# UNIVERSIDAD NACIONAL DE SAN CRISTÓBAL DE HUAMANGA

## FACULTAD DE INGENIERÍA DE MINAS, GEOLOGÍA Y CIVIL

## ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS


# "COMPONENTE SOFTWARE PARA CONTROLAR LAS VERSIONES DEL ESQUEMA DE BASE DE DATOS RELACIONAL, AHREN CONTRATISTAS GENERALES SAC - AYACUCHO, 2017"


| | |
|---|---|
| **Tesis presentada por** | **:** Bach. PALOMINO PARIONA, Alex |
| **Para optar el título profesional de** | **:** INGENIERO DE SISTEMAS |
| **Tipo de investigación** | **:** Aplicada y Descriptiva |
| **Asesor (a)** | **:** Ing. CARRILLO RIVEROS, Elinar |



AYACUCHO - PERU

2018

## DEDICATORIA

A mi mamá por ser un ejemplo a seguir de trabajo y colaboración con los demás.

A mi papá por ayudarme y apoyarme siempre con sus consejos y su ejemplo de perseverancia, rectitud, integridad y ética.

A mis hermanos por estar conmigo y por la paciencia que me han tenido.

A mi novia Ida Gissela Mauricio Huaraca, que con su valor y entrega ha sido una persona incondicional en mi vida, ha sido mi soporte, mi mejor amiga, mi consejera, mi todo para seguir adelante y no bajar los brazos en los momentos difíciles y sobre todo por su innegable dedicación, amor y paciencia.

A mis amigos que compartieron los años de aprendizaje, y que me apoyaron para concluir mis metas.

## AGRADECIMIENTO

A la Universidad Nacional de San Cristóbal de Huamanga, mi alma máter, a todos los docentes que de alguna forma han contribuido en mi formación profesional.

A mi asesora de tesis la Ing. Elinar CARRILLO RIVEROS, por la orientación y la ayuda que me bridó para la realización de esta tesis, por su apoyo y amistad que me permitieron aprender mucho más que lo estudiado en el pregrado.

A todas las personas que de una u otra manera estuvieron a mi lado, que me enseñaron y me dieron ánimos. Gracias a todos.

## CONTENIDO

## CAPÍTULO III

## METODOLOGÍA DE LA INVESTIGACIÓN

## CAPÍTULO IV

## ANÁLISIS RESULTADOS DE LA INVESTIGACIÓN



## CAPÍTULO V

## CONCLUSIONES Y RECOMENDACIONES

## RESUMEN

El problema central en AHREN Contratistas Generales SAC es la deficiente administración de las versiones del esquema de bases de datos relacionales, estos esquemas están geográficamente distribuidos, al agregar cambios en el esquema de base de datos relacional, conlleva a que un personal relacionado con el desarrollo de la aplicación tiene que viajar a las diferentes obras para actualizar dichos cambios en el esquema de base datos relacional. Ello implica mayores costos a la empresa y que las versiones del esquema de bases de datos relacionales sean incompatibles entre ellas y con la versión de la aplicación.

El objetivo de este trabajo de investigación es desarrollar el componente software de control de versiones para administrar eficientemente las versiones del esquema de base de datos relacional en AHREN Contratistas Generales SAC, el tipo de investigación es aplicada.

La presente investigación se realizó en los diferentes esquemas de bases de datos relacionales de la aplicación offline de control de almacenes en obra de AHREN Contratistas Generales SAC, durante el año 2017.

Para la investigación se utilizó la metodología ágil scrum, técnicas en control de versiones y Entity Framework 6, para crear los objetos necesarios que representan el esquema de base de datos relacional, de esta manera mantener sincronizada la aplicación y su respectivo esquema de base de datos relacional.


**PALABRAS CLAVE:** Control de versiones, bases de datos relacionales, Metodología ágil Scrum, Entity Framework 6.

# INTRODUCCIÓN

Mantener el control de todo el esquema de base de datos relacional en su desarrollo y evolución es una tarea compleja para hacerla manualmente. El componente software proporcionará una forma única de comunicación e interacción entre la aplicación y la base datos relacional; puesto que, en la mayoría de las aplicaciones, la base de datos relacional es de vital importancia y necesaria para agilizar los procesos y mantener operacional toda la aplicación.

La motivación principal para desarrollar un componente software, que controlar las versiones del esquema de bases de datos relacionales de manera eficiente, permitiendo la administración de los cambios y actualizaciones de la aplicación en la base de datos relacional, sin tener la necesidad de la intervención del desarrollador, de esta manera optimizar tiempo y la corrección de errores durante el despliegue de la aplicación.

Actualmente existen sistemas manejadores de control de versiones genéricas, pero con diversas limitaciones; puesto que, estos sistemas de control de versiones están diseñadas para controlar la evolución del código fuente y no así un esquema de base de datos relacional.

Los objetivos específicos son: a) Analizar, diseñar la capa de modelo de datos para configurar las versiones del esquema de base de datos relacional. b) Implementar y probar la capa de contexto para administrar las versiones del esquema de base de datos relacional.

# CAPÍTULO I
# PROBLEMA DE LA INVESTIGACIÓN

## 1.1 DIAGNÓSTICO Y ENUNCIADO DEL PROBLEMA

AHREN Contratistas Generales SAC, es una empresa Ayacuchana, fundada el año 2007, dedicada al servicio de consultoría, ejecución de obras, provisión de bienes y servicios desde la etapa de planificación, consultoría, ejecución y puesta en operación de los proyectos a su cargo.

Las bases de datos relacionales son el elemento central de la mayoría de las aplicaciones, porque contienen la información relacionada al negocio, sin embargo, no se les ha dado la importancia requerida. Las bases de datos relacionales son tan complejas que evolucionan en el tiempo; por otro lado, no importa qué metodología de desarrollo se utilice; por lo general, cuando se requiere un cambio significativo en el código fuente de la aplicación, dicho cambio casi siempre afecta a la base de datos relacional. Administrar los cambios de una base de datos relacional no es una tarea fácil de hacer manualmente ni mucho menos por un solo desarrollador.

La problemática con el control de versiones del esquema de bases de datos relacionales es muy amplia, simplemente consultando con profesionales que se dedican al desarrollo de aplicaciones empresariales, se obtiene respuestas superficiales de cómo crear "versiones" copia del esquema de bases de datos relacionales para cada despliegue de la aplicación; en general, mencionan la palabra "scripts", sabiendo que dicho problema cuesta mucho.

Los desarrolladores de aplicaciones son los encargados de realizar las modificaciones al esquema de base de datos relacional, los principales cambios que se hace son durante el diseño y desarrollo inicial de la aplicación; así mismo, se sabe que existen proyectos muy grandes que evolucionan para satisfacer las necesidades del cliente, ahí es donde resalta la necesidad de un componente software para controlar las versiones del esquema de base de datos relacional que facilite administrar las diferentes versiones de éste y evite problemas irreversibles.

El control de versiones, es la administración de los cambios que ocurren sobre un producto, donde una versión es un estado del producto en un momento específico, siendo el código fuente el objeto que más se administra además de otros componentes como documentos o imágenes que formen parte de la aplicación.

El control se puede llevar manualmente, aunque existen herramientas para automatizar este proceso y es, desde el punto de vista, la vía más económica en cuanto a tiempo y recursos de almacenamiento. Un control manual implicaría guardar una copia de un documento y su registro descriptivo cada vez que se realice un cambio; por otro lado, para regresar a una versión anterior se deberá aplicar el inverso de cada uno de los cambios que contiene dicho registro.

El problema existente en AHREN Contratistas Generales SAC es la deficiente administración de las versiones del esquema de bases de datos relacionales de la aplicación offline de control de almacenes en obra, estos esquemas de bases de datos relacionales están en diferentes versiones y geográficamente distribuidos, cuando se agrega un cambio en la aplicación también involucra cambios en el esquema de base de datos

relacional. Esto genera problemas para enviar la aplicación a las diferentes obras; puesto que, un personal relacionado con el desarrollo de la aplicación tiene que viajar a las diferentes obras para que pueda crear las columnas, llaves foráneas, operaciones entre columnas y tablas faltantes en el esquema de base datos relacional. Ello implica mayores costos a la empresa así mismo genera que las versiones del esquema de bases de datos relacionales sean incompatibles con la versión de la aplicación.

Por lo tanto, no existe una manera eficiente de controlar las versiones del esquema de base de datos relacional de la aplicación offline de control de almacenes en obra de los diferentes centros de costo que administra AHREN Contratistas Generales SAC.

## 1.2 DEFINICIÓN DEL PROBLEMA DE INVESTIGACIÓN

### 1.2.1 PROBLEMA GENERAL

¿Cómo el componente software controla las versiones del esquema de base de datos relacional, AHREN Contratistas Generales SAC, 2017?

### 1.2.2 PROBLEMAS ESPECÍFICOS

a) ¿Cómo la capa de modelo de datos permite configurar las versiones del esquema de base de datos relacional?

b) ¿Cómo la capa de contexto permite administrar las versiones del esquema base de datos relacionales?

## 1.3 OBJETIVOS DE LA INVESTIGACIÓN

### 1.3.1 OBJETIVO GENERAL

Desarrollar el componente software mediante técnicas e instrumentos, la metodología ágil Scrum, control de versiones y patrón Code First de Entity Framework 6, con la finalidad de controlar las versiones del esquema de base de datos relacional, AHREN Contratistas Generales SAC, 2017.

### 1.3.2 OBJETIVOS ESPECÍFICOS

a) Analizar y diseñar la capa de modelo de datos para configurar las versiones del esquema de base de datos relacional.

b) Implementar y probar la capa de contexto para administrar las versiones del esquema de base de datos relacional.

## 1.4 JUSTIFICACIÓN Y DELIMITACIÓN DE LA INVESTIGACIÓN

### 1.4.1 IMPORTANCIA ECONÓMICA

La implementación de este componente software de control de versiones del esquema de base de datos relacional, permitirá reducir el esfuerzo físico de implantación de diferentes aplicaciones que cuenten con acceso a base de datos relacional, de esta manera se procederá a reducir el tiempo y costo en la implantación.

### 1.4.2 JUSTIFICACIÓN

En la actualidad la empresa AHREN Contratistas Generales, distribuye la aplicación offline de control de almacenes en obra a diferentes centros de costo. La aplicación está diseñada para que funcione sin internet, ello requiere que los esquemas de base de datos relacional estén sincronizados

con la versión de la aplicación, las versiones del esquema de base de datos relacional y la aplicación varían de un centro de costo a otro.

Al contar con el componente software de control de versiones del esquema de base de datos relacional se automatizará la generación y actualización del esquema de base de datos relacional, además de simplificar el esfuerzo de procesamiento de tareas complejas. La generación de scripts diferenciales imposibilita llevar el control manual del esquema de base de datos relacional geográficamente distribuidos, esto conlleva a la ineficiencia en controlar las diferentes versiones del esquema de base de datos relacional, un componente software de control de versiones apoyará en la adaptación a este tipo de entornos. Así mismo, este proyecto de investigación servirá como guía a los desarrolladores de aplicaciones empresariales, arquitectos de software e investigadores.

Por lo tanto, es necesario implementar el componente software para versionar el esquema de base de datos relacional; de esta manera, sincronizar los cambios del esquema de base de datos relacional en lugares geográficamente distribuidos, de manera consistente, reproducible y probable; con la capacidad de volver a una versión estable (código fuente y esquema de base de datos relacional) en caso de que algo falle.

### 1.4.3 DELIMITACIÓN

La presente investigación se limitó a implementar y justificar el componente software para controlar las versiones del esquema de bases de datos relacionales. Haciendo uso de la revisión documentaria; además, la información que se levantará será de los esquemas de bases de datos relacionales de la aplicación offline de control de almacenes en obra de AHREN Contratistas Generales SAC, 2017.

# CAPÍTULO II
# REVISIÓN DE LITERATURA

## 2.1 ANTECEDENTES DE LA INVESTIGACIÓN

Según la investigación de Ramos, D. (2012), concluye que, al utilizar un sistema de control de versiones genéricos, estos no satisfacen las necesidades de controlar las versiones del esquema de base de datos relacional, dado que estos sistemas no fueron creados para ello; por otra parte, la mayoría de las herramientas especializadas en el control de versiones de esquemas de bases de datos relacionales usan sistemas de control de versiones genéricos, estas herramientas especializadas que utilizan un sistema de control de versiones como apoyo, tienen otro obstáculo muy marcado, que es la compatibilidad con tipos de bases de datos relacionales, es decir, se especializan en un tipo de bases de datos en específico dejando fuera a los demás tipos de bases de datos.

Según la investigación de Callejas, M., Peñalosa, D., & Alacón, A. (2011), concluyen que para la selección correcta del uso de un determinado componente software de persistencia, previamente se deben realizar distintas comparaciones entre éstos. Los aspectos comúnmente evaluados son los que tienen que ver con cuál tiene un mayor rendimiento en cuanto a uso de memoria, tiempo de ejecución, entre otros. De igual manera es necesario tener en cuenta aspectos como volumen de trabajo o de información que fluye a través del sistema, así como el tiempo de respuesta, el costo por transacción y medidas de utilización de recursos, para evaluar el rendimiento de una aplicación.

## 2.2 MARCO TEÓRICO

### 2.2.1 COMPONENTE SOFTWARE

El componente software es un paquete coherente de artefactos de software que puede ser desarrollado independientemente y entregado como una unidad, este puede ser compuesto e intercambiado con otro componente para construir algo mucho más grande. (Szyperski, 2002).

Un componente software es una parte no trivial, casi independiente, y reemplazable de un sistema con una funcionalidad dentro de un contexto y una arquitectura bien definida y provee de un conjunto de interfaces. (Krutchen & Kroll, 2002).

### 2.2.2 SOFTWARE

Es el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación. Considerando esta definición, el concepto de software va más allá de los programas de computación en sus distintos estados: código fuente, binario o ejecutable; también su documentación, los datos a procesar e incluso la información de usuario forman parte del software: es decir, abarca todo lo intangible, todo lo no físico relacionado. (Pressman, 2010).

Según Santos, et al. (2009), existe una importante línea de desarrollo de aplicaciones, en esencia aplicaciones implementadas con la tecnología web y que únicamente requieren un navegador estándar para su utilización que ofrece de forma gratuita, y que junto con otras soluciones conforman una interesante oferta para las instituciones.

### 2.2.3 TECNOLOGIA CLIENTE/SERVIDOR

En la arquitectura cliente – servidor, una o varias computadoras son los servidores, que responden a un número grande de clientes conectados a través de la red. Los clientes suelen ser computadoras personales de propósito general orientadas al usuario final; en ocasiones, los servidores son intermediarios entre los clientes y otros servidores más especializados. Con el auge de internet, la arquitectura cliente – servidor ha adquirido una mayor relevancia, ya que es un principio básico de funcionamiento de la World Wide Web: un usuario que mediante un visualizador (cliente) solicita a un servicio (página HTML, etc.) a una computadora que funciona como servidor (Trejo, 2002).

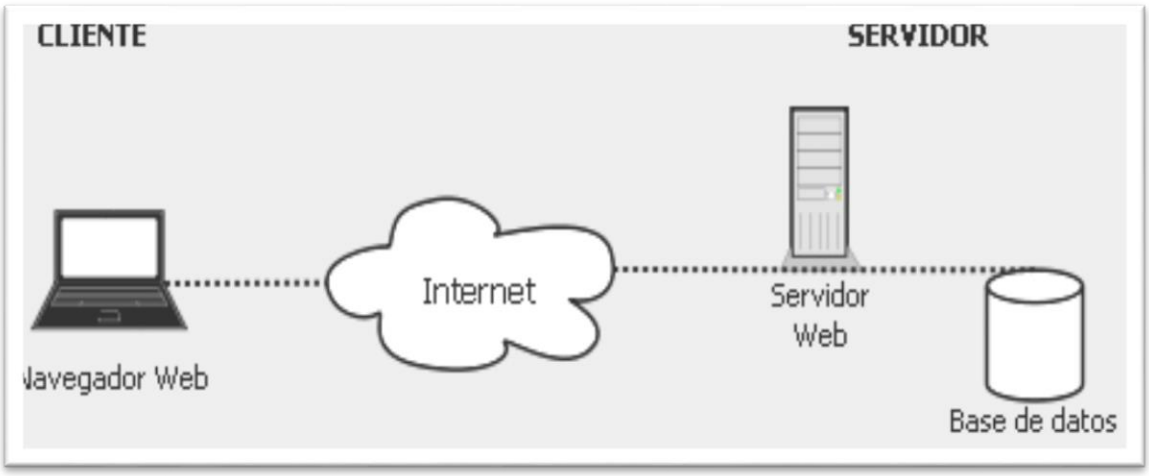



Figura N° 2.1: Tecnología cliente – servidor (Trejo, 2002).

La tecnología Cliente/Servidor engloba entonces los elementos siguientes: una red de computadoras, un protocolo de comunicación, varias computadoras y los sistemas operativos de red correspondientes a cada una de ellas; todos ellos son elementos que deben ser correctamente elegidos para no invertir recursos en unir piezas que sean incompatibles entre sí. Cliente/Servidor se refiere a una arquitectura o división de

responsabilidades en las aplicaciones distribuidas de software (Pressman, 2011).

Según Pressman (2005), Cliente/Servidor es un tipo especial de Sistemas Distribuidos. El Cliente es la aplicación (a quién se le conoce como parte frontal o front-end) y el Servidor (back – end) que puede ser de diferentes tipos dependiendo del área de interés del constructor del sistema distribuido.

### 2.2.4 CAPA CONTEXTO

La capa contexto fue creado para facilitar a los desarrolladores el acceso y la manipulación del esquema de bases de datos relacionales. En pocas palabras, la capa contexto está conformada por clases que encapsulan los patrones y funciones utilizados con mayor frecuencia al trabajar con Entity Framework. (Gueddari, 2014).

Según Griffin (2013), la capa contexto representa una sesión del esquema de base de datos relacional y permite a los desarrolladores de aplicaciones consultar y guardar datos en ella. En la capa contexto, un DbSet define una propiedad para cada clase, lo que permite a los desarrolladores de aplicaciones consultar y realizar operaciones con estas clases.

Según MSDN (2014), en la capa contexto, una instancia representa una combinación de los patrones de unidad de trabajo y repositorio, de esta manera se puede usar para consultar una base de datos y agrupar los cambios.

### 2.2.5 CAPA MODELO DE DATOS

La capa modelo de datos está conformado por objetos primariamente definidos por su identidad, se le denomina entidad. (Evans, 2003).

Según Miguel y Piatthini (1993), la capa modelo de datos está conformado por conjunto de conceptos, reglas y convenciones que nos permiten describir los datos del mundo real.

Según Korth & Silberschatz (2002), la capa modelo de datos está conformada por abstracciones del mundo real. Esta capa abstrae la representación de entidades y sus características (relaciones, atributos) dentro de un esquema de base de datos relacional.

### 2.2.6 TIPO DE DATO

Un tipo de dato informático o simplemente tipo de dato es un atributo de los datos que indica al ordenador sobre la clase de datos que se va a trabajar. En esencia, es un espacio en memoria con restricciones sobre el cual se puede operar para un fin determinado. (Cardelli & Wegner, 1985).

### 2.2.7 FRAMEWORK

Uno de los grandes retos de la programación orientada a objetos es facilitar la combinación y el acceso a cualquier tipo de información, como una característica integrada a un lenguaje de programación. (Vanegas, 2013).

Un framework es un conjunto de mejores prácticas, métodos y herramientas. En el desarrollo de software, un framework o infraestructura digital, es una estructura conceptual y tecnológica de soporte definido, normalmente con

artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto. (Escobar, Losavio, & Ortega, 2012).

### 2.2.8 ENTITY FRAMEWORK

Entity Framework como un conjunto de componentes del software que pueden ser usados por los programadores para acceder a datos y a servicios de datos. Es frecuentemente usado por los programadores para modificar y acceder a datos almacenados en un sistema gestor de bases de datos relacionales, así como también a datos en fuentes no relacionales. (Mindy & Erick, 2011).

Entity Framework es una mejora a ADO.NET creada por Microsoft, que dota a los desarrolladores de un mecanismo adicional para acceder a los datos y trabajar con los resultados, además de los Data Readers y Data Sets utilizados en versiones anteriores.

Entity Framework (EF) abstrae la estructura de la base de datos y todo el acceso y almacenamiento de datos se realiza frente a un modelo conceptual de datos que refleja los objetos de negocio. Los ítems usados para representar el modelo conceptual se conocen como Entidades. (Suárez, 2011).

Según MSDN (2014), es un framework de mapeo objeto relacional (ORM) que permite a los desarrolladores trabajar con datos relacionales como objetos de dominio específico, eliminando la necesidad de gran parte del código de acceso a datos que el desarrollador usualmente necesita escribir.

