDBIntegracion(queryEmp);
            String empId = readerEmp;

            Console.WriteLine("insertando informacion en base de integracion");
            String addEvt = "insert into event(office_location_id,home_location_id,event_category_id,
event_start_date_time,event_end_date_time,employee_id_reporter,created_by) values('" + officeId + "','"+
homeId + "','" + evtCatId + "','" + row["startdate"] + "','"+row["endDate"]+"','"+ empId + "','Integracion');";
            DBConn.ConnectDBIntegracion(addEvt);
        }
    }
  }
}
```

MockUpData.cs

```
using System;
using System.Data.SqlClient;
using System.Collections.Generic;
using System.Collections;
using integracionAlCubo;

namespace integracionAlCubo
{
   internal class MockUpData
   {
      public static ArrayList GetMockupEmployee()
      {
         ArrayList list = new();
         IDictionary<string, string> row;

         //Start adding data
         row = new Dictionary<string, string>(); row.Add("employee_id", "677"); row.Add("cost_center",
"16821"); row.Add("Address", "Col Miramonte Av Bernal #XXX San Salvador San Salvador");
row.Add("resMunicipio", "COLON"); row.Add("resDepartamento", "LA LIBERTAD"); row.Add("Cargo", "Gerente
Desarrollo"); row.Add("employee_id_supervisor", "312"); row.Add("start_date", "2012/12/19");
row.Add("office", "ANTIGUO CUSCATLAN"); row.Add("WorkPlaceMode", "On Site"); row.Add("account_id",
"612"); row.Add("InternetProvider", ""); list.Add(row);
         row = new Dictionary<string, string>(); row.Add("employee_id", "657"); row.Add("cost_center",
"16141"); row.Add("Address", "Col Miramonte Av Bernal #XXX San Salvador San Salvador");
row.Add("resMunicipio", "MEJICANOS"); row.Add("resDepartamento", "SAN SALVADOR"); row.Add("Cargo",
"Agente"); row.Add("employee_id_supervisor", ""); row.Add("start_date", "2012/11/30"); row.Add("office",
"ANTIGUO CUSCATLAN"); row.Add("WorkPlaceMode", "On Site"); row.Add("account_id", "55");
row.Add("InternetProvider", ""); list.Add(row);
         row = new Dictionary<string, string>(); row.Add("employee_id", "790"); row.Add("cost_center",
"16964"); row.Add("Address", "Col Miramonte Av Bernal #XXX San Salvador San Salvador");
row.Add("resMunicipio", "APOPA"); row.Add("resDepartamento", "SAN SALVADOR"); row.Add("Cargo",
"Agente"); row.Add("employee_id_supervisor", ""); row.Add("start_date", "2013/8/10"); row.Add("office",
"ANTIGUO CUSCATLAN"); row.Add("WorkPlaceMode", "On Site"); row.Add("account_id", "55");
row.Add("InternetProvider", ""); list.Add(row);
         //...
         row = new Dictionary<string, string>(); row.Add("employee_id", "2865"); row.Add("cost_center",
"16955"); row.Add("Address", "Resd Holanda Pj 30 San Salvador San Salvador"); row.Add("resMunicipio",
"APOPA"); row.Add("resDepartamento", "SAN SALVADOR"); row.Add("Cargo", "Agente");
row.Add("employee_id_supervisor", "2792"); row.Add("start_date", "2018/5/19"); row.Add("office", "ANTIGUO
CUSCATLAN"); row.Add("WorkPlaceMode", "WFH"); row.Add("account_id", "55"); row.Add("InternetProvider",
"Tigo Solution"); list.Add(row);

         return list;

      }


      public static ArrayList GetMockupEvent()
```

```
        {
            ArrayList list = new();
            IDictionary<string, string> row;

            //Start adding data
            row = new Dictionary<string, string>(); row.Add("employeeid", "2799"); row.Add("city", "San Salvador
- San Salvador"); row.Add("Account", "17000"); row.Add("startdate", "2021-4-21 3:30"); row.Add("endDate",
"2021-4-21 12:0"); row.Add("eventId", "14"); row.Add("event", "TOG-System/App issues"); row.Add("typeId",
"15"); row.Add("type", "TOG"); row.Add("Comments", ""); list.Add(row);
            row = new Dictionary<string, string>(); row.Add("employeeid", "2799"); row.Add("city", "San Salvador
- San Salvador"); row.Add("Account", "17000"); row.Add("startdate", "2021-4-22 3:0"); row.Add("endDate",
"2021-4-22 12:0"); row.Add("eventId", "18"); row.Add("event", "Other (Add on Comments)");
row.Add("typeId", "19"); row.Add("type", "TOG"); row.Add("Comments", ""); list.Add(row);
            row = new Dictionary<string, string>(); row.Add("employeeid", "2799"); row.Add("city", "San Salvador
- San Salvador"); row.Add("Account", "17000"); row.Add("startdate", "2021-5-26 5:58"); row.Add("endDate",
"2021-5-26 6:7"); row.Add("eventId", "13"); row.Add("event", "TOG-Power Outages"); row.Add("typeId", "14");
row.Add("type", "TOG"); row.Add("Comments", ""); list.Add(row);
            //...
            row = new Dictionary<string, string>(); row.Add("employeeid", "1763"); row.Add("city", "La Libertad -
Santa Tecla"); row.Add("Account", "17006"); row.Add("startdate", "2021-12-28 7:7"); row.Add("endDate",
"2021-12-28 7:43"); row.Add("eventId", "14"); row.Add("event", "TOG-System/App issues"); row.Add("typeId",
"15"); row.Add("type", "TOG"); row.Add("Comments", ""); list.Add(row);

            return list;