Usando Entity Framework, los desarrolladores realizan consultas a través de LINQ para recuperar y manipular datos como objetos fuertemente tipados. La implementación del ORM de Entity Framework provee de un traductor de consultas para que los desarrolladores puedan enfocarse en la lógica específica de la aplicación en lugar de los fundamentos de acceso a datos, es útil en tres escenarios (ver Figura N° 2.2). a) Si existe una base de datos o si se quiere diseñar la base de datos por delante de las otras partes de la aplicación, llamado Database First. b) Si se quiere enfocar en las clases de dominio y luego crear la base de datos desde las clases de dominio, llamado Code First. c) Si quiere diseñar el esquema de base de datos en el diseñador visual y luego crear la base de datos y sus clases, llamado Model First.

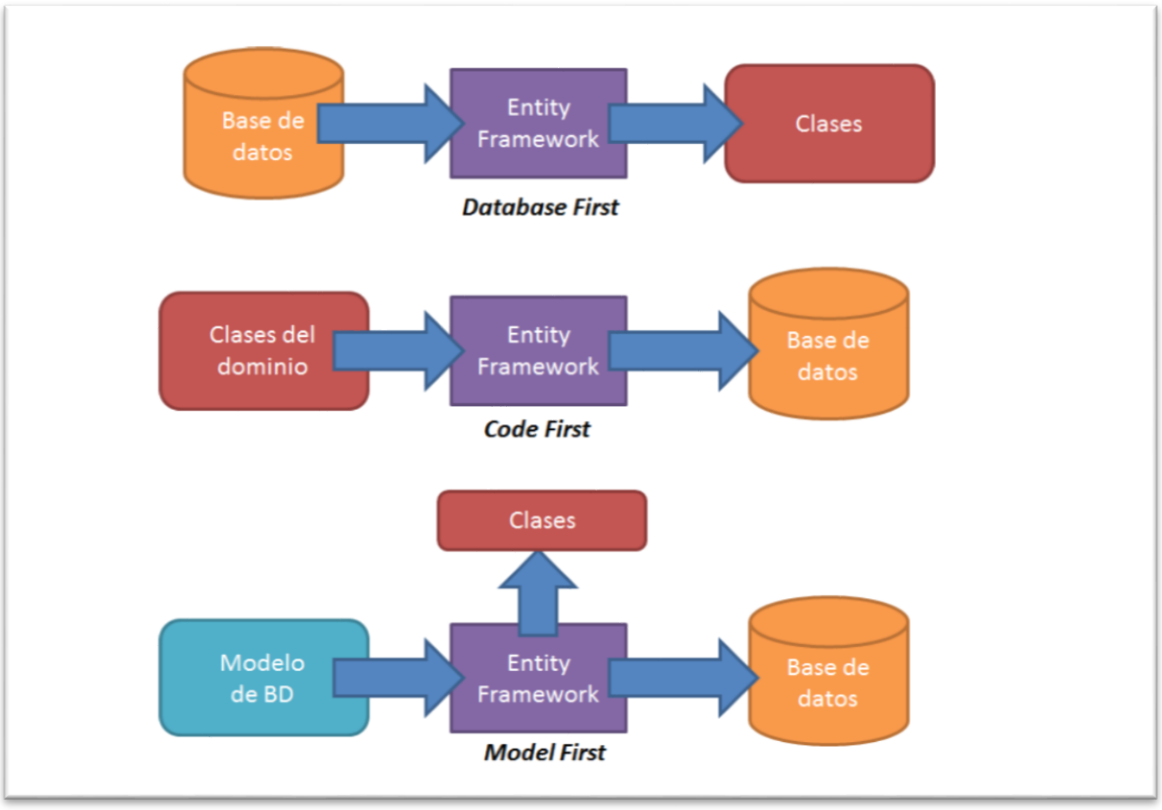



Figura N° 2.2: Escenarios de Entity Framework (MSDN, Entity Framework, 2014).

Según Microsoft (2014), Entity Framework es un framework con técnicas y métodos en programación utilizada para convertir tipos incompatibles en objetos que pueden usarse en lenguajes de programación. Las conversiones de estos tipos incompatibles en objetos de programación causan problemas a los desarrolladores, conocidos como diferencias entre esquema de base de datos relacional y el modelo de objetos en el mapeo objetos relacional (ORM). Entity Framework reduce la incompatibilidad mediante el mapeo de tablas relacionales, columnas y restricciones de clave externa dentro del modelo lógico para entidades y relaciones en el modelo conceptual.

### 2.2.9 ORM

Según MSDN (2014), es una herramienta para almacenar datos de objetos de dominio a bases de datos relacionales, de forma automatizada, sin mucha programación. Los ORM incluyen: objetos de clase del Dominio, objetos del esquema de base de datos relacional e información de mapeo de como los objetos del dominio mapean a objetos del esquema de base de datos relacional (Ver Figura N° 2.3).

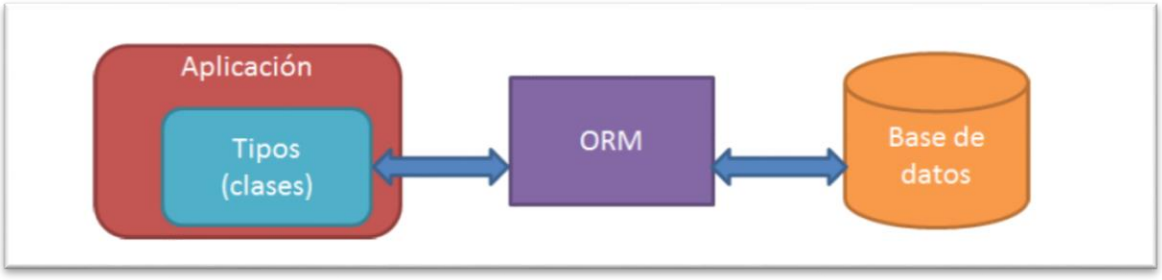



Figura N° 2.3: Objeto de mapeo relacional (MSDN, 2014)

El ORM permite enlazar un lenguaje de programación orientado a objetos con una base de datos relacional (mapeo de atributos de un objeto, con filas y columnas de una tabla de una base de datos relacional), permitiendo al desarrollador mantener una perspectiva orientada a objetos y cuidar que

se consiga aportar una solución a la lógica de negocio, eliminando el obstáculo que existía al tratar de conseguir la compatibilidad entre objetos con modelos relacionales (Object Relational Impedance Mismatch). Los ORM cumplen con las siguientes condiciones (Bauer & King, 2007):

a) Permite operaciones CRUD para la persistencia de objetos.
b) Trabaja con un lenguaje para especificar las consultas (lenguaje de consulta) que permita trabajar con clases o atributos de las clases.
c) Permite diferentes formas de especificación de mapeo de clases (metadatos) con las correspondientes tablas.
d) Permite conocer el estado de los objetos para determinar si han sido modificados.
e) Admite el uso de cargas retardadas de objetos (lazy loading) como una forma de optimizar las consultas.

Un ORM entonces debe encapsular la interacción entre la aplicación y la base de datos relacional, dejando a los desarrolladores la libertad de concentrarse en temas de negocios.

Los frameworks de persistencia no son únicamente herramientas utilizadas para el mapeo de objetos de negocio con bases relacionales, sino que además de contener un ORM ofrecen servicios que transparentan hacia el desarrollador la gestión de un objeto a una base de datos y viceversa. En la figura N° 2.4 se puede observar la estructura de un framework de persistencia.

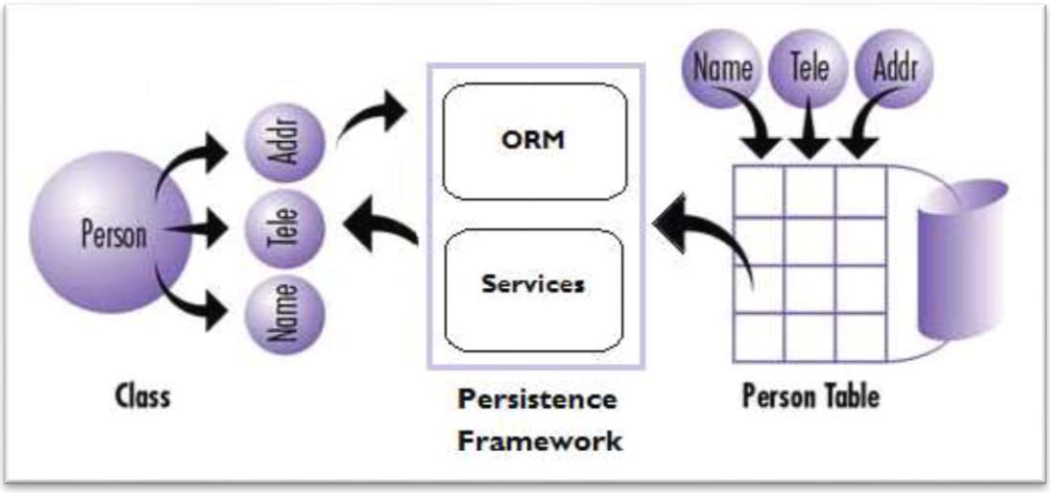



Figura N° 2.4: Framework de Persistencia (Bauer & King, 2007)

Entre los servicios que un framework de persistencia ofrece se puede nombrar los siguientes (IBM Corporation, 2004):

a) Persistencia. - Permite que un usuario almacene objetos en una sesión y pueda acceder a ellos en una sesión posterior.

b) Consultas. - Permite a las aplicaciones consultar y recuperar objetos basados en una variedad de criterios.

c) Transaccionalidad. - El soporte transaccional permite al desarrollador de la aplicación definir una unidad atómica de trabajo. En terminología de base de datos, significa que el sistema debe ser capaz de aplicar un conjunto de cambios en la base de datos, o que debe asegurarse de que ninguno de los cambios se aplique. Un framework de persistencia al menos debería proveer operaciones transaccionales como commit y rollback.

d) Concurrencia. - Los sistemas multiusuario orientados a objetos deben controlar el acceso concurrente a los objetos. Cuando un objeto es

accedido simultáneamente por muchos usuarios, el sistema debe proporcionar un mecanismo para asegurar que las modificaciones del objeto en el almacenamiento persistente se producen de una manera predecible y controlada. Los frameworks de persistencia pueden aplicar controles de tipo pesimista u optimista.

e) Relaciones. - Permite manejar las relaciones que se utilice entre los objetos. En los sistemas relacionales, las relaciones entre las entidades se implementan utilizando claves externas o claves primarias. En los sistemas orientados a objetos, las relaciones suelen ser explícitamente implementadas a través de los atributos.

### 2.2.10 LINQ

LINQ o Language Integrated Query, es un conjunto de herramientas de Microsoft para realizar todo tipo de consultas a distintas fuentes de datos: objetos, xmls, bases de datos y entre otros. Para ello, usa funciones propias, que unifica las operaciones más comunes en todos los entornos; con esto, se consigue un mismo lenguaje para todo tipo de tareas con datos. (Hummel, 2017).

Según MSDN (2014), Language Integrated Query (LINQ) es una herramienta que permite utilizar sintaxis de objetos sobre elementos de una base de datos relacional. Esta sintaxis permite integrar el código orientado a la base de datos en el contexto de la aplicación, proveyendo acceso a características como IntelliSense; una herramienta que facilita la edición de código suministrando acceso rápido a elementos en el contexto en el que se encuentra el cursor.

### 2.2.11 BASE DE DATOS RELACIONAL

Una base de datos es un conjunto de datos relacionados entre sí, datos que representan algún aspecto del mundo real, que pueden registrarse de manera lógica y coherente para que tengan un significado implícito y un propósito específico para un grupo de usuarios que la necesite. (Elmasri & Navathe, 2000).

Osorio (2008), asevera que el usuario de un sistema de base de datos puede realizar consultas de tablas, inserción de nuevas tupas, actualizaciones o borrado de las existentes. Estas operaciones se realizan mediante unos lenguajes conocidos como lenguajes de consulta relacional.

### 2.2.12 VERSIÓN DE ESQUEMA DE BASE DE DATOS RELACIONAL

Según Collins, B., Fitzpatrick, B., & Pilato M. (2008), el control de versiones es el arte de administrar los cambios que sufre la información. Es una actividad cuyo objetivo primordial es facilitar, informar y mantener estructurados los avances, retrocesos y modificaciones que sufre un producto mientras es desarrollado.

Se llama control de versiones a los métodos y herramientas disponibles para controlar todo lo referente a los cambios en el tiempo de un archivo. (Borrell, 2006).

Un término importante y que sirve de apoyo para el control de versiones del esquema de base de datos relacional, son los deltas scripts, que son script en lenguaje SQL, que resultan de la comparación de dos estados de una base de datos permitiendo resumir los cambios que ocurrieron entre los dos estados y de ser necesarios aplicarlos para igualar ambos esquemas de base de datos relacional. (Downs, 2008). La Figura N° 2.5 muestra la jerarquía

de las bases de datos, la idea es que el desarrollador trabaje sobre una copia local de la base de datos, genere versiones parciales administrando scripts de cambios mínimos y cuando se mande a pruebas o se comience un nuevo ciclo de desarrollo, se genere un script de sincronización que esté accesible al resto del equipo de trabajo y un responsable haga la sincronización de los cambios con la base de datos final.

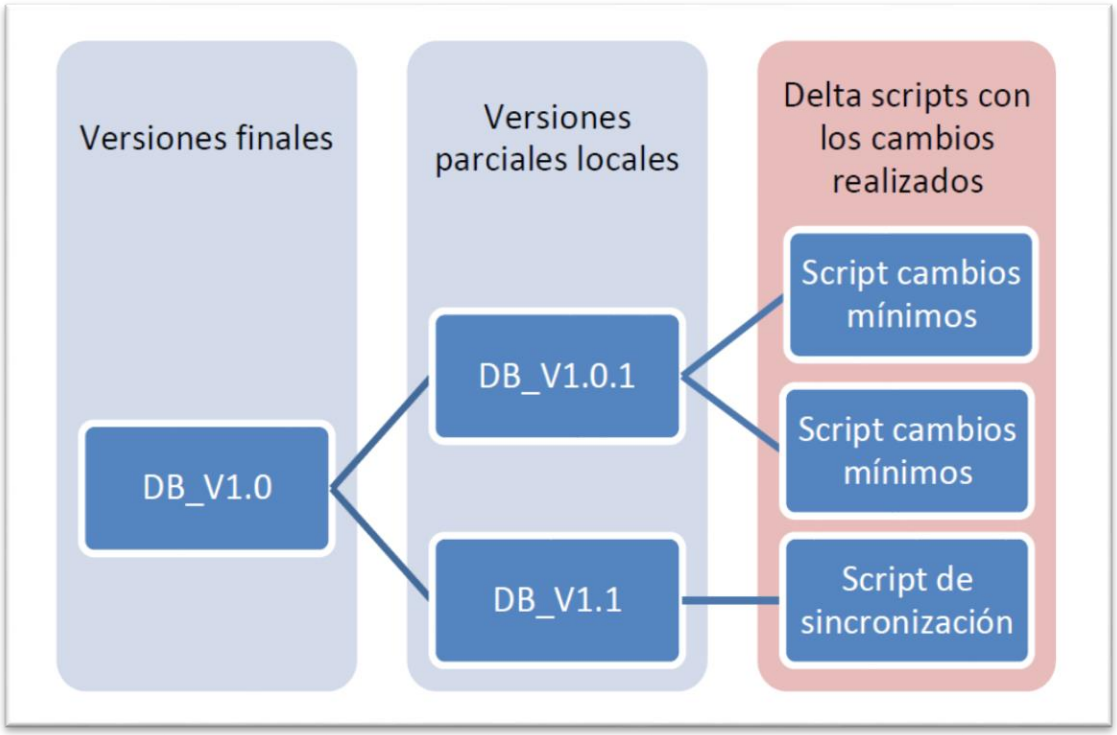



Figura N° 2.5: Jerarquía de base de datos propuesta para el control de versiones con Delta Scripts (Hayase & Matsushita, 2005).

El uso de los scripts puede variar entre scripts de cambios mínimos (pocas operaciones sobre la base de datos) o scripts de sincronización que sirvan para aplicar los cambios necesarios para que las bases de datos queden uniformes. Es decir, los scripts de *cambios* mínimos serían para uso del desarrollador y los scripts de sincronización para el proyecto. (Hayase & Matsushita, 2005).

Una base de datos es un conjunto de datos relacionados entre sí, datos que representan algún aspecto del mundo real, que pueden registrarse de manera lógica y coherente para que tengan un significado implícito y un propósito específico para un grupo de usuarios que la necesite. (Elmasri & Navathe, 2007).

### 2.2.13 IDENTIFICACIÓN DE LOS CAMBIOS

La identificación de los cambios abarca la estructura de la aplicación y los componentes individuales para hacerlos accesibles de alguna forma. Es identificar los componentes y objetos del esquema de base de datos relacional a través de su ciclo de vida y dar un seguimiento en las aplicaciones que estén relacionados. Las actividades que se incluyen en la Identificación de la configuración son (Keyes, 2004):

a) Identificación y creación de un esquema el mismo que refleje la jerarquía de la base de datos relacional.
b) Identificación de la versión del componente a ser incluido en la versión entregable.
c) Establecer las líneas base de la configuración.

### 2.2.14 CONTROL DE CAMBIOS

El control de cambios es la combinación de procesos y herramientas que ayudarán con la gestión de versiones del esquema de base de datos relacional durante el desarrollo de la aplicación. (Pressman, 2002). Durante el proceso de control de cambios del esquema de base de datos relacional se deberá tener en cuenta las siguientes terminologías:

a) **Versión**: Instancia de la base de datos relacional que es diferente en algo significativo a otras instancias del esquema de base de datos relacional.
b) **Revisión**: Modificación secundaria.

c) **Variante**: Versiones que coexisten.
d) **Release**: Versión que se distribuye a los clientes.

Según Berlack (1992), el control de cambios comprende el control del lanzamiento de una versión y los cambios del esquema de base de datos relacional a través del ciclo de desarrollo de la aplicación, es el proceso para gestionar la preparación, justificación, evaluación, coordinación, disposición, e implementación de cambios que afectarán los ítems de la configuración y la documentación. El objetivo del control de cambios es establecer mecanismos que ayudarán a asegurar la calidad en la producción del software, como también garantizar que cada versión del esquema de base de datos relacional contenga los elementos necesarios y que todos los elementos en una versión trabajen correctamente en conjunto.

Según Keyes (2014), las actividades que se incluyen en el control de cambios son:

a) Definición de procesos de cambio.
b) Mantenimiento de líneas base.
c) Establecimiento de políticas y procedimientos de control de cambios.
d) Procesamiento de cambios.
e) Control del lanzamiento de un producto.

### 2.2.15 AUDITORIA DE LOS CAMBIOS

La auditoría de los cambios viene a ser el estado final del control de versiones del esquema de base de datos relacional, en la cual se revisará y controlará que el esfuerzo empleado ha sido alcanzado satisfactoriamente con todos los requerimientos establecidos en las líneas base de la aplicación (Pressman, 2002).

Según Everett & Raymond (2007). La auditoría de cambios permite confirmar que el registro del control de versiones del esquema de base de datos relacional es completo, consistente y preciso. Así mismo, son realizadas durante el ciclo de vida de la aplicación para demostrar que los planes y procesos utilizados son adecuados, así como identificar oportunidades de mejora.

### 2.2.16 GESTIÓN DE PROYECTO ÁGIL CON SCRUM

Díaz & Dago (2008), define a SCRUM, como una colección de procesos para la gestión de proyectos, que permite centrarse en la entrega de valor para el cliente y la potenciación del equipo para lograr su máxima eficiencia, dentro de un esquema de mejora continua.

Según Palacio (2008), SCRUM es una metodología ágil de gestión de proyectos cuyo objetivo primordial es elevar al máximo la productividad de un equipo. Reduce al máximo la burocracia y actividades no orientadas a producir software que funcionen y produce resultados en periodos muy breves de tiempo. Como método, Scrum enfatiza valores y prácticas de gestión, sin pronunciarse sobre requerimientos, prácticas de desarrollo, implementación y demás cuestiones técnicas. Más bien delega completamente en el equipo la responsabilidad de decidir la mejor manera de trabajar para ser lo más productivos posibles.

Scrum define un conjunto de prácticas y roles, que pueden tomarse como punto de partida para definir el proceso de desarrollo que se ejecutará durante un proyecto. Se basa en: el desarrollo de los requisitos del proyecto en bloques temporales cortos y fijos; la priorización de los requisitos por valor para el cliente y coste de desarrollo en cada iteración; potenciación del

equipo, que se compromete a entregar unos requisitos y para ello se le otorga la autoridad necesaria para organizar su trabajo; y la sistematización de la colaboración y la comunicación tanto entre el equipo y con el cliente (Münch, 2010).

Scrum se basa en los principios del manifiesto ágiles: privilegiar el valor de la gente sobre el valor de los procesos; entregar software funcional lo más pronto posible; predisposición y respuesta al cambio; fortalecer la comunicación y la colaboración; comunicación verbal directa entre los implicados en el proyecto; simplicidad; supresión de artefactos innecesarios en la gestión del proyecto. (Kent, 2001).

Según Schwaber & Sutherland (2011), Scrum es un marco de trabajo de procesos que ha sido utilizado para gestionar el desarrollo de productos complejos desde principios de los años 90. Scrum no es un proceso o una técnica para construir productos; en lugar de eso, es un marco de trabajo dentro del cual se pueden emplear varios procesos y técnicas. Scrum hace patente la eficacia relativa de tus prácticas de gestión de producto y de desarrollo de modo que puedas mejorarlas.

Sáez (2013), da a conocer que, la palabra Scrum es un término utilizado en rugby para establecer la unión después de una interrupción. Scrum es un proceso ágil que se puede usar para gestionar y controlar desarrollos complejos de software y productos usando practicas iterativas e incrementales, así define un proceso empírico, iterativo e incremental de desarrollo que intenta obtener ventajas al respecto a los procesos definidos (cascada, espiral, prototipos, etc.) mediante la aceptación de la naturaleza caótica del desarrollo de software, y la utilización de prácticas tendientes a manejar la impredecibilidad y el riesgo a niveles aceptables. Aunque scrum

estaba previsto que fuera para la gestión de proyectos de desarrollo de software, se puede usar también para la ejecución de equipos de mantenimiento de software o como un enfoque de gestión de programas (Scrum de Scrums).

Scrum es un marco de trabajo iterativo e incremental para el desarrollo de proyectos, productos y aplicaciones. Estructura el desarrollo en ciclos de trabajo llamados Sprints. Son iteraciones de 1 a 4 semanas, y se van sucediendo una detrás de otra, Los Sprints son de duración fija, terminan en una fecha específica, aunque no se haya terminado el trabajo, nunca se alargan. Al comienzo de cada Sprint, un equipo multifuncional selecciona los elementos (Requisitos del cliente) de una lista priorizada. Se comprometen a terminar los elementos al final del Sprint. Durante el Sprint no se puede cambiar los elementos elegidos. (Kniberg, 2007).

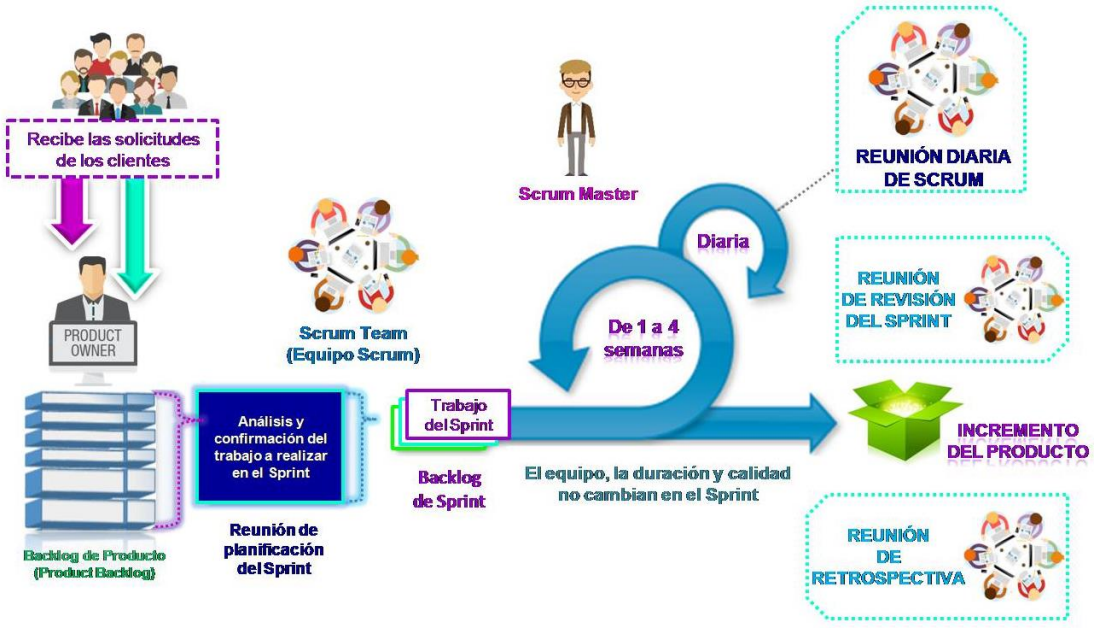



Figura N° 2.6: Proceso Scrum, Ciclo Iterativo (Deemer, Benefield, Larman, & Vodde, 2009)

### 2.2.16.1 ROLES SCRUM

Pham (2010), los roles y responsabilidades que Scrum define en un proyecto (Ver Figura N° 2.7).

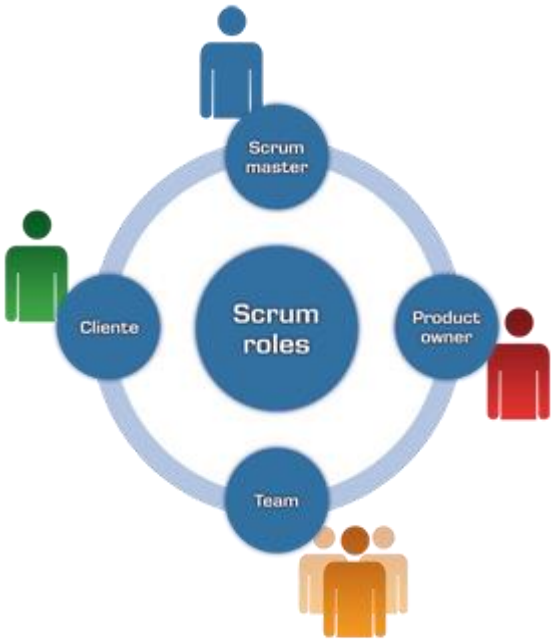



Figura N° 2.7: Roles Scrum (Pham, 2010).

#### 2.2.16.1.1 PRODUCT OWNER (PROPIETARIO DEL PRODUCTO)

Es el representante de todas las personas interesadas en el resultado del proyecto. Se encarga de la definición de los objetivos del proyecto y colabora con el equipo para planificar, revisar y dar detalle a los objetivos de cada iteración (Palacios, 2008).

El Product Owner actúa como intermediario entre el Equipo Scrum y el cliente, es responsable de asegurarse de que el equipo está trabajando en las entregas correctas en el momento adecuado (Hanley, 2015). Además de trabajar en estrecha colaboración con el Equipo Scrum en otros roles acordados, el Product Owner principalmente realiza lo siguiente:

a) Administra el Product Backlog.
b) Decide qué entregables deben ser producidos y cuando.
c) Controla el calendario de lanzamiento de los productos entregables terminados.
d) Busca cualquier aclaración de fuentes externas para el Equipo Scrum sobre el Product Backlog.
e) La comunicación entre el equipo Scrum y los clientes es un papel primordial del Product Owner. Por lo tanto, se debe realizar por alguien con habilidades de comunicación escrita y verbal. Además, la realización de esta función requiere de alguien con buenas habilidades de gestión de negociación.

#### 2.2.16.1.2 SCRUM MASTER (FACILITADOR)

Es el líder del equipo responsable de que todos los participantes sigan las reglas y los procesos de Scrum. Asegura la disponibilidad de las herramientas en cada iteración, facilita las reuniones de Scrum y está encargado de eliminar los impedimentos que el equipo tenga para continuar con su trabajo (Pham, 2010).

Scrum Master asegura que el equipo siga las mejores prácticas de Scrum y se adhieran a las normas y reglas que el equipo se compromete a seguir. Normalmente este rol cumple con las siguientes características: protege el equipo de las perturbaciones externas, entrenador, facilitador, moderador y anima al equipo constantemente como líder del grupo Scrum (Hanley, 2015).

#### 2.2.16.1.3 EQUIPO (TEAM)

Es el grupo de personas, responsable de transformar el Backlog de la iteración en un incremento de la funcionalidad del software. Tiene autoridad

para reorganizarse y definir las acciones necesarias o sugerir remoción de impedimentos: Auto-gestionado, Auto-organizado, Multi-funcional (Palacios, 2010).

La dimensión del equipo total de Scrum no debería ser superior a veinte, el número ideal es diez, más o menos dos. Si hay más, lo más recomendable es formar varios equipos. No hay una técnica oficial para coordinar equipos múltiples, pero se han documentado experiencias de hasta 800 miembros, divididos en Scrums de Scrums, definiendo un equipo central que se encarga de la coordinación, las pruebas cruzadas y la rotación de los miembros (Münch, 2010).

#### 2.2.16.1.4 CUSTUMER (CLIENTE)

Grupo interesado en el desarrollo del proyecto del cual es parte el propietario del producto (Palacios, 2008).

### 2.2.16.2 ACTIVIDADES SCRUM

Según Pham (2010), las actividades son el listado de las tareas que un equipo debe llevar a cabo dentro de una iteración, puntualmente son:

#### 2.2.16.2.1 REUNIÓN DE PLANIFICACIÓN DEL SPRINT (SPRING PLANNING MEETING)

Donde el cliente presenta al equipo la lista de requisitos, indicando la prioridad de cada uno de ellos. Además, el equipo debe realizar una planificación de cómo se realizarán y en qué momento se abordarán cada uno de los requisitos planteados, estimando tiempo y asignando tareas a cada uno de los integrantes (Pham, 2010).

Según Hanley (2015), Sprint planning es una reunión, en la que participarán el Product Owner, el Scrum Master y el equipo Scrum, con la intención de seleccionar de la lista Backlog del producto las funcionalidades sobre las que se va a trabajar y que darán valor al producto.

Antes de comenzar la reunión el Product Owner tendrá que preparar el Backlog. La reunión se realizará durante ocho horas dividido en 2 partes de 4 horas.

**Primera parte de la reunión**:

a) El equipo selecciona los ítems para transformarlos en entregables.
b) El equipo hace sugerencias, pero es el Product Owner el que decidirá si formarán parte del Sprint.
c) El equipo seleccionará el elemento a implementar de los seleccionados por el Product Owner para ese Sprint.

**Segunda parte de la reunión:**

a) El equipo hará las preguntas necesarias que tengan sobre el Product Backlog al Product Owner.
b) El equipo se encargará de encontrar la solución adecuada para transformar la parte seleccionada en una funcionalidad entregable.

El resultado de la segunda parte de la reunión es una lista denominada Sprint Backlog con las tareas, estimaciones y las asignaciones de trabajo al equipo para poder empezar a desarrollar la funcionalidad.

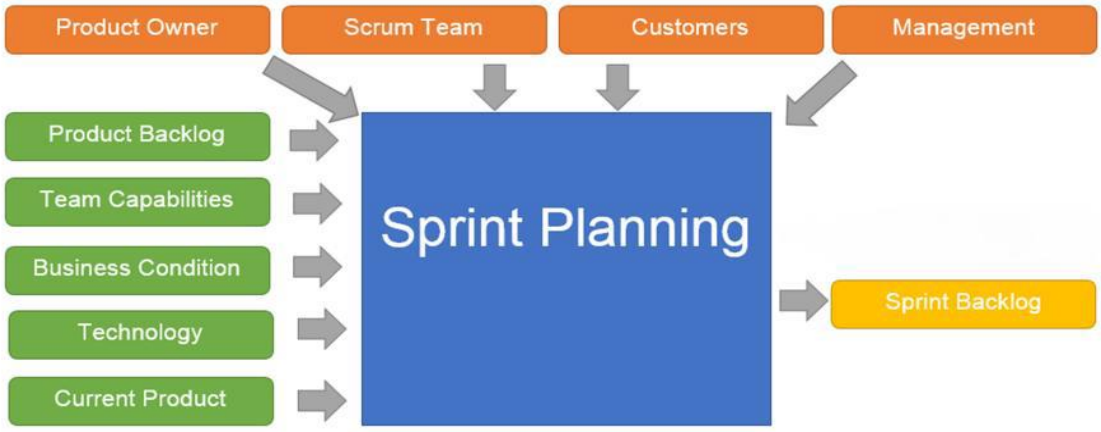



Figura N° 2.8: Entradas / Salidas de un Sprint Planning (Hanley, 2015).

**2.2.16.2.2 REUNIÓN DIARIA DE SCRUM (DAILY SCRUM)**

Reunión a diario que se debe realizar para lograr la sincronización, cada miembro inspecciona el trabajo de los demás para poder hacer las adaptaciones necesarias, actualiza el estado de la pila de iteración (Sprint Backlog) y el gráfico de trabajo pendiente (Palacios, 2008).

Segun Hanley (2015), el Equipo Scrum se reúne diariamente para gestionar su carga de trabajo y garantizar que el progreso está siendo realizado. El Scrum Master facilita los Daily Scrum y la participación de todos los miembros del equipo es obligatoria. Estas reuniones se realizan de pie y no debe exceder los 15 minutos de duración. Si hay elementos significativos durante esta sesión no deben ser ignorados debido a la falta de tiempo.

Durante el Daily Scrum, el Scrum Master revisará lo siguiente con cada miembro del equipo:

a) Lo que se llevó a cabo desde la reunión Daily Scrum anterior.

b) ¿Qué le impidió cumplir con los objetivos?
c) ¿Qué es lo que van a llevar a cabo antes de la próxima sesión?

El Sprint Backlog se actualiza como resultado de estas sesiones diarias para reflejar el estado actual del Sprint. Estas reuniones son beneficiosas para todos, puesto que se comprueba:
a) Si el equipo ha cumplido las expectativas.
b) Si el equipo ha entendido al cliente.

**2.2.16.2.3 REUNIÓN DE REVISIÓN DE LA ITERACIÓN (SPRINT REVIEW MEETING)**

Reunión informal donde el equipo presenta al cliente los requisitos completados en la iteración, de modo que se pueda apreciar un esbozo del producto final. Aquí el cliente puede replantear los objetivos y prioridades para la siguiente iteración o para algún cambio dentro del proyecto final (Pham, 2010).

Segun Hanley (2015), cada Sprint tiene la intención de culminar con la finalización de ciertas entregas, esto implica que la revisión del Sprint se llevará a cabo una vez que esas entregas se hayan completado.

Durante estas sesiones que se realizan al final de un Sprint, el Equipo Scrum demuestra todos los elementos del Sprint Backlog terminadas. El Product Owner es responsable de aceptar o rechazar cualquier artículo del Sprint Backlog. En caso de ser rechazado un elemento, el Product Owner lo eliminará del Sprint Backlog. Tales aspectos son revisados y re priorizados para su posible inclusión en Sprints posteriores.

La revisión del Sprint debe mantenerse lo más informal posible con menos énfasis en la ceremonia y más en la sustancia, tales como las decisiones que

indican el estado de elementos del Sprint Backlog: aceptada, rechazado o necesita mejoras.

#### 2.2.16.2.4 REPLANTEACIÓN DEL PROYECTO (SPRINT RETROSPECTIVE)

Mientras se está desarrollando una iteración, el cliente puede empezar a realizar cambios a los requisitos priorizados a fin de enfocar de mejor manera la siguiente iteración del proyecto, modificaciones de la manera que era conveniente y cambiando incluso el contexto del proyecto si así lo requiere (Pham, 2010).

Segun Hanley (2015), se trata de una mirada retrospectiva al Sprint recientemente terminado que tendrá una duración de máxima de 3 horas donde asistirán el Scrum Master, el Equipo Scrum y el Product Owner. La reunión Sprint Retrospective debe llevarse a cabo inmediatamente después de la finalización de un Sprint y debe centrarse en las siguientes tres cosas:

a) ¿Qué es lo que el Equipo Scrum hizo bien durante el Sprint?
b) ¿Qué es lo que no salió según lo planeado durante el Sprint?
c) ¿Qué mejoras se necesitan hacer para que las cosas pueden ir aún mejor en el próximo Sprint?

El Scrum Master facilita estas sesiones y es obligatorio para todo el Equipo Scrum asistir y proporcionar retroalimentación. El Scrum Master documenta estas sesiones y actualiza las lecciones aprendidas y mejora la documentación para futuras consultas, también toma nota de las respuestas del Equipo en un formulario. Si hay puntos que merezca atención se añadirán para el siguiente Sprint, considerándolo como un Backlog no funcional, pero de alta prioridad.

### 2.2.16.3 ARTEFACTOS SCRUM

Hanley (2015), Scrum propone ciertas herramientas llamadas Artefactos de Scrum que ayudan a los profesionales a documentar el proyecto en general. El Equipo Scrum utiliza estos artefactos como ayudas visuales para gestionar y realizar un seguimiento de los progresos que hizo en el proyecto, así como en los Sprints.

#### 2.2.16.3.1 PRODUCT BACKLOG (PILA DEL PRODUCTO)

Representa la visión y expectativas del cliente respecto a los objetivos y entregas del proyecto. El cliente es el responsable de crear y gestionar esta lista, y plasma en ella las expectativas de los resultados a obtener. Contiene los requisitos de alto nivel, el esfuerzo estimando en horas de trabajo para cada requisito, las posibles iteraciones, las entregas a realizar, y si es necesario que se considere dentro de cada requisito (Pham, 2010).

Se describe de forma reducida todos los requerimientos (Historias de usuario) priorizados y con una estimación de tiempos. También se muestra información global del desarrollo. La gestión del Product Backlog se puede llevar a cabo mediante una hoja simple de cálculo o a través de herramienta de software más completas (Sáez, 2013).

| | | | | | Estimacion de Esfuerzo | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Item** | **Elementos** | **Prioridad** | **Estimacion del Valor** | **Estimacion de Esfuerzo Inicial** | **1** | **2** | **3** | **4** | **5** | **6** |
| 1 | Como comprador quiero poner un libro en el carrito de compras | 1 | 7 | 5 | | | | | | |
| 2 | como comprador quiero quitar un libro del carrito de compras | 2 | 6 | 2 | | | | | | |
| 3 | mejorar el rendimiento del procesador de transacciones | 3 | 6 | 13 | | | | | | |
| 4 | investigar soluciones para acelerar la validacion de tarjetas de credito. | 4 | 6 | 20 | | | | | | |

Figura N° 2.9: Product Backlog (Pham, 2010).

### 2.2.16.3.2 SPRINT BACKLOG (PILA DE LA ITERACIÓN)

Elaborada en la reunión de planificación del Sprint, es el plan para completar los requisitos seleccionados para la iteración y que se compromete a demostrar al cliente al finalizar la iteración, en forma de un entregable. Permite identificar las tareas de negocio y los problemas que se presentan a lo largo de una iteración (Palacios, 2010).

Según dice Hanley (2015), en la estructura de Scrum, todas las actividades de los proyectos diseñados para ofrecer elementos del mismo como Product Backlog se realizan a través de un evento conocido como el Sprint o iteración. Los Sprints suelen limitarse a duraciones entre 1 y 4 semanas. El objetivo del sprint es crear las condiciones ideales para que el equipo de trabajo Scrum logre terminar todos los objetivos que se tenga en el Sprint Backlog.

Es un sub conjunto de la pila de productos. Un documento detallado de los requerimientos (Historias de usuario) de cada sprint. Estos se pueden subdividir en tareas (items) de 4-16 horas. El producto de software resultante debe ser potencialmente usable. Para decidir la carga máxima al equipo (trabajo máximo que puede soportar) y una puntuación a cada uno de los requerimientos. Los primeros Sprints a desarrollar suelen abordar las historias de usuario más críticos (Sáez, 2013).

| | | | | | Estimacion de Esfuerzo | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Item** | **Elemento Product Backlog** | **Sprint (Tarea)** | **Voluntario** | **Esfuerzo estimado inicial** | **1** | **2** | **3** | **4** | **5** | **6** |
| 1 | Como comprador quiero poner un libro en el carrito de compras | modificar la base de datos | | 5 | | | | | | |
| | | crear pagina web( interfaz de usuario) | | 8 | | | | | | |
| | | crear pagina web( logica java script) | | 13 | | | | | | |
| | | escribir puebas de aceptacion | | 13 | | | | | | |
| | | actualizar la pagina de ayuda del comprador | | 3 | | | | | | |
| 2 | mejorar el rendimiento del procesador de transacciones | juntar el codigo DCP y completar los test del nivel de carga | | 5 | | | | | | |
| | | completar la orden de maquina para pRank | | 8 | | | | | | |
| | | Cambiar el DCP y el lector para usar el API http de pRank | | 13 | | | | | | |

Figura N° 2.10: Sprint Backlog (Sáez, 2013).