        }
    }
}
```

DBConn.cs

```
using System;
using System.Data.SqlClient;
using System.Collections.Generic;
using System.Collections;
using integracionAlCubo;
using System.Data;

namespace integracionAlCubo
{
    internal class DBConn
    {
        public static void ConnectDBRRHH(String query)
        {
            Console.WriteLine("Getting Connection ...");
            /*var datasource = @"DESKTOP-PC\SQLEXPRESS";//your server
            var database = "Students"; //your database name
            var username = "sa"; //username of server to connect
            var password = "password"; //password
```

```
    //your connection string
    string connString = @"Data Source=" + datasource + ";Initial Catalog="
            + database + ";Persist Security Info=True;User ID=" + username + ";Password=" + password;
    //create instanace of database connection
    SqlConnection conn = new SqlConnection(connString);
    try
    {
        Console.WriteLine("Openning Connection ...");
        //open connection
        conn.Open();
        Console.WriteLine("Connection successful!");
        using (SqlCommand command = new SqlCommand(query, conn))
        {
            command.ExecuteNonQuery(); //execute the Query
            Console.WriteLine("Query Executed.");
        }
    }
    catch (Exception e)
    {
        Console.WriteLine("Error: " + e.Message);
    }
    Console.Read();*/
}



private void ConnectDBOperacion(String query)
{
    Console.WriteLine("Getting Connection ...");
    var datasource = @"DESKTOP-PC\SQLEXPRESS";//your server
    var database = "Students"; //your database name
    var username = "sa"; //username of server to connect
    var password = "password"; //password
    //your connection string
    string connString = @"Data Source=" + datasource + ";Initial Catalog="
            + database + ";Persist Security Info=True;User ID=" + username + ";Password=" + password;
    //create instanace of database connection
    SqlConnection conn = new SqlConnection(connString);
    try
    {
        Console.WriteLine("Openning Connection ...");

        //open connection
        conn.Open();
        Console.WriteLine("Connection successful!");

        using (SqlCommand command = new SqlCommand(query, conn))
        {
            command.ExecuteNonQuery(); //execute the Query
            Console.WriteLine("Query Executed.");
        }
```

```
            }
            catch (Exception e)
            {
                Console.WriteLine("Error: " + e.Message);
            }
            Console.Read();
        }

        public static void ConnectDBIncidencias(String query)
        {
            Console.WriteLine("Getting Connection ...");

            var datasource = @"DESKTOP-PC\SQLEXPRESS";//your server
            var database = "Students"; //your database name
            var username = "sa"; //username of server to connect
            var password = "password"; //password

            //your connection string
            string connString = @"Data Source=" + datasource + ";Initial Catalog="
                    + database + ";Persist Security Info=True;User ID=" + username + ";Password=" + password;

            //create instanace of database connection
            SqlConnection conn = new SqlConnection(connString);
            try
            {
                Console.WriteLine("Openning Connection ...");
                //open connection
                conn.Open();

                Console.WriteLine("Connection successful!");
                using (SqlCommand command = new SqlCommand(query, conn))
                {
                    command.ExecuteNonQuery(); //execute the Query
                    Console.WriteLine("Query Executed.");
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Error: " + e.Message);
            }
            Console.Read();
        }

        public static void ConnectDBIntegracion(String query)
        {
            Console.WriteLine("Getting Connection ...");
            var datasource = @"server-tesis-udb.database.windows.net";//@"DESKTOP-PC\SQLEXPRESS";//your
server
            var database = "db-tesis-udb"; //your database name
            var username = "desarrollo"; //username of server to connect
            var password = "Tesis$123"; //password
```

```
            //your connection string
            string connString = @"Data Source=" + datasource + ";Initial Catalog="
                    + database + ";Persist Security Info=True;User ID=" + username + ";Password=" + password;
            //create instanace of database connection
            SqlConnection conn = new SqlConnection(connString);
            try
            {
                Console.WriteLine("Openning Connection ...");
                //open connection
                conn.Open();
                Console.WriteLine("Connection successful! Executing {0}", query);
                using (SqlCommand command = new SqlCommand(query, conn))
                {
                    command.ExecuteNonQuery(); //execute the Query
                    Console.WriteLine("Query Executed.");
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Error: " + e);
            }
            finally { conn.Close(); Console.WriteLine("Connection Closed!"); }
            //Console.Read();
        }



        public static String ReadDBIntegracion(String query)
        {
            Console.WriteLine("Getting Connection ...");
            var datasource = @"server-tesis-udb.database.windows.net";//@"DESKTOP-PC\SQLEXPRESS";//your
server
            var database = "db-tesis-udb"; //your database name
            var username = "desarrollo"; //username of server to connect
            var password = "Tesis$123"; //password
            String response="";

            //your connection string
            string connString = @"Data Source=" + datasource + ";Initial Catalog="
                    + database + ";Persist Security Info=True;User ID=" + username + ";Password=" + password;
            //create instanace of database connection
            SqlConnection conn = new SqlConnection(connString);
            try
            {
                Console.WriteLine("Openning Connection ...");
                //open connection
                conn.Open();
                Console.WriteLine("Connection successful!");
                using (SqlCommand command = new SqlCommand(query, conn))
                {
```

```
                SqlDataReader reader = command.ExecuteReader();//execute the Query
                while (reader.Read())
                {
                    var dataRecord = (IDataRecord)reader;
                    response = Convert.ToString(value: dataRecord.GetValue(0));
                }
                Console.WriteLine("Query Executed.");
                //conn.Close(); Console.WriteLine("Connection Closed!");
            }
        }
        catch (Exception e)
        {
            Console.WriteLine("Error: " + e);
        }
        finally { conn.Close(); Console.WriteLine("Connection Closed!"); }

        return response;

        //Console.Read();

    }
  }
}
```