### 2.2.16.3.3 GRÁFICOS DE TRABAJO PENDIENTE (BURNDOWN CHARTS)

Muestra la velocidad a la que se están completando los requisitos. Permite deducir si el tiempo es el requerido para la finalización del proyecto o iteración. Se puede representarlo en intervalos de días u horas pendientes para la culminación de las tareas de iteración (Münch, 2010).

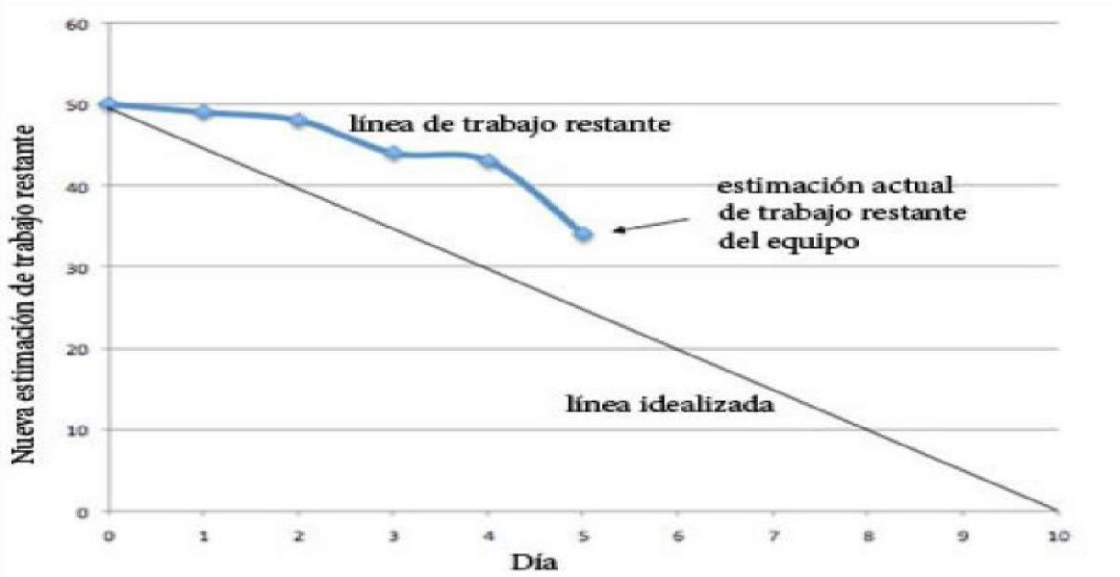



Figura N° 2.11: Gráfica de Sprint restante (Münch, 2010).

# CAPÍTULO III
# METODOLOGÍA DE LA INVESTIGACIÓN

## 3.1 TIPO Y NIVEL DE INVESTIGACIÓN

### 3.1.1 TIPO DE INVESTIGACION

Según Hernández, Fernández, & Baptista (2006), la investigación aplicada "es una investigación cuya finalidad es la resolución de problemas prácticos inmediatos en orden a transformar las condiciones del acto productivo y a mejorar la calidad del producto tomando como criterio el grado de abstracción del trabajo y para otros según el uso que se pretende dar al conocimiento".

Para Hernández (2000), la investigación aplicada "busca la aplicación o utilización de los conocimientos que se adquieren. La investigación aplicada se encuentra estrechamente vinculada con la investigación básica, pues depende de los resultados y avances de esta última".

La investigación aplicada se entiende como aquella que genera conocimientos o métodos dirigidos al sector productivo de bienes y servicios, ya sea con el fin de mejorarlo y hacerlo más eficiente, o con el fin de obtener productos nuevos y competitivos en dicho sector (Fidias, 2006).

Se desarrolló un producto software; componente software para controlar las versiones del esquema de base de datos relacional, 2017; usando la metodología ágil Scrum. La utilización práctica de los conocimientos implica buscar el conocer para hacer, para actuar, para comparar, para construir

y para modificar, permitiendo lograr un producto o servicio (Zorrilla, 1993).

Por estas consideraciones el tipo de la investigación es aplicada.

### 3.1.2 NIVEL DE INVESTIGACION

Según Hernández et al. (2006), una investigación descriptiva muestra, narra, reseña o identifica hechos, situaciones, rasgos, características de n objeto de estudio o se diseñan productos, modelos, prototipos, guías etc. Pero no se dan explicaciones o razones del porqué de las situaciones, los hechos, los fenómenos, etc.

Por estas consideraciones el nivel de investigación es descriptivo.

## 3.2 DISEÑO DE LA INVESTIGACIÓN

Según Hernández et al. (2006), el término diseño se refiere al plan o estrategia concebida para obtener la información que se desee.

Hernández et al. (2006), el diseño no experimental: se define como la investigación que se realiza sin manipular deliberadamente variables y en los que sólo se observan los fenómenos en su ambiente natural para después analizarlos.

Para Hernández et al. (2006), el diseño no experimental se divide tomando en cuenta el tiempo durante se recolectan los datos, estos son: diseño Transversal, donde se recolectan datos en un solo momento, en un tiempo único, su propósito es describir variables y su incidencia de interrelación en un momento dado, y el diseño Longitudinal, donde se recolectan datos a través del tiempo en puntos o periodos, para hacer inferencias respecto al cambio, sus determinantes y sus consecuencias.

En la investigación se observó y se generó los datos en un tiempo determinado, medimos dichos valores en un único momento sin manipular las variables, por tanto; la presente investigación es de diseño no experimental y transversal.

## 3.3 POBLACIÓN Y MUESTRA

### 3.3.1 POBLACIÓN

La población de estudio analizada, estuvo compuesta por todos los esquemas de bases de datos relacionales de AHREN Contratistas Generales SAC, 2017.

### 3.3.2 MUESTRA

Se tomó una muestra por conveniencia, no probabilística y por juicio de experto a un esquema de base de datos relacional de AHREN Contratistas Generales SAC, 2017.

## 3.4 VARIABLES E INDICADORES

### 3.4.1 DEFINICIÓN CONCEPTUAL DE LAS VARIABLES

#### 3.4.1.1 VARIABLE INDEPENDIENTE

**COMPONENTE SOFTWARE:** Es una unidad constituida por interfaces reutilizables y comportamiento específico, es un modelo de componentes o arquitecturas que puede ser desplegado y compuesto de forma independiente según un estándar para definir determinada estructura de software y funcionalidad. Software independiente que permite ser utilizada dentro de otras arquitecturas de software.

### 3.4.1.2 INDICADORES DE LA VARIABLE INDEPENDIENTE

**CAPA MODELO DE DATOS:** Es una versión extendida del modelo Entidad - Relación, que especifica el modelo conceptual de los datos usando varias técnicas de modelado. También se refiere a un conjunto de conceptos que describen un objeto del mundo real.

**CAPA CONTEXTO:** Donde se lleva a cabo las diferentes operaciones como: añadir, borrar, relacionar tablas, realizar operaciones entre columnas y guardar el estado de dichos cambios para controlar las versiones del esquema de base de datos.

### 3.4.1.3 VARIABLE DEPENDIENTE

**VERSIÓN DEL ESQUEMA DE BASE DE DATOS RELACIONAL:** Es la configuración lógica de todo o parte de un esquema de base de datos relacional. Puede existir de dos formas: como representación visual y como un conjunto de fórmulas conocidas como restricciones de integridad que controlan un esquema de base de datos relacional. Estas fórmulas se expresan en un lenguaje de definición de datos, tal como SQL.

### 3.4.1.4 INDICADORES DE LA VARIABLE DEPENDIENTE

**IDENTIFICACIÓN DEL CAMBIO:** Consiste en identificar los objetos básicos y objetos compuestos con sus características o atributos, relaciones entre objetos, evoluciones, módulos, listados, nombres de variables, entre otros sobre los cuales recaen actualizaciones o modificaciones. Siendo necesario documentarlos adecuadamente.

**CONTROL DE CAMBIOS:** Es la combinación de procedimientos humanos y las herramientas automáticas. A nivel del desarrollo de la aplicación es preocupante el control de cambio, ya que una diminuta perturbación en el código puede crear un gran fallo en el producto, pero también puede reparar un gran fallo. El cambio incontrolado lleva rápidamente al caos.

**AUDITORIA DE LOS CAMBIOS:** Es validar que la versión del esquema de la base de datos relacional sea una colección de entidades correcta y completa. Así mismo, la auditoria de los cambios ayuda al equipo de desarrollo de la aplicación a mantener un orden.

### 3.4.2 DEFINICIÓN OPERACIONAL DE LAS VARIABLES

**VARIABLE INDEPENDIENTE**

X: Componente software.

**INDICADORES VARIABLE INDEPENDIENTE**

X1: Capa modelo de datos.

X2: Capa contexto.

**VARIABLE DEPENDIENTE**

Y: Versión del esquema de base de datos relacional

**INDICADORES VARIABLE DEPENDIENTE**

Y1: Identificación de los cambios

Y2: Control de cambios

Y3: Auditoria de los cambios

**OPERACIONALIZACIÓN DE LAS VARIABLES**

Se muestra en el Anexo A.

## 3.5 TÉCNICAS E INSTRUMENTOS

### 3.5.1 TÉCNICAS PARA RECOLECTAR INFORMACIÓN

Se consideró la técnica "Análisis Documental", para documentar el componente software de control de versiones del esquema de base de datos relacional, capas de modelo de datos, capa de contexto, metodologías y tecnologías que permiten la manipulación de base de datos relacional.

Se consideró la técnica de "Observación", para realizar el análisis de la capa modelo de datos, capa de contexto, metodologías y tecnologías que permitan la manipulación de la base de datos relacional para construir el componente software de control de versiones del esquema de base de datos relacional.

### 3.5.2 INSTRUMENTOS PARA RECOLECTAR INFORMACIÓN

Se ha diseñado el instrumento "Ficha Análisis Documental" (Ver Anexo B), para el estudio del componente software de control de versiones del esquema de base de datos relacional que permita implementar la capa modelo de datos y la capa contexto para la manipulación de la base de datos relacional desde la aplicación.

Se ha diseñado el instrumento "Ficha Observación" (Ver Anexo C), para la técnica "Observación". Muestra la información de la capa modelo de datos y la capa de contexto para la implementación del componente software de control de versiones del esquema de base de datos relacional que permita la manipulación de la base de datos relacional.

### 3.5.3 HERRAMIENTAS PARA EL TRATAMIENTO DE DATOS

Las herramientas tecnológicas que se utilizan son seleccionadas debido a la interoperabilidad que presenta el componente software, base de datos relacional; se propone la implementación de un componente software de control de versiones del esquema de base de datos relacional, para la construcción de nuevas aplicaciones. Por lo cual seleccionamos las tecnologías según la tabla 3.1.

| NOMBRE | FABRICANTE | SERVICIO |
|---|---|---|
| WINDOWS 10 | Microsoft Corporation | Es la versión más reciente de Microsoft Windows, línea de sistemas operativos producida por Microsoft Corporation. |
| VISUAL STUDIO 2017 | Microsoft Corporation | Es un entono integrado de desarrollo para Sistemas Operativos Windows. Soporta varios lenguajes de programación como C#, Visual Basic, .NET etc. |
| C# | Microsoft Corporation | Es un lenguaje de programación orientado a objetos diseñado para la infraestructura de lenguaje común. |
| SQLITE | Richard Hipp | SQLite es un sistema de gestión de bases de datos relacional compatible con ACID, escrita en C. |
| .NET CORE | Microsoft Corporation | ASP.NET Core es un nuevo framework de código abierto y multiplataforma para la creación de aplicaciones modernas conectadas a Internet, como aplicaciones web y APIs Web. |

Tabla N° 3.1: Herramientas para el desarrollo del componente software de control de versiones del esquema de base de datos relacional.

### 3.5.4 TÉCNICAS PARA APLICAR SCRUM

Para el presente trabajo de investigación, utilizamos la metodología ágil SCRUM descrita con base teórica en el capítulo II, sección 2.2.16, la cual se puede resumir en la siguiente figura.

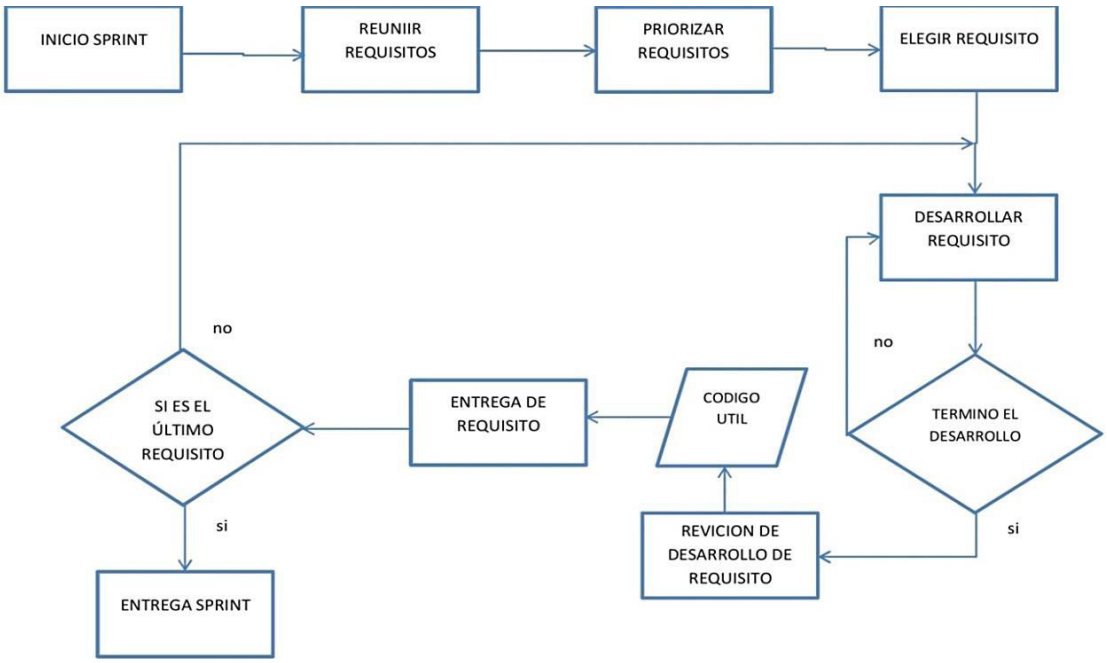



Figura N° 3.1: Flujograma de la metodología Scrum (Elaboración propia).

Adicionalmente revisando el marco teórico en el capítulo II, sección 2.2.16, formulamos el proceso, que considera las fases para construir aplicaciones, usando Scrum donde dictamina la utilización de tres artefactos esenciales en un proyecto: la pila del proyecto, con los requerimientos totales del aplicativo a desarrollar; la pila de iteración, que determina los requerimientos a desarrollar en cada iteración; y la gráfica de trabajo pendiente, que representa el trabajo que falta por hacer en un proyecto en el tiempo (Laura, 2006), como se muestra en las tablas N° 3.2 y 3.3.

| TAREA | ARTEFACTO | TÉCNICA | RESPONSABLE |
|---|---|---|---|
| Asignar roles primarios y secundarios | Propietarios del proyecto | Identificar los responsables de los roles primarios y secundarios | Propietario del Producto y equipo |
| Elaboración de la pila del proyecto | Matriz de la pila del Proyecto | Revisión de requerimientos y elaboración de la matriz o pila de Iteraciones | Equipo |
| Definir la cantidad de iteraciones | Integrar las iteraciones a la pila del proyecto | Clasificar las tareas de la pila del proyecto | Equipo |
| Asignar prioridades del proyecto | Integrar las prioridades a la pila del proyecto | Evaluar y asignar prioridades a todas las tareas de la pila del proyecto. | Equipo |
| Asignar horas esfuerzo | Integrar las horas esfuerzo a la pila del proyecto | Evaluar y estimar las horas esfuerzo para cada actividad de la pila del proyecto | Equipo |
| Elaborar la matriz de la pila del proyecto | Pila del proyecto integrado | Matriz o pila de proyecto concluido | Equipo |

Tabla N° 3.2: Entregables y sub-entregables de artefactos Scrum (Laura, 2006).

| TAREA | ARTEFACTO | TÉCNICA | RESPONSABLE |
|---|---|---|---|
| Reunión de planificación del Sprint. | Pila de iteración aprobada por ítem | Reunión de planificación, distribución de tareas | Propietario, Facilitador y equipo |
| Elaborar la pila de iteración primaria | Pila de iteración primaria | Identificación de las tareas en cada ítem. | Facilitador y equipo |
| Revisión de la pila de iteración | Pila de iteración actualizada | Revisión del avance y recálculo del esfuerzo. | Equipo |
| Mostrar resultados para cada tarea. | Capturas por cada iteración | Mostrar los avances capturadas | Equipo |
| Realizar el gráfico | Gráfica de tarea pendiente de la primera iteración | Realizados para elaborar el gráfico de trabajo pendiente. | Equipo |
| Realizar la reunión de aprobación | Aprobación de la primera pila de iteración | Reunión de aprobación | Propietario, facilitador y equipo |
| Pasar a la siguiente iteración | Elaborar las matrices de la siguiente iteración | Revisar faces de desarrollo de software y UML | Facilitador y equipo |

Tabla N° 3.3: Entregables de Scrum en el ciclo de desarrollo (Guzmán et al., 2012).

# CAPÍTULO IV

# ANÁLISIS RESULTADOS DE LA INVESTIGACIÓN

## 4.1 RESULTADOS DE LA RECOLECCIÓN DE DATOS

Los análisis de los resultados obtenidos son desde foros y páginas de internet en donde se puede observar que, la mayoría de los desarrolladores usan scripts incrementales para realizar las operaciones en la base de datos relacional tales como crear columnas, eliminar columnas, entre otras operaciones, de tal modo podemos observar que otra mayoría utiliza SCV, pero esta tecnología no está diseñada para controlar las versiones del esquema de base de datos, se sabe que SCV se utiliza para controlar las versiones del código fuente (Ver Figura N° 4.1).

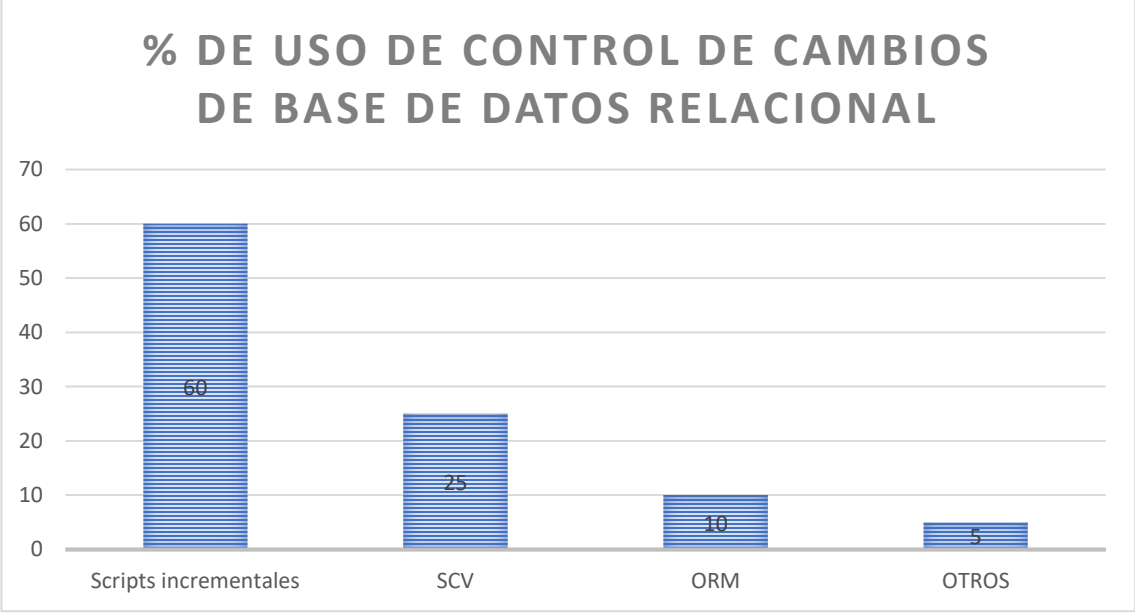



Figura N°4.1: Métodos usadas para el control de versiones del esquema de base de datos relacional (Elaboración propia).

La mayoría afirma que usar scripts incrementales es difícil en sentido que administrarlo y seguir un orden para realizar los cambios es muy tedioso (Ver Figura N° 4.2).

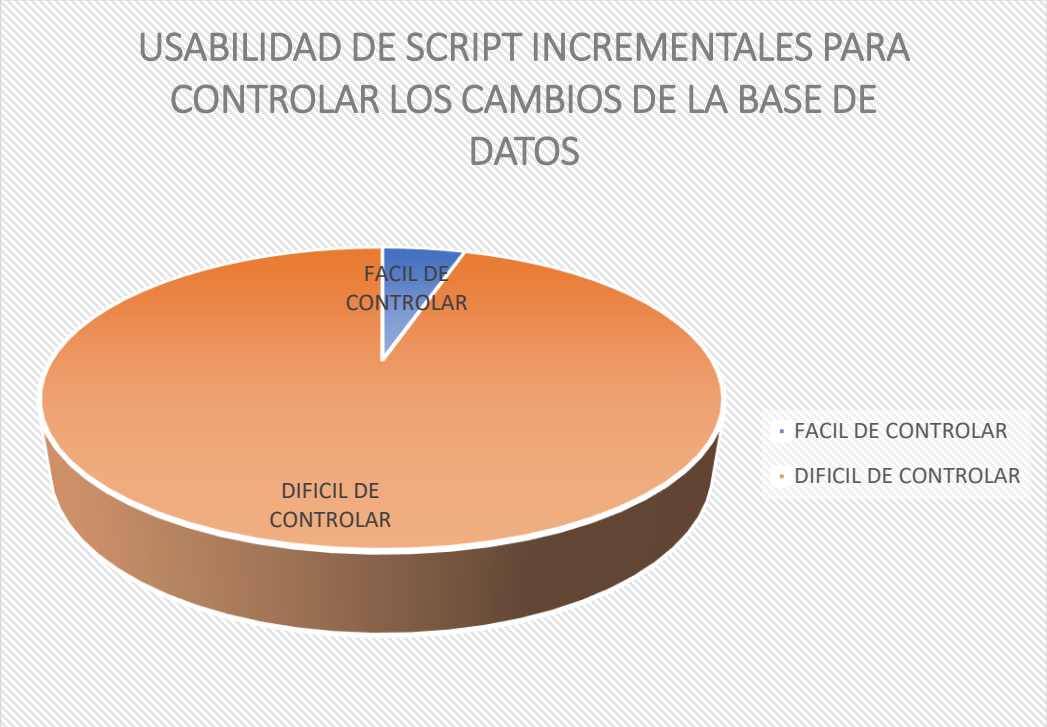



Figura N°4.2: Usabilidad de scripts incrementales para el control de versiones del esquema de base de datos relacional (Elaboración propia).

## 4.2 RESULTADOS DE LOS ARTEFACTOS DE LA METODOLOGÍA

Según las fases del desarrollo de componentes software usando el marco de trabajo Scrum, desarrolladas en las tablas 3.2 y 3.3 del capítulo III sección 3.5.4, descrito en el capítulo II sección 2.2.16, se diseña las características, roles y artefactos Scrum con el análisis y diseño de la pila de iteración, implementación, y prueba. A continuación, mostramos los resultados de los artefactos de acuerdo al proceso Scrum.

### 4.2.1 ROLES

#### 4.2.1.1 ROLES PRINCIPALES

En el grupo tenemos los roles que están comprometidos enteramente con el desarrollo del componente software para lograr el funcionamiento del aplicativo.

**PRODUCT OWNER (PROPIETARIO DEL PRODUCTO)**

El Representante de la gerencia de obras. A quien no se nombrará en el presente trabajo de investigación por políticas de seguridad de la empresa en cuestión, representa al propietario del producto, éste representa a todos los interesados con el proyecto, verificando continuamente el avance del mismo, evaluando los diferentes logros alcanzados al final de cada iteración y tomando decisiones respecto a las futuras iteraciones, de tal modo que el componente software que controla las versiones del esquema de base de datos relacional, sea acorde a las expectativas de la gerencia de obras.

**SCRUM MASTER (FACILITADOR)**

El Representante de la gerencia de innovación. A quien no se nombrará en el presente trabajo de investigación por políticas de seguridad de la empresa en cuestión, representa al SCRUM MASTER responsable de la correcta dirección de los avances del proyecto. Durante cada iteración se comprometió a verificar la disponibilidad de todos los elementos que requiera el equipo para su correcto desarrollo del proyecto. Además, actúa como intermediario para facilitar una correcta comunicación entre el equipo de trabajo y el propietario del producto.

**EQUIPO (TEAM)**

**Responsable**: Bach. Alex Palomino Pariona.

Encargado del desarrollo del aplicativo en cada una de las iteraciones. Comprometido con el correcto funcionamiento del producto entregado al final de cada iteración. En concordancia con las buenas prácticas de Scrum, realiza el análisis, diseño, implementación, evaluación y puesta en marcha del componente software que controle las versiones del esquema de base de datos relacional.

#### 4.2.1.2 ROLES SECUNDARIOS

Los roles externos, dan un mayor grado de satisfacción y eficiencia del componente software de control de versiones del esquema de base de datos relacional que va a ser entregado y usado en las diferentes obras.

**CUSTOMER (CLIENTE)**

El cliente representa una generalización mayor del propietario del producto. Dentro del presente proyecto es el responsable del área al que será entregado el componente software de control de versiones del esquema de base de datos relacional completo para su posterior uso y puesta en marcha en las diferentes obras de la empresa.

### 4.2.2 ARTEFACTOS SCRUM

Scrum dictamina la utilización de 3 artefactos esenciales en un proyecto: la pila del proyecto, con los requisitos totales del aplicativo a desarrollar; la pila de la iteración, que determina los requisitos a desarrollar en cada iteración; y la gráfica de trabajo pendiente, que representa el trabajo que falta por hacer en un proyecto en el tiempo.

#### 4.2.2.1 PILA DEL PROYECTO

Para determinar la "pila de productos" se requiere una reunión entre el "Scrum Master" y el "dueño del producto", en esta reunión el Dueño del

Producto explica a detalle lo que desea que haga el componente, poco a poco el Scrum Master interioriza la idea y ayuda al dueño del producto a pulir la necesidad.

El dueño del producto está en todo el derecho de listar todo lo que crea por conveniente para formar parte de la pila de producto; por su lado el Scrum Master con la experiencia obtenida en anteriores proyectos busca un nivel de equilibrio entre el lenguaje del Dueño de Producto y el lenguaje del Equipo(Team).

Como resultado de esta reunión de aproximadamente 5 horas se obtiene una primera lista de elementos que posteriormente se convierten en la "Pila de Producto" (Tabla N° 4.1).

| ÍTEM | DESCRIPCIÓN |
|---|---|
| 1 | Instalación de herramientas tecnológicas. |
| 2 | Construir el proyecto. |
| 3 | Diseñar la arquitectura. |
| 4 | Instalación de dependencias. |
| 5 | Configuración del proyecto. |
| 6 | Configuración de modelo de datos. |
| 7 | Configuración del contexto de proyecto. |
| 8 | Configuración de providers |
| 9 | Testear el componente software. |
| 10 | Elaborar documentación del componente software. |

Tabla N°4.1: Lista de elementos (Elaboración Propia).

El dueño del producto tiene el deber de dar prioridades a cada elemento de la pila de producto, de tal manera que la lista quede de forma ascendente por prioridad.

Esto es normalmente una práctica desconocida para el Dueño del Producto, por esa razón, el Scrum Master enseña al Dueño del Producto a hacerla, esta reunión dura aproximadamente 1 hora, como resultado se obtiene la lista de elementos priorizados. (Tabla N° 4.2).

| ÍTEM | DESCRIPCIÓN | PRIORIDAD |
|---|---|---|
| 1 | Construir el proyecto. | 1 |
| 2 | Diseñar la arquitectura. | 2 |
| 3 | Instalación de dependencias. | 3 |
| 4 | Configuración del proyecto. | 4 |
| 5 | Configuración de modelo de datos. | 5 |
| 6 | Configuración del contexto de proyecto. | 6 |
| 7 | Configuración de providers | 7 |
| 8 | Testear el componente software. | 8 |
| 10 | Elaborar documentación del componente software. | 9 |
| 11 | Instalación de herramientas tecnológicas. | 10 |

Tabla N°4.2: Asignación de prioridades (Elaboración Propia).

El siguiente paso es asignar un valor de esfuerzo, habilidad de los desarrollares, estimado a cada elemento, esto es deber del "Equipo", Scrum especifica muchas maneras de cómo se hace este cálculo, en la investigación se utilizó el método póker para estimar el esfuerzo por cada elemento ver (Tabla N° 4.3).

| DESCRIPCIÓN | PERSONAS | ESFUERZO INICIAL |
|---|---|---|
| Construir el proyecto. | 1 | 5.5 |
| Diseñar la arquitectura. | 1 | 3 |
| Instalación de dependencias. | 1 | 1 |
| Configuración del proyecto. | 1 | 7.5 |
| Configuración de modelo de datos. | 1 | 7 |
| Configuración del contexto de proyecto. | 1 | 8 |
| Testear el componente software. | 1 | 4 |
| Elaborar documentación del componente software. | 1 | 2 |
| Instalación de herramientas tecnológicas. | 1 | 2 |

Tabla N°4.3: Estimación de esfuerzo inicial (Elaboración propia).

Estas estimaciones de esfuerzo se pueden actualizar en cada sprint a medida que se aprenden cosas nuevas; por tanto, es una actividad continua de re-priorización de la pila de producto que evoluciona constantemente. Entonces después de este proceso tenemos la "pila de producto" lista para iniciar el primer Sprint ver (Tabla N° 4.4).

| DESCRIPCIÓN | PRIORIDAD | ESFUERZO INICIAL (DIAS) |
|---|---|---|
| Construir el proyecto. | 1 | 5 |
| Diseñar la arquitectura. | 2 | 8 |
| Instalación de dependencias. | 3 | 3 |
| Configuración del proyecto. | 4 | 12 |
| Configuración de modelo de datos. | 5 | 7 |
| Configuración del contexto de proyecto. | 6 | 7 |
| Testear el componente software. | 7 | 4 |

| | | |
|---|---|---|
| Generar el instalador del componente software. | 8 | 2 |
| Elaborar manual de usuario del componente software. | 9 | 2 |
| Instalación de herramientas tecnológicas. | 10 | 2 |

Tabla N°4.4: Pila de Producto (Elaboración propia).

Los elementos de la pila de producto pueden variar significativamente en tamaño y esfuerzo. Los elementos grandes se parten en elementos más pequeños durante el taller de refinamiento de la pila de producto o la reunión de planificación del sprint, y los elementos pequeños pueden ser consolidados.

#### 4.2.2.2 ITERACIONES (REUNIÓN DE PLANIFICACIÓN DEL SPRINT)

##### 4.2.2.2.1 SPRINT BACKLOG 01

En esta reunión el equipo y el dueño del producto revisan los elementos de alta prioridad para definir la cantidad de elementos que se desarrollan en el primer sprint, el equipo estima en función al esfuerzo y conocimiento del tema y se compromete entregar los siguientes elementos, que son los de más alta prioridad ver (Tabla N° 4.5).

| Elemento | Prioridad |
|---|---|
| Construir el proyecto. | 1 |
| Diseñar la arquitectura. | 2 |

Tabla N°4.5: Elementos para el primer Sprint (Elaboración propia).

Sin embargo, existe en la pila de productos un elemento de muy baja prioridad que encaja perfectamente en este Sprint y como el "Equipo" en Scrum también tiene autoridad para seleccionar elementos de menor

prioridad para incluirlos en el siguiente Sprint, se incluye el elemento ver (Tabla N° 4.6).

| Elemento | Prioridad |
|---|---|
| Instalación de herramientas tecnológicas. | 10 |

Tabla N°4.6: Elemento agregado al primer Sprint (Elaboración propia).

Esta es una práctica muy frecuente en la metodología Scrum, de hecho, una de sus principales características es su adaptabilidad.

La reunión del primer sprint tuvo una duración aproximada de 2 horas, donde clarifican los objetos y el contexto de cada elemento. A continuación, se realizan la planificación detallada de las tareas para saber cómo implementar los elementos seleccionados en el Sprint.

En esta etapa del proceso el "Equipo" estima cuánto tiempo tiene cada miembro del equipo para trabajar en el Sprint, en otras palabras, su día laboral medio menos el tiempo que pasan en reuniones, leyendo el correo, comiendo, etc. Para la mayoría de la gente esto significa 4-7 horas de tiempo disponible al día para el trabajo exclusivo en el desarrollo del Sprint ver (Tabla N° 4.7).

| Longitud del Sprint | | | 3 semanas |
|---|---|---|---|
| Días laborables durante el Sprint | | | 15 días |
| Miembro del equipo | Días disponibles en el sprint | Horas disponibles por día | Total horas disponibles |
| Alex Palomino Pariona | 15 | 7 | 105 |

Tabla N°4.7: Estimación de tiempo disponible para Sprint (Elaboración propia).

El siguiente paso es dividir cada elemento en tareas individuales, que se guardan en un documento llamado pila del producto ver (Figura N° 4.3).

| Elemento de la pila de producto | Tarea del Sprint | Voluntario | Esfuerzo estimado inicial | Nuevo esfuerzo estimado al final del día | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | D10 | D11 | D12 | D13 | D14 | D15 |
| Instalación de herramientas tecnológicas | Seleccionar las herramientas tecnológicas. | APP | 1 | | | | | | | | | | | | | | | |
| | Instalar y configurar las herramientas tecnológicas. | APP | 1 | | | | | | | | | | | | | | | |
| Construir el proyecto | Concebir el nuevo proyecto. | APP | 1 | | | | | | | | | | | | | | | |
| | Identificar los objetivos del negocio. | APP | 2 | | | | | | | | | | | | | | | |
| | Prueba de funcionalidad. | APP | 2 | | | | | | | | | | | | | | | |
| Diseñar la arquitectura. | Análisis patrones arquitectónicos. | APP | 2 | | | | | | | | | | | | | | | |
| | Identificar compontes del modelo arquitectónico. | APP | 2 | | | | | | | | | | | | | | | |
| | Diseñar un nuevo modelo arquitectónico. | APP | 2 | | | | | | | | | | | | | | | |
| | Formalizar y generar el documento de diseño. | APP | 2 | | | | | | | | | | | | | | | |
| | | REAL | 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | DESEADO | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Figura N°4.3: Pila del primer Sprint (Elaboración propia).

Otra de las prácticas claves de Scrum una vez empezado el Sprint es el Scrum Diario.

Es una reunión corta (10 minutos en este Sprint) que se celebra todos los días a una hora prefijada, el equipo asiste a la reunión de pie para hacerla corta.

Es la oportunidad del equipo de informar a los demás sobre el progreso y los obstáculos. En el Scrum Diario, cada miembro del equipo, uno por uno, informa sobre tres cosas a los otros miembros del equipo:

a) Que han hecho desde la última reunión.
b) Que tienen planificado hacer antes de la siguiente reunión.
c) Cualquier bloqueo o impedimento que tengan.

En este sprint durante las primeras 10 días se logró estar debajo de la curva deseada, esto quiere decir que la estimación del esfuerzo estuvo bien, ver (Figura N° 4.4 y Figura N° 4.5).

| Elemento de la pila de producto | Tarea del Sprint | Voluntario | Esfuerzo estimado inicial | Nuevo esfuerzo estimado al final del día | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | D10 | D11 | D12 | D13 | D14 | D15 |
| Instalación de herramientas tecnológicas | Seleccionar las herramientas tecnológicas. | APP | 1 | 1 | | | | | | | | | | | | | | |
| | Instalar y configurar las herramientas tecnológicas. | APP | 1 | 1 | | | | | | | | | | | | | | |
| Construir el proyecto | Concebir el nuevo proyecto. | APP | 1 | | 1 | | | | | | | | | | | | | |
| | Identificar los objetivos del negocio. | APP | 2 | | | 1 | 1 | | | | | | | | | | | |
| | Prueba de funcionalidad. | APP | 2 | | | | | 2 | | | | | | | | | | |
| Diseñar la arquitectura. | Análisis patrones arquitectónicos. | APP | 2 | | | | | | 1 | 0.5 | 0.5 | | | | | | | |
| | Identificar compontes del modelo arquitectónico. | APP | 2 | | | | | | | | | 0.5 | 0.5 | | | | | |
| | Diseñar un nuevo modelo arquitectónico. | APP | 2 | | | | | | | | | | | | | | | |
| | Formalizar y generar el documento de diseño. | APP | 2 | | | | | | | | | | | | | | | |
| | | REAL | 15 | 13 | 12 | 11 | 10 | 8 | 7 | 6.5 | 6 | 5.5 | 5 | 0 | 0 | 0 | 0 | 0 |
| | | DESEADO | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Figura N°4.4: Pila del primer Sprint avance 10 días (Elaboración propia).

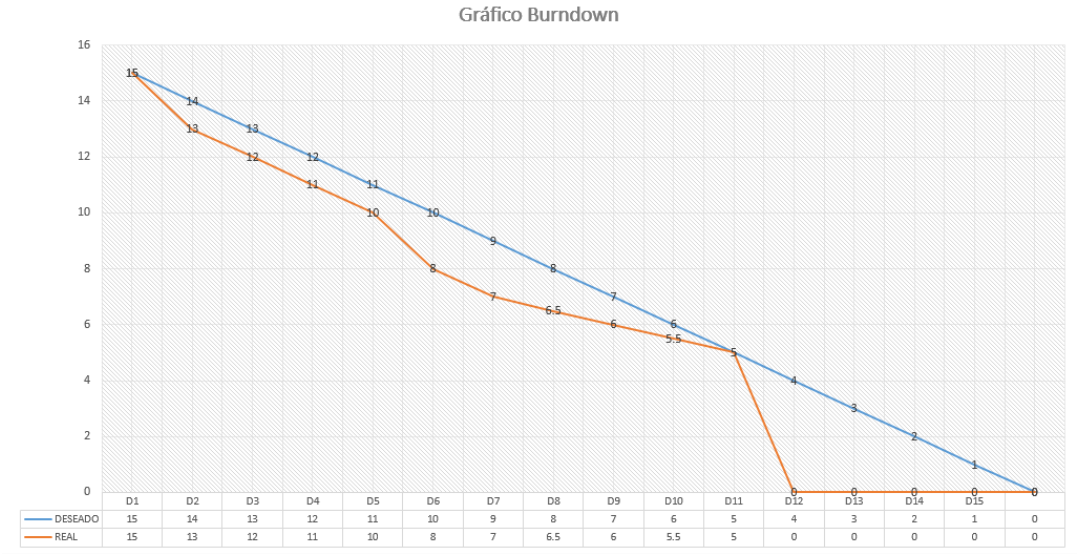



Figura N°4.5: Gráfico burndown avance 10 días (Elaboración propia).

En el sprint después del día 10 se tuvo algunos problemas para completar las tareas, pero oportunamente se solucionaron, de esta manera se culminó el sprint en el tiempo acordado con el product onwer, ver (Figura N° 4.6 y Figura N° 4.7).

| Elemento de la pila de producto | Tarea del Sprint | Voluntario | Esfuerzo estimado inicial | Nuevo esfuerzo estimado al final del día | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | D10 | D11 | D12 | D13 | D14 | D15 |
| Instalación de herramientas tecnológicas | Seleccionar las herramientas tecnológicas. | APP | 1 | 1 | | | | | | | | | | | | | | |
| | Instalar y configurar las herramientas tecnológicas. | APP | 1 | 1 | | | | | | | | | | | | | | |
| Construir el proyecto | Concebir el nuevo proyecto. | APP | 1 | | 1 | | | | | | | | | | | | | |
| | Identificar los objetivos del negocio. | APP | 2 | | | 1 | 1 | | | | | | | | | | | |
| | Prueba de funcionalidad. | APP | 2 | | | | | 2 | | | | | | | | | | |
| Diseñar la arquitectura. | Análisis patrones arquitectónicos. | APP | 2 | | | | | | 1 | 0.5 | 0.5 | | | | | | | |
| | Identificar compontes del modelo arquitectónico. | APP | 2 | | | | | | | | | 0.5 | 0.5 | 1 | | | | |
| | Diseñar un nuevo modelo arquitectónico. | APP | 2 | | | | | | | | | | | | 0.5 | 0.5 | 1 | |
| | Formalizar y generar el documento de diseño. | APP | 2 | | | | | | | | | | | | | | | 2 |
| | | REAL | 15 | 13 | 12 | 11 | 10 | 8 | 7 | 6.5 | 6 | 5.5 | 5 | 4 | 3.5 | 3 | 2 | 0 |
| | | DESEADO | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Figura N°4.6: Primer Sprint completado al 100% (Elaboración propia).

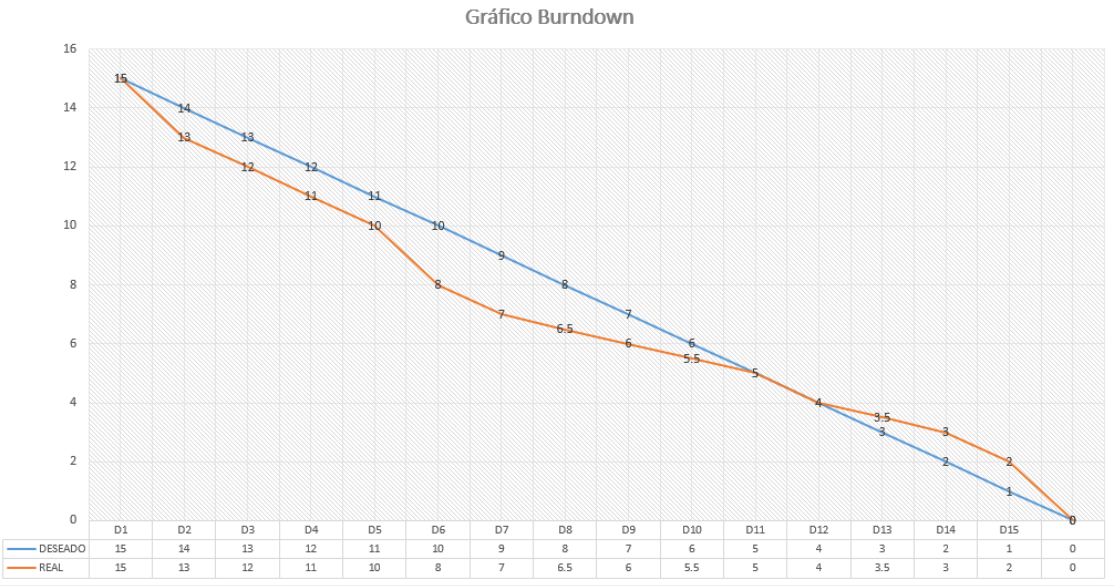



Figura N°4.7: Gráfico burndown al 100% (Elaboración propia).

En este Sprint no hubo ningún bloqueo o impedimento durante los 15 días de duración y los informes diarios respecto a su avance permitió optimizar tiempo y recursos.

#### 4.2.2.2.2 RESULTADOS DEL SPRINT 01

Elaboración de la estructura del proyecto principal para probar le componente software, ver (Figura N° 4.8).

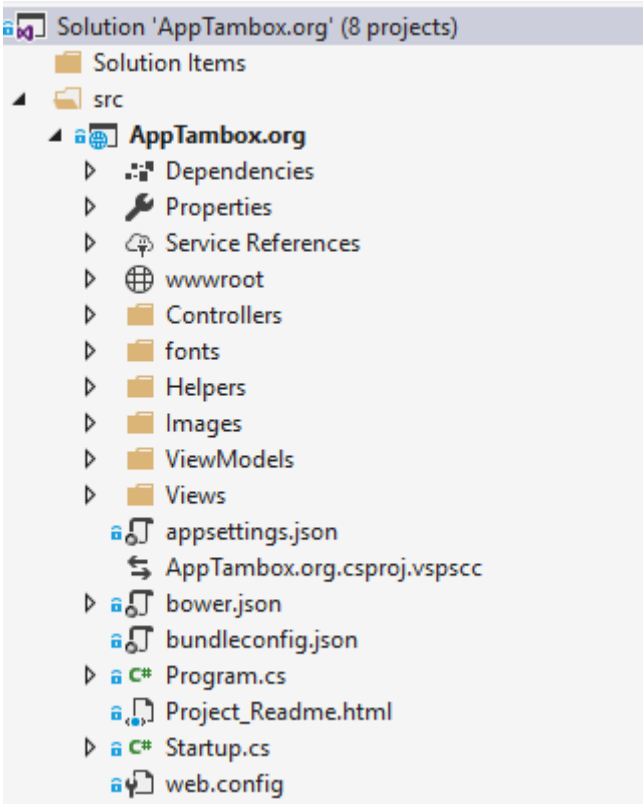



Figura N°4.8: Creación del proyecto (Elaboración propia).

Elaboración de la estructura completa (componentes de la arquitectura) del proyecto para probar el componente software, ver (Figura N° 4.9).

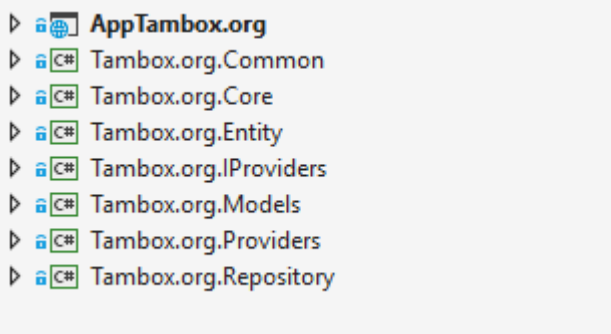



Figura N°4.9: Componentes de la arquitectura (Elaboración propia).

Vista General del proyecto principal (identificación de objetos del negocio), ver (Figura N° 4.10)

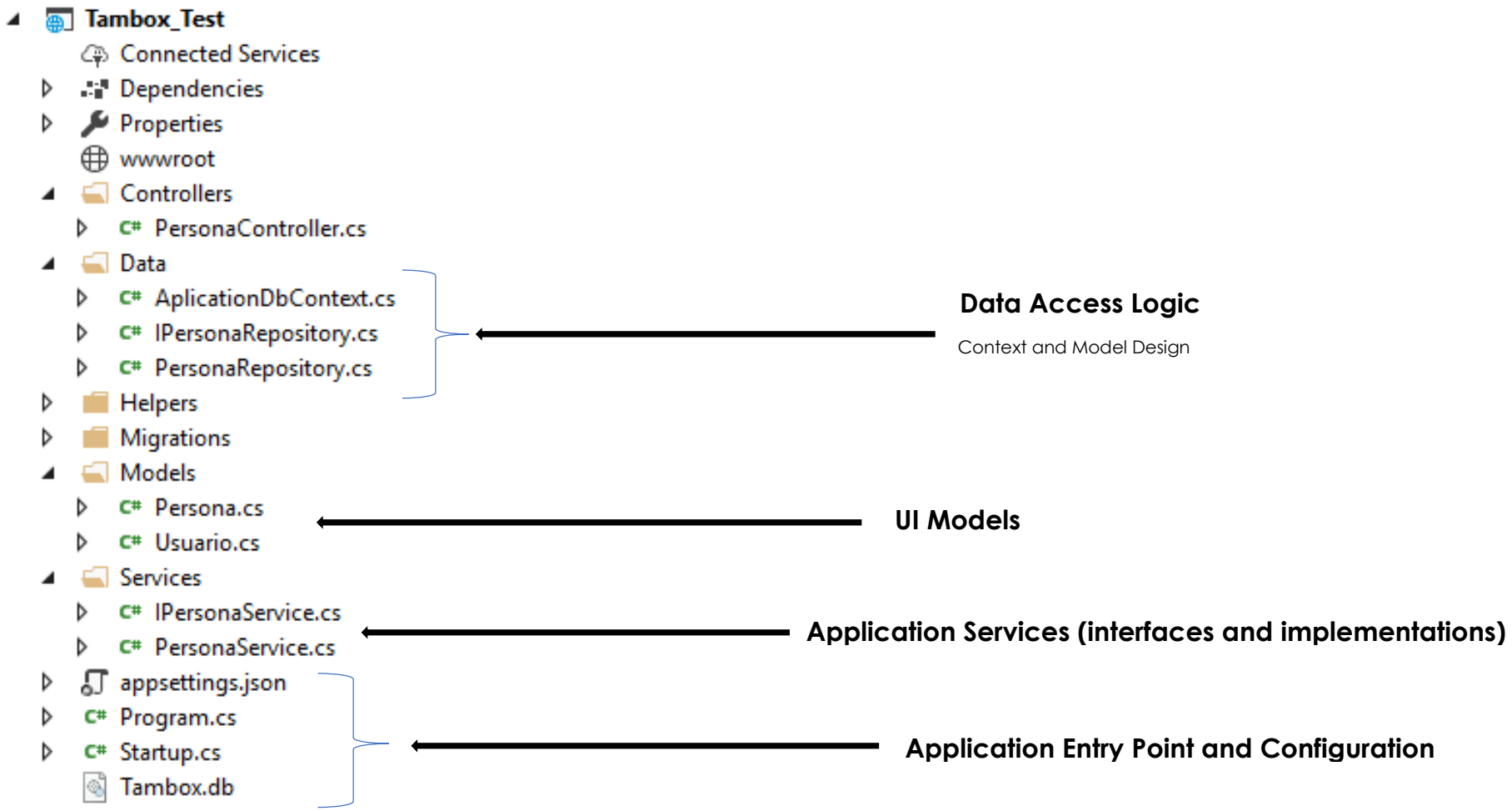



Figura N°4.10: Identificación de objetos del negocio (Elaboración propia).

#### 4.2.2.2.3 SPRINT BACKLOG 02

En esta reunión el equipo y el dueño del negocio revisan los siguientes elementos de alta prioridad que quedan en la pila del producto para definir la cantidad de elementos que se desarrollan en el segundo sprint, el equipo estima en función al esfuerzo y conocimiento del tema y se comprometen a entregar los siguientes elementos, ver (Tabla N° 4.8).

| Elemento | Prioridad |
|---|---|
| Instalación de dependencias. | 3 |
| Configuración del proyecto. | 4 |

Tabla N°4.8: Elementos para el segundo Sprint (Elaboración propia).

La reunión del segundo sprint tuvo una duración de aproximadamente 2 horas siendo de suma importancia lograr entender la lógica del dueño de producto en cómo se debe configurar los diferentes escenarios en el proyecto así mismo indica que el proyecto no debe tener ninguna dependencia sin resolver, ver (Tabla N° 4.9 y la Figura N° 4.11).

| Longitud del Sprint | | | 3 semanas |
|---|---|---|---|
| Días laborables durante el Sprint | | | 15 días |
| Miembro del equipo | Días disponibles durante el sprint | Horas disponibles por día | Total horas disponibles |
| Alex Palomino Pariona | 15 | 7 | 105 |

Tabla N°4.9: Estimación de tiempo disponible para Sprint (Elaboración propia).

| Elemento de la pila de producto | Tarea del Sprint | Voluntario | Esfuerzo estimado inicial | Nuevo esfuerzo estimado al final del día | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | D10 | D11 | D12 | D13 | D14 | D15 |
| Instalación de dependencias. | Instalación de SDK para NetCore | APP | 0.5 | | | | | | | | | | | | | | | |
| | Instalación de otras dependencias adicionales | APP | 0.5 | | | | | | | | | | | | | | | |
| Configuración del proyecto. | Definir parametros de configuración | APP | 2 | | | | | | | | | | | | | | | |
| | Configuración de inyección de dependencia | APP | 1 | | | | | | | | | | | | | | | |
| | Configuración de DomainResolver | APP | 2 | | | | | | | | | | | | | | | |
| | Configuración de aspectos globales del proyecto | APP | 2 | | | | | | | | | | | | | | | |
| Configuración de modelo de datos | Creación de clases entidad | APP | 2 | | | | | | | | | | | | | | | |
| | Creacion de clases modelo | APP | 2 | | | | | | | | | | | | | | | |
| | Creación de controlador de versiones de DB | APP | 3 | | | | | | | | | | | | | | | |
| | | REAL | 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | DESEADO | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Figura N°4.11: Pila del segundo Sprint (Elaboración propia).

Como ya se mencionó anteriormente, cada miembro del equipo, informa sobre tres cosas a los otros miembros del equipo:

a) Que han hecho desde la última reunión.
b) Que tienen planificado hacer antes de la siguiente reunión.
c) Cualquier bloqueo o impedimento que tengan.

En el segundo sprint, durante los 4 primeros días el esfuerzo estimado estuvo acorde a las capacidad y experiencia del desarrollador, por lo tanto, podemos decir que la estimación fue correcta (Figura N° 4.12 y Figura N° 4.13), desde el día 5 hasta el día 10, se logró realizar las tareas del sprint en menos tiempo de lo estimado (Figura N° 4.14 y Figura N° 4.15), después del día 10 ocurrieron algunos problemas para terminar el sprint con un retraso total de 1.5 días (Figura N° 4.16 y Figura N° 4.17).

| Elemento de la pila de producto | Tarea del Sprint | Voluntario | Esfuerzo estimado inicial | Nuevo esfuerzo estimado al final del día | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | D10 | D11 | D12 | D13 | D14 | D15 |
| Instalación de dependencias. | Instalación de SDK para NetCore | APP | 0.5 | 0.5 | | | | | | | | | | | | | | |
| | Instalación de otras dependencias adicionales | APP | 0.5 | 0.5 | | | | | | | | | | | | | | |
| Configuración del proyecto. | Definir parametros de configuración | APP | 2 | | 1 | 1 | | | | | | | | | | | | |
| | Configuración de inyección de dependencia | APP | 1 | | | | 1 | | | | | | | | | | | |
| | Configuración de DomainResolver | APP | 2 | | | | | | | | | | | | | | | |
| | Configuración de aspectos globales del proyecto | APP | 2 | | | | | | | | | | | | | | | |
| Configuración de modelo de datos | Creación de clases entidad | APP | 2 | | | | | | | | | | | | | | | |
| | Creacion de clases modelo | APP | 2 | | | | | | | | | | | | | | | |
| | Creación de controlador de versiones de DB | APP | 3 | | | | | | | | | | | | | | | |
| | | REAL | 15 | 14 | 13 | 12 | 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | DESEADO | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Figura N°4.12: Pila del segundo Sprint primeros 4 días (Elaboración propia).

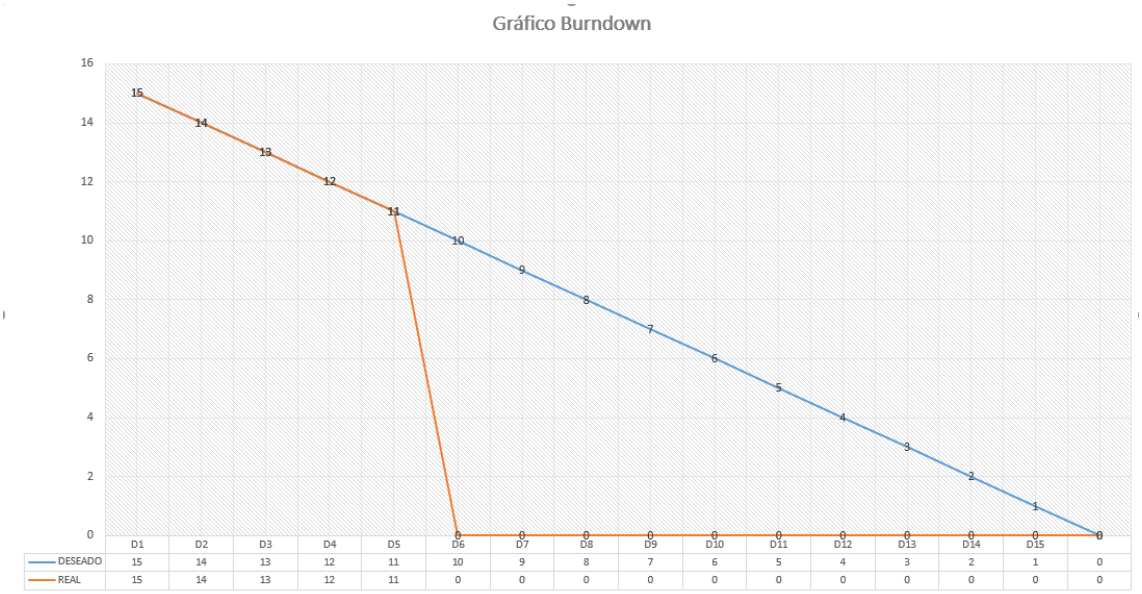



Figura N°4.13: Gráfico burndown primeros 4 días (Elaboración propia).

| Elemento de la pila de producto | Tarea del Sprint | Voluntario | Esfuerzo estimado inicial | Nuevo esfuerzo estimado al final del día | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | D10 | D11 | D12 | D13 | D14 | D15 |
| Instalación de dependencias. | Instalación de SDK para NetCore | APP | 0.5 | 0.5 | | | | | | | | | | | | | | |
| | Instalación de otras dependencias adicionales | APP | 0.5 | 0.5 | | | | | | | | | | | | | | |
| Configuración del proyecto. | Definir parametros de configuración | APP | 2 | | 1 | 1 | | | | | | | | | | | | |
| | Configuración de inyección de dependencia | APP | 1 | | | | 1 | | | | | | | | | | | |
| | Configuración de DomainResolver | APP | 2 | | | | | 2 | | | | | | | | | | |
| | Configuración de aspectos globales del proyecto | APP | 2 | | | | | | 2 | | | | | | | | | |
| Configuración de modelo de datos | Creación de clases entidad | APP | 2 | | | | | | | 0.5 | 0.5 | 0.5 | 0.5 | | | | | |
| | Creacion de clases modelo | APP | 2 | | | | | | | | | | | | | | | |
| | Creación de controlador de versiones de DB | APP | 3 | | | | | | | | | | | | | | | |
| | | REAL | 15 | 14 | 13 | 12 | 11 | 9 | 7 | 6.5 | 6 | 5.5 | 5 | 0 | 0 | 0 | 0 | 0 |
| | | DESEADO | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Figura N°4.14: Pila del segundo Sprint primeros 10 días (Elaboración propia).

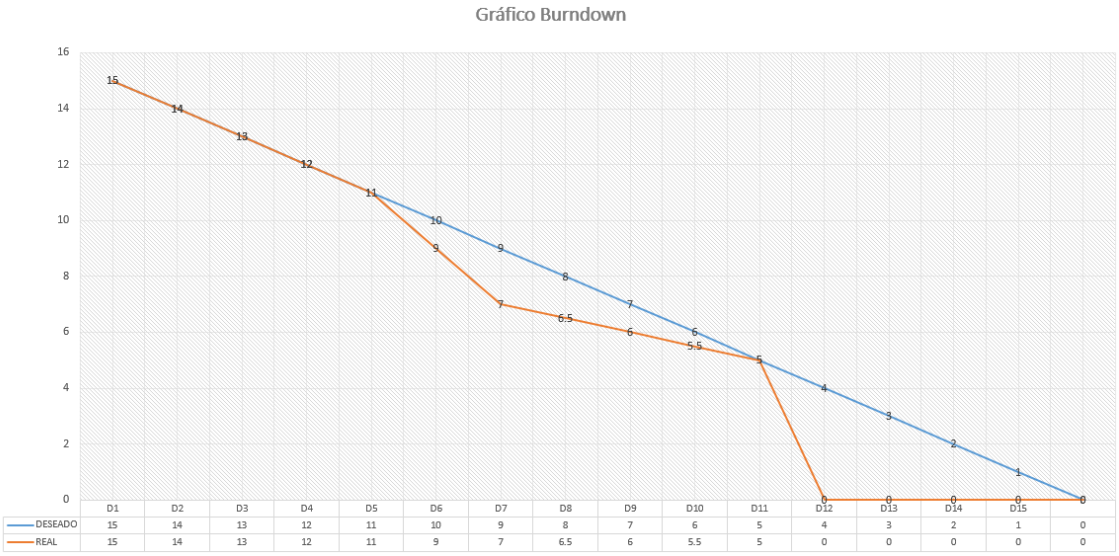
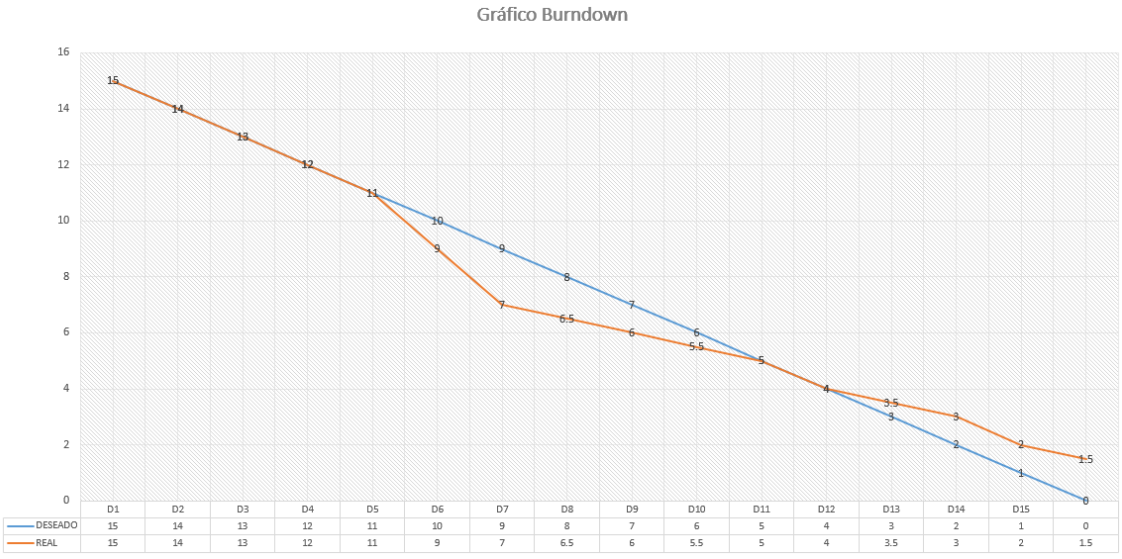



Figura N°4.15: Gráfico burndown primeros 10 días (Elaboración propia).

| Elemento de la pila de producto | Tarea del Sprint | Voluntario | Esfuerzo estimado inicial | Nuevo esfuerzo estimado al final del día | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | D10 | D11 | D12 | D13 | D14 | D15 |
| Instalación de dependencias. | Instalación de SDK para NetCore | APP | 0.5 | 0.5 | | | | | | | | | | | | | | |
| | Instalación de otras dependencias adicionales | APP | 0.5 | 0.5 | | | | | | | | | | | | | | |
| Configuración del proyecto. | Definir parametros de configuración | APP | 2 | | 1 | 1 | | | | | | | | | | | | |
| | Configuración de inyección de dependencia | APP | 1 | | | | 1 | | | | | | | | | | | |
| | Configuración de DomainResolver | APP | 2 | | | | | 2 | | | | | | | | | | |
| | Configuración de aspectos globales del proyecto | APP | 2 | | | | | | 2 | | | | | | | | | |
| Configuración de modelo de datos | Creación de clases entidad | APP | 2 | | | | | | | 0.5 | 0.5 | 0.5 | 0.5 | | | | | |
| | Creacion de clases modelo | APP | 2 | | | | | | | | | | | 1 | 0.5 | 0.5 | | |
| | Creación de controlador de versiones de DB | APP | 3 | | | | | | | | | | | | | | 1 | 0.5 |
| | | REAL | 15 | 14 | 13 | 12 | 11 | 9 | 7 | 6.5 | 6 | 5.5 | 5 | 4 | 3.5 | 3 | 2 | 1.5 |
| | | DESEADO | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Figura N°4.16: Pila del segundo Sprint completado al 100% (Elaboración propia).




| | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | D10 | D11 | D12 | D13 | D14 | D15 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DESEADO | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| REAL | 15 | 14 | 13 | 12 | 11 | 9 | 7 | 6.5 | 6 | 5.5 | 5 | 4 | 3.5 | 3 | 2 | 1.5 |

Figura N°4.17: Gráfico burndown completado al 100% (Elaboración propia).

#### 4.2.2.2.4 RESULTADOS SPRINT BACKLOG 02

Instalación de las dependencias y paquetes necesarios para iniciar el proyecto de control de versiones del esquema de base de datos relacional, fueron necesarios instalar paquete NUGET (Figura N° 4.18), y paquetes de SDK (Figura 4.19) y otras dependencias para manipular documentos y ficheros de configuración (Figura N° 4.20).

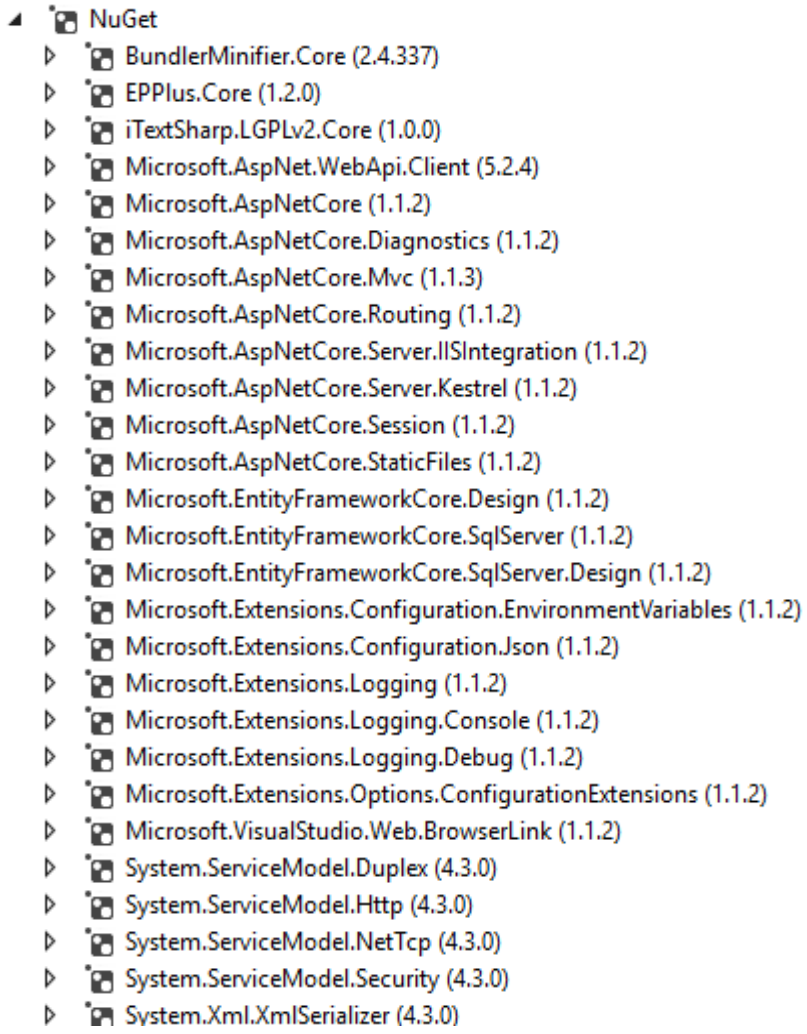



Figura N°4.18: Dependencias Nuget instaladas (Elaboración propia).

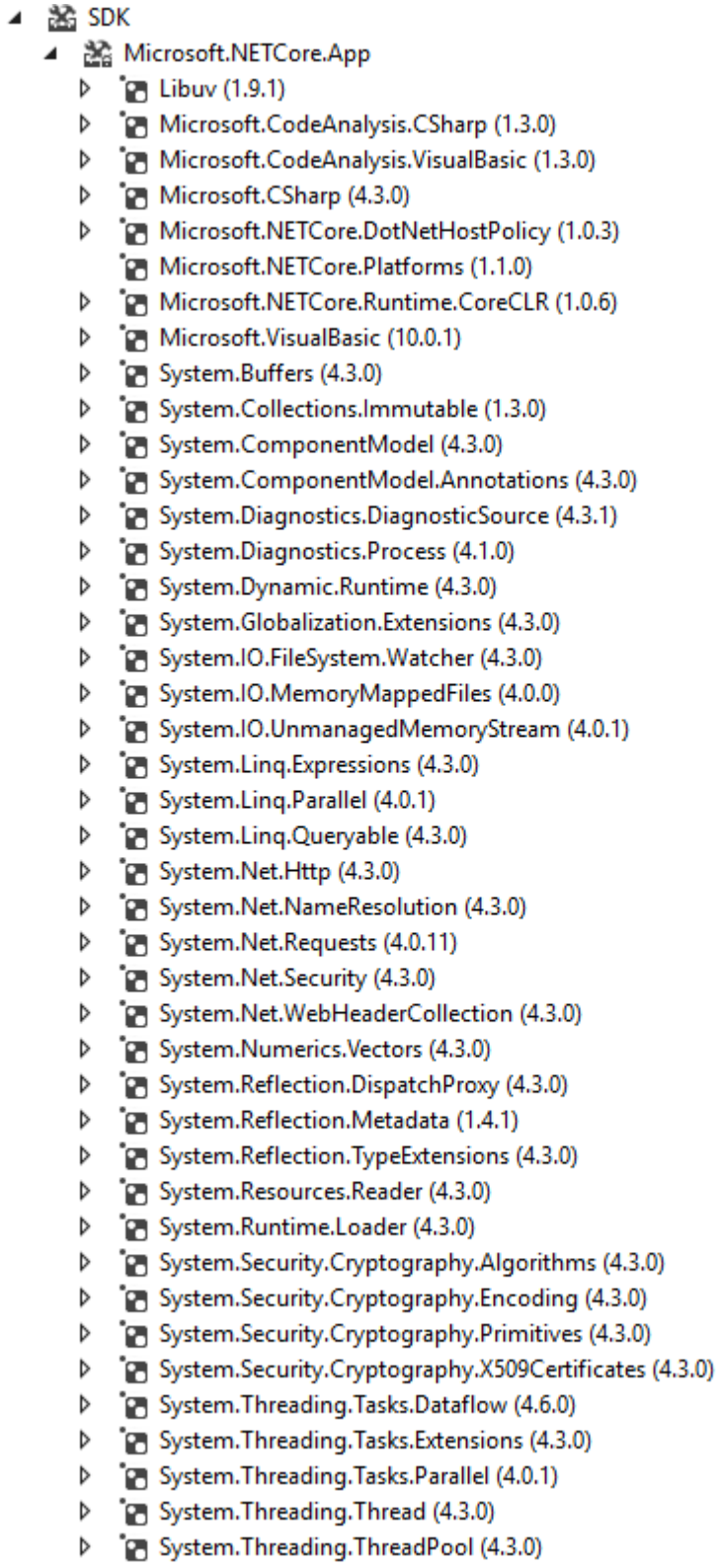



Figura N°4.19: Dependencias SDK instaladas (Elaboración propia).

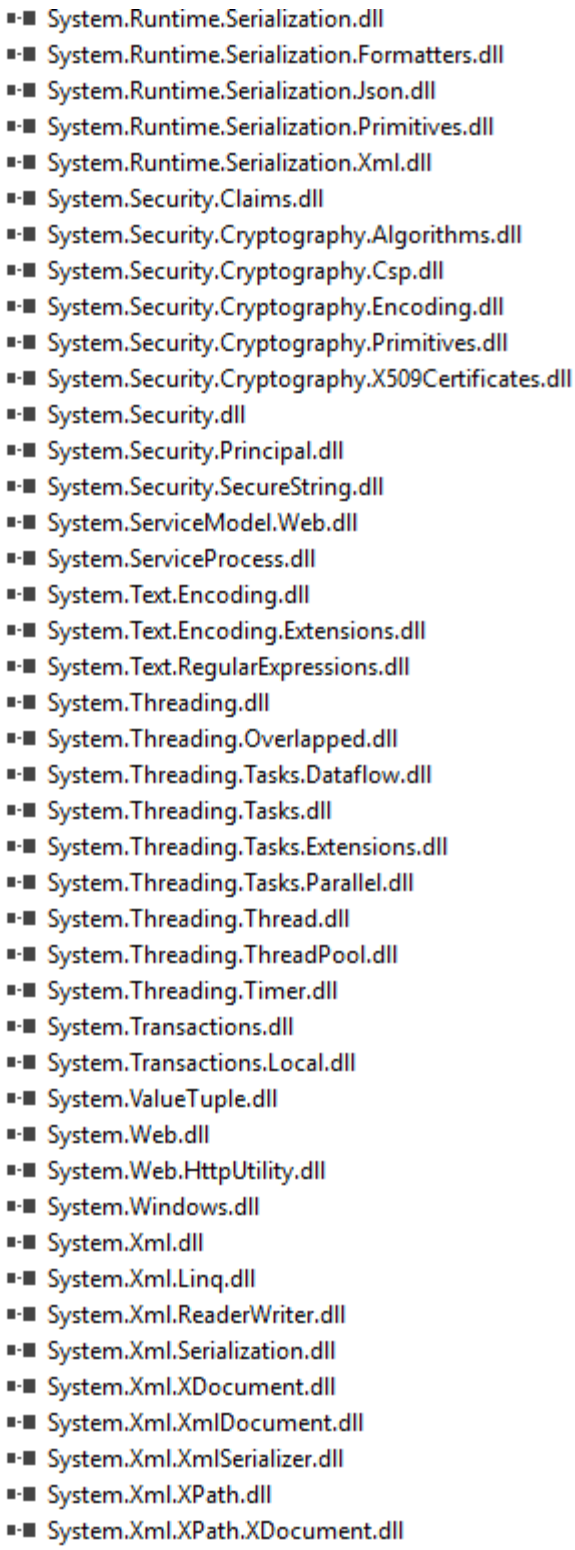



Figura N°4.20: Dependencias para la manipulación de documentos (Elaboración propia).

La siguiente clase es para configurar el inicio o arranque del proyecto, ver (Tabla N° 4.10).

```
public Startup(IHostingEnvironment env)
        {
            var builder = new ConfigurationBuilder()
                .SetBasePath(env.ContentRootPath)
                .AddJsonFile("appsettings.json", optional: true, reloadOnChange:
true)
                .AddJsonFile($"appsettings.{env.EnvironmentName}.json", optional:
true)
                .AddEnvironmentVariables();
            string sAppPath = env.ContentRootPath;
            string swwwRootPath = env.WebRootPath;
            Configuration = builder.Build();
```

Tabla N° 4.10: Clase para configurar el inicio del proyecto (Elaboración propia)

Clase principal para configurar las inyecciones de dependencia, ver Tabla N° 4.11.

```
public void ConfigureServices(IServiceCollection services)
        {
            services.AddDistributedMemoryCache();
            services.AddSession(options =>
            {
                options.IdleTimeout = TimeSpan.FromMinutes(10);
            });
            services.AddSingleton<IHttpContextAccessor, HttpContextAccessor>();
            services.AddScoped<AppTambox.org.Helpers.SessionUsuario>();
            services.AddMvc();
            services.AddDbContext<EFDbContext>(options                          =>
options.UseSqlite(Configuration.GetConnectionString("TamboxDatabase")));

            services.Configure<AppSettings>(Configuration);
            DependecyInjection(services);
        }
public   void   Configure(IApplicationBuilder   app,   IHostingEnvironment   env,
ILoggerFactory loggerFactory)
        {
            loggerFactory.AddConsole(Configuration.GetSection("Logging"));
            loggerFactory.AddDebug();

            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
                app.UseBrowserLink();
            }
            else
            {
                app.UseExceptionHandler("/Home/Error");
            }

            app.UseStaticFiles();
            app.UseSession();
            app.UseMvc(routes =>
            {
                routes.MapRoute(
                    name: "default",
                    template: "{controller=Account}/{action=Index}/{id?}");
            });
        }
```

Tabla N° 4.11: Clase para configurar las inyecciones de dependencia (Elaboración propia)

Método que realiza las inyecciones de dependencia de todo el proyecto, Tabla N° 4.12.

```
private void DependecyInjection(IServiceCollection services)
        {
            services.AddSingleton<IContenedorProvider, ContenedorProvider>();
            services.AddSingleton<IProductoProvider, ProductoProvider>();
            services.AddSingleton<IUsuarioProvider, UsuarioProvider>();
            services.AddSingleton<IValeEntradaProvider, ValeEntradaProvider>();
            services.AddSingleton<IValeSalidaProvider, ValeSalidaProvider>();
            services.AddSingleton<IPartidaProvider, PartidaProvider>();
            services.AddSingleton<IAlmacenProvider, AlmacenProvider>();
            services.AddSingleton<IKardexProvider, KardexProvider>();
            services.AddSingleton<IRequerimientoProvider,
RequerimientoProvider>();
            services.AddSingleton<IInsumoProvider, InsumoProvider>();
            services.AddSingleton<IPersonaProvider, PersonaProvider>();
            services.AddSingleton<IEscalaRemunerativaProvider,
EscalaRemunerativaProvider>();
            services.AddSingleton<ITareoProvider, TareoProvider>();
            services.AddSingleton<IExpedienteProvider, ExpedienteProvider>();
            services.AddSingleton<IPersonaHistorialProvider,
PersonaHistorialProvider>();
            services.AddSingleton<IUserSessionProvider, UserSessionProvider>();
            services.AddSingleton<ISubContratoProvider, SubContratoProvider>();
            services.AddSingleton<IValeCombustibleProvider,
ValeCombustibleProvider>();
            services.AddSingleton<IMaquinariaProvider, MaquinariaProvider>();
```

Tabla N° 4.12: Clase para configurar las inyecciones de dependencia del proyecto (Elaboración propia)

Configuración del origen de datos, ver Tabla N° 4.13.

```
{
  "Logging": {
    "IncludeScopes": false,
    "LogLevel": {
      "Default": "Debug",
      "System": "Information",
      "Microsoft": "Information"
    }
  },
  "ConnectionStrings": {
    "TamboxDatabase": "Filename=dir::database.db"
  },
  "EndpointUrl": "wcf-service.svc",
  "EndpointDns": "192.168.1.41",
  "EndpointPort": ":8787"
}
```

Tabla N° 4.13: Configuración de origen de datos (Elaboración propia)

Creación y configuración de Domain resolver, ver Tabla N° 4.14.

```
public static string GetIpDomainApi(string domain, string port)
        {
            if        (domain.Split(new        char[]        {        '.'        },
StringSplitOptions.RemoveEmptyEntries).Length == 4)
            {
                IPAddress address;
                if (IPAddress.TryParse(domain, out address))
                {
                    return "http://" + address + port;
                }
                else
                {
                    return "http://" + "127.0.0.1" + port;
                }
            }
            else if (domain.Equals("localhost"))
            {
                var ip = domain;
                return "http://" + ip + port;
            }
            else
            {
                var                  result                  =                  new
HttpClient().GetAsync(domain).Result.Content.ReadAsStringAsync().Result;
                var ip = result.Split('>')[3].Split('-')[0].Trim();
                return "http://" + ip + port;
            }
        }
```

Tabla N° 4.14: configuración de domain resolver (Elaboración propia)

En la tabla N° 4.15, se muestra la configuración del modelo de datos: Clase para almacenar parámetros generales del aplicativo.

```
[Table("T_USER_SESSION")]
    public class UserSessionModel
    {
        [Key] [Column("ID_USER_SESION")] public decimal idUserSession { get; set;
}
        [Column("PARAMETER_1")] public string fechaActivacion { get; set; }
        [Column("PARAMETER_2")] public string fechaLogin { get; set; }
        [Column("PARAMETER_3")] public string computerName { get; set; }
        [Column("PARAMETER_4")] public string tamboxKey { get; set; }
        [Column("PARAMETER_5")] public string daysActivate { get; set; }
        [Column("PARAMETER_6")] public string daysCurrent { get; set; }
        [Column("FECHA_CREACION")] public string dateCurrent { get; set; }
        [Column("KEY")] public string key { get; set; }
```

Tabla N° 4.15: Clase para almacenar parámetros generales de aplicativo (Elaboración propia)

En la tabla N° 4.16, se muestra la clase UsuarioModel, esta clase es la encargada para acceder al sistema.

```
[Table("T_USUARIO")]
    public class UsuarioModel
    {
        public UsuarioModel()
        {
        }
        [Key]
        [Column("ID_USUARIO")]
        public decimal idUser { get; set; }
        [Column("USUARIO")]
        public string user { get; set; }
        [Column("CONTRASENIA")]
        public string pass { get; set; }
        [Column("ESTADO")]
        public bool status { get; set; }
        [Column("ID_PERSONA")]
        public decimal? idPersona { get; set; }

    }
```

Tabla N° 4.16: Clase modelo de acceso al aplicativo (Elaboración propia)

En la tabla N° 4.17, se muestra la clase AppSettings, esta clase es la encargada de configurar los puntos de acceso desde otros sistemas.

```
public class AppSettings
    {
        public string EndpointUrl { get; set; }
        public string EndpointDns { get; set; }
        public string EndpointPort { get; set; }
    }
```

Tabla N° 4.17: Clase de configuración de accesos (Elaboración propia)

Clase ContenedorModel es la clase para guardar data para todo tipo de objetos por ejemplo TipoPersona, TipoDocumento, etc, se puede observar en la Tabla N° 4.18.

```
[Table("T_CONTENEDOR")]
    public class ContenedorModel
    {
        [Key]
        [DataMemberAttribute()]
        [Column("ID_CONTENEDOR")]
        public decimal idContenedor { get; set; }
        [DataMemberAttribute()]     [Column("DENOMINACION")]     public     string
denominacion { get; set; }
        [DataMemberAttribute()] [Column("ABREVIATURA")] public string abreviatura
{ get; set; }
        [DataMemberAttribute()] [Column("ID_TABLA")] public decimal idTabla { get;
set; }
        [DataMemberAttribute()] [Column("ESTADO")] public bool estado { get; set;
}
        [DataMemberAttribute()] [Column("CODIGO")] public string codigo { get; set;
}
        [DataMemberAttribute()]    [Column("ID_SUB_TABLA")]    public    decimal?
```

Tabla N° 4.18: Clase para para guardar tipos de datos genéricos (Elaboración propia)

#### 4.2.2.2.5 SPRINT BACKLOG 03

En este sprint el equipo modifica la estimación inicial de "Configuración del contexto del proyecto" de un esfuerzo inicial 7 a 12, en función al conocimiento del tema y se compromete a entregar los siguientes elementos, ver (Tabla N° 4.19).

| Elemento | Prioridad |
|---|---|
| Configuración del contexto de proyecto. | 5 |
| Testear el componente software. | 6 |
| Elaborar documentación del componente software. | 7 |

Tabla N°4.19: Elementos para el tercer Sprint (Elaboración propia).

La reunión del tercer sprint tuvo una duración de aproximadamente 4 horas, esta vez el enfoque es la creación de algunas partes importantes del componente adicionalmente se pide que se haga los test y se genere la documentación necesaria, ver (Tabla N° 4.20 y Figura N° 4.21).

| Longitud del Sprint | | | 4 semanas |
|---|---|---|---|
| Días laborables durante el Sprint | | | 18 días |
| Miembro del equipo | Días disponibles durante el sprint | Horas disponibles por día | Total horas disponibles |
| Alex Palomino Pariona | 18 | 7 | 126 |

Tabla N°4.20: Estimación de tiempo disponible para Sprint (Elaboración propia).

| Elemento de la pila de producto | Tarea del Sprint | Voluntario | Esfuerzo estimado inicial | Nuevo esfuerzo estimado al final del día | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | D10 | D11 | D12 | D13 | D14 | D15 | D16 | D17 | D18 | |
| Configuración del contexto de proyecto. | Creación de soporte para linea de comados | APP | 3 | | | | | | | | | | | | | | | | | | | |
| | Creación de clases con soporte Scaffolding | APP | 3 | | | | | | | | | | | | | | | | | | | |
| | Creación de clases para soportar Tipos de datos | APP | 3 | | | | | | | | | | | | | | | | | | | |
| | Creación de clases Contexto de DB | APP | 3 | | | | | | | | | | | | | | | | | | | |
| Testear el componente software | testear funcioanlidad de cambio de versiones | APP | 2 | | | | | | | | | | | | | | | | | | | |
| | Testear creación de tablas | APP | 2 | | | | | | | | | | | | | | | | | | | |
| Elaborar documentación del componente | Documentar avances del componente | APP | 2 | | | | | | | | | | | | | | | | | | | |
| | | REAL | 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | DESEADO | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |

Figura N°4.21: Pila del tercer Sprint (Elaboración propia).

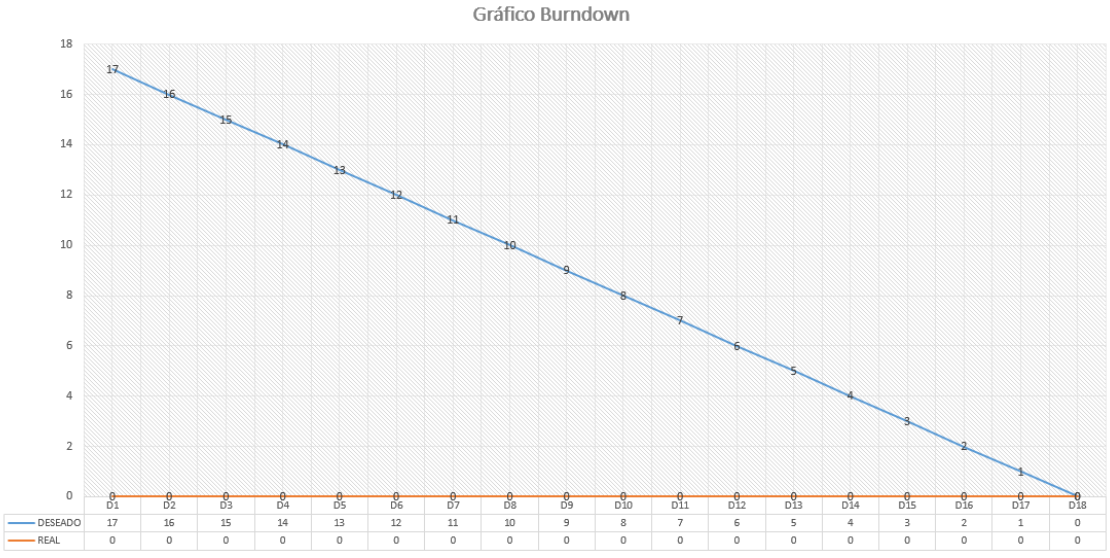



Figura N°4.22: Gráfico burndown inicial del tercer sprint (Elaboración propia).

En este Sprint algunos problemas de tiempo y muchas tareas no se concluyeron satisfactoriamente, requiriendo que se reúnan con del dueño del producto y revisar el desarrollo en su integridad, (Figura N° 4.23 y Figura N° 4.24).

| Elemento de la pila de producto | Tarea del Sprint | Voluntario | Esfuerzo estimado inicial | Nuevo esfuerzo estimado al final del día | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | D10 | D11 | D12 | D13 | D14 | D15 | D16 | D17 | D18 |
| Configuración del contexto de proyecto. | Creación de soporte para linea de comados | APP | 3 | 0.5 | 0.5 | 1 | 1 | | | | | | | | | | | | | | |
| | Creación de clases con soporte Scaffolding | APP | 3 | | | | | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | | | | | | | | |
| | Creación de clases para soportar Tipos de datos | APP | 3 | | | | | | | | | | | 1 | 0.5 | 0.5 | 1 | | | | |
| | Creación de clases Contexto de DB | APP | 3 | | | | | | | | | | | | | | | 1 | 1 | 0.5 | 0.5 |
| Testear el componente software | testear funcioanlidad de cambio de versiones | APP | 2 | | | | | | | | | | | | | | | | | | |
| | Testear creación de tablas | APP | 2 | | | | | | | | | | | | | | | | | | |
| Elaborar documentación del componente software | Documentar avances del componente | APP | 2 | | | | | | | | | | | | | | | | | | |
| | | | REAL | 18 | 17.5 | 17 | 16 | 15 | 14.5 | 14 | 13.5 | 13 | 12.5 | 12 | 11 | 10.5 | 10 | 9 | 8 | 7 | 6.5 | 6 |
| | | | DESEADO | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Figura N°4.23: Pila del tercer Sprint completado al 100% (Elaboración propia).

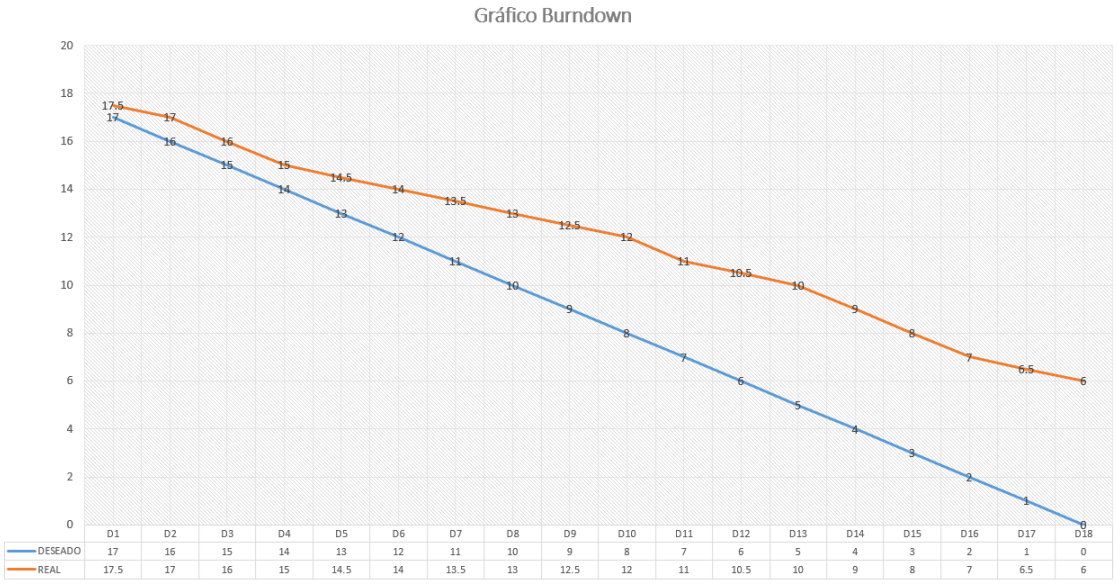



Figura N°4.24: Gráfico burndown del tercer sprint al 100% (Elaboración propia).

#### 4.2.2.2.6 RESULTADOS SPRINT BACKLOG 03

Configuración del contexto del proyecto: la tabla N° 4.21 muestra la clase que acepta caracteres especiales para aplicar algún patrón generado por el usuario.

```
internal class AnsiTextWriter
    {
        private readonly TextWriter _writer;

        public AnsiTextWriter(TextWriter writer) => _writer = writer;

        public void WriteLine(string text)
        {
            Interpret(text);
            _writer.Write(Environment.NewLine);
        }

        private void Interpret(string value)
        {
            var matches = Regex.Matches(value, "\x1b\\[([0-9]+)?m");

            var start = 0;
            foreach (Match match in matches)
            {
                var length = match.Index - start;
                if (length != 0)
                {
                    _writer.Write(value.Substring(start, length));
                }

                Apply(match.Groups[1].Value);

                start = match.Index + match.Length;
            }

            if (start != value.Length)
            {
                _writer.Write(value.Substring(start));
            }
        }
```

Tabla N° 4.21: Clase para usar patrones especiales (Elaboración propia)

En la tabla N° 4.22, se muestran métodos para generar mensaje a colores en las diferentes llamadas al componente.

```
internal abstract class CommandBase
    {
        public virtual void Configure(CommandLineApplication command)
        {
            var          verbose          =          command.Option("-v|--verbose",
Resources.VerboseDescription);
            var           noColor           =           command.Option("--no-color",
Resources.NoColorDescription);
            var       prefixOutput       =       command.Option("--prefix-output",
Resources.PrefixDescription);

            command.HandleResponseFiles = true;

            command.OnExecute(
                () =>
                    {
                        Reporter.IsVerbose = verbose.HasValue();
                        Reporter.NoColor = noColor.HasValue();
                        Reporter.PrefixOutput = prefixOutput.HasValue();

                        Validate();

                        return Execute();
                    });
        }

        protected virtual void Validate()
        {
        }

        protected virtual int Execute() => 0;
    }
```

Tabla N° 4.22: Clase genérica para mostrar colores (Elaboración propia)

En la tabla N° 4.23, se muestran métodos con soporte para realizar las operaciones sobre esquema de base de datos relacionales.

```
internal class DatabaseCommand : HelpCommandBase
    {
        public override void Configure(CommandLineApplication command)
        {
            command.Description = Resources.DatabaseDescription;

            command.Command("drop", new DatabaseDropCommand().Configure);
            command.Command("update", new DatabaseUpdateCommand().Configure);

            base.Configure(command);
        }
    }
internal partial class DatabaseDropCommand : ContextCommandBase
    {
        private CommandOption _force;
        private CommandOption _dryRun;

        public override void Configure(CommandLineApplication command)
        {
            command.Description = Resources.DatabaseDropDescription;

            _force                  =                  command.Option("-f|--force",
Resources.DatabaseDropForceDescription);
            _dryRun                  =                   command.Option("--dry-run",
Resources.DatabaseDropDryRunDescription);

            base.Configure(command);
        }
    }
internal partial class DatabaseUpdateCommand : ContextCommandBase
    {
        private CommandArgument _migration;

        public override void Configure(CommandLineApplication command)
        {
            command.Description = Resources.DatabaseUpdateDescription;

            _migration              =              command.Argument("<MIGRATION>",
Resources.MigrationDescription);

            base.Configure(command);
        }
    }
internal class DbContextCommand : HelpCommandBase
    {
        public override void Configure(CommandLineApplication command)
        {
            command.Description = Resources.DbContextDescription;

            command.Command("info", new DbContextInfoCommand().Configure);
            command.Command("list", new DbContextListCommand().Configure);
            command.Command("scaffold", new DbContextScaffoldCommand().Configure);

            base.Configure(command);
        }
    }
```

```
internal partial class DbContextScaffoldCommand : ProjectCommandBase
    {
        private CommandArgument _connection;
        private CommandArgument _provider;
        private CommandOption _dataAnnotations;
        private CommandOption _context;
        private CommandOption _contextDir;
        private CommandOption _force;
        private CommandOption _outputDir;
        private CommandOption _schemas;
        private CommandOption _tables;
        private CommandOption _useDatabaseNames;
        private CommandOption _json;

        public override void Configure(CommandLineApplication command)
        {
            command.Description = Resources.DbContextScaffoldDescription;

            _connection             =             command.Argument("<CONNECTION>",
Resources.ConnectionDescription);
            _provider               =               command.Argument("<PROVIDER>",
Resources.ProviderDescription);

            _dataAnnotations       =       command.Option("-d|--data-annotations",
Resources.DataAnnotationsDescription);
            _context        =         command.Option("-c|--context        <NAME>",
Resources.ContextNameDescription);
            _contextDir       =       command.Option("--context-dir       <PATH>",
Resources.ContextDirDescription);
            _force                  =                 command.Option("-f|--force",
Resources.DbContextScaffoldForceDescription);
            _outputDir       =       command.Option("-o|--output-dir       <PATH>",
Resources.OutputDirDescription);
            _schemas      =      command.Option("--schema      <SCHEMA_NAME>...",
Resources.SchemasDescription);
            _tables      =      command.Option("-t|--table      <TABLE_NAME>...",
Resources.TablesDescription);
            _useDatabaseNames       =       command.Option("--use-database-names",
Resources.UseDatabaseNamesDescription);
            _json = Json.ConfigureOption(command);

            base.Configure(command);
        }
    }
```

Tabla N° 4.23: Clase para realizar operaciones sobre bases de datos relacionales (Elaboración propia).

En la tabla N° 4.24, se muestran métodos con comandos para realizar el control de cambios en los esquemas de bases de datos relacionales.

```
internal partial class MigrationsAddCommand : ContextCommandBase
    {
        private CommandArgument _name;
        private CommandOption _outputDir;
        private CommandOption _json;

        public override void Configure(CommandLineApplication command)
        {
            command.Description = Resources.MigrationsAddDescription;

            _name                    =                    command.Argument("<NAME>",
Resources.MigrationNameDescription);

            _outputDir       =       command.Option("-o|--output-dir       <PATH>",
Resources.MigrationsOutputDirDescription);
            _json = Json.ConfigureOption(command);

            base.Configure(command);
        }
    }
```

```
internal class MigrationsCommand : HelpCommandBase
    {
        public override void Configure(CommandLineApplication command)
        {
            command.Description = Resources.MigrationsDescription;

            command.Command("add", new MigrationsAddCommand().Configure);
            command.Command("list", new MigrationsListCommand().Configure);
            command.Command("remove", new MigrationsRemoveCommand().Configure);
            command.Command("script", new MigrationsScriptCommand().Configure);

            base.Configure(command);
        }
    }
```

```
public class ForeignKeyEventData : EventData
    {
        public ForeignKeyEventData(
            [NotNull] EventDefinitionBase eventDefinition,
            [NotNull]      Func<EventDefinitionBase,      EventData,      string>
messageGenerator,
            [NotNull] IForeignKey foreignKey)
            : base(eventDefinition, messageGenerator)
        {
            ForeignKey = foreignKey;
        }

        public virtual IForeignKey ForeignKey { get; }
    }
```

Tabla N° 4.24: Clases genéricas para ejecutar comandos para el control de cambios de los esquemas de base de datos relacional (Elaboración propia)
La tabla N° 4.25, muestra el esquema inicial de base de datos relacional (Captura de código fuente usando el componente software), creación de la configuración del control de versiones etapa inicial.

```
protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "T_ALMACEN",
                columns: table => new
                {
                    ID_ALMACEN = table.Column<decimal>(nullable: false),
                    DENOMINACION = table.Column<string>(nullable: true),
                    ESTADO = table.Column<bool>(nullable: false),
                    ID_EXPEDIENTE = table.Column<decimal>(nullable: true)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_T_ALMACEN", x => x.ID_ALMACEN);
                });
migrationBuilder.CreateTable(
                name: "T_CONTENEDOR",
                columns: table => new
                {
                    ID_CONTENEDOR = table.Column<decimal>(nullable: false),
                    ABREVIATURA = table.Column<string>(nullable: true),
                    CODIGO = table.Column<string>(nullable: true),
                    DENOMINACION = table.Column<string>(nullable: true),
                    ESTADO = table.Column<bool>(nullable: false),
                    ID_SUB_TABLA = table.Column<decimal>(nullable: true),
                    ID_TABLA = table.Column<decimal>(nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_T_CONTENEDOR", x => x.ID_CONTENEDOR);
                });
```

```
      migrationBuilder.CreateTable(
                name: "T_INSUMO",
                columns: table => new
                {
                    ID_INSUMO = table.Column<decimal>(nullable: false),
                    CANTIDAD = table.Column<decimal>(nullable: true),
                    CODIGO_S10 = table.Column<string>(nullable: true),
                    CUADRILLA = table.Column<decimal>(nullable: true),
                    DENOMINACION = table.Column<string>(nullable: true),
                    FECHA_CREACION = table.Column<DateTime>(nullable: true),
                    FECHA_MODIFICACION = table.Column<DateTime>(nullable: true),
                    ID_META_PARTIDA = table.Column<decimal>(nullable: true),
                    ID_USUARIO_CREACION = table.Column<decimal>(nullable: true),
                    ID_USUARIO_MODIFICACION    =    table.Column<decimal>(nullable:
true),
                    IDC_CLASIFICACION = table.Column<decimal>(nullable: true),
                    IDC_TIPO_COSTO = table.Column<decimal>(nullable: true),
                    IDC_UNIDAD_MEDIDA = table.Column<decimal>(nullable: true),
                    ID_S_INSUMO = table.Column<decimal>(nullable: true),
                    OBSERVACION = table.Column<double>(nullable: true),
                    PARCIAL = table.Column<decimal>(nullable: true),
                    PRECIO_UNITARIO = table.Column<decimal>(nullable: true)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_T_INSUMO", x => x.ID_INSUMO);
                });


migrationBuilder.CreateTable(
                name: "T_KARDEX",
                columns: table => new
                {
                    ID_KARDEX = table.Column<decimal>(nullable: false),
                    CANT_ENTRADA = table.Column<decimal>(nullable: true),
                    CANT_SALDO = table.Column<decimal>(nullable: true),
                    CANT_SALIDA = table.Column<decimal>(nullable: true),
                    COSTO_UNITARIO = table.Column<decimal>(nullable: true),
                    FECHA_CREACION = table.Column<DateTime>(nullable: true),
                    FECHA_OPERACION = table.Column<DateTime>(nullable: true),
                    ID_DESTINO = table.Column<decimal>(nullable: true),
                    ID_EXPEDIENTE = table.Column<decimal>(nullable: true),
                    ID_KARDEX_SIACPI = table.Column<decimal>(nullable: true),
                    ID_ORIGEN = table.Column<decimal>(nullable: true),
                    ID_PRODUCTO = table.Column<decimal>(nullable: true),
                    IDC_MARCA = table.Column<decimal>(nullable: true),
                    IDC_UNIDAD_MEDIDA = table.Column<decimal>(nullable: true),
                    OBSERVACION = table.Column<string>(nullable: true),
                    VAL_CANT_SALDO = table.Column<decimal>(nullable: true),
                    VAL_CANT_ENTRADA = table.Column<decimal>(nullable: true),
                    VAL_CANT_SALIDA = table.Column<decimal>(nullable: true)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_T_KARDEX", x => x.ID_KARDEX);
                });
```

```
    migrationBuilder.CreateTable(
                name: "T_META_PARTIDA",
                columns: table => new
                {
                    ID_META_PARTIDA = table.Column<decimal>(nullable: false),
                    CODIGO_META = table.Column<string>(nullable: true),
                    DENOMINACION_META = table.Column<string>(nullable: true),
                    ID_EXPEDIENTE = table.Column<decimal>(nullable: true),
                    METRADO = table.Column<decimal>(nullable: true)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_T_META_PARTIDA", x => x.ID_META_PARTIDA);
                });

            migrationBuilder.CreateTable(
                name: "T_PERSONA",
                columns: table => new
                {
                    ID_PERSONA = table.Column<decimal>(nullable: false),
                    APE_MATERNO = table.Column<string>(nullable: true),
                    APE_PATERNO = table.Column<string>(nullable: true),
                    ID_S_PERSONA = table.Column<decimal>(nullable: true),
                    NOMBRE_COMPLETO = table.Column<string>(nullable: true),
                    NOMBRES = table.Column<string>(nullable: true),
                    NRO_DOC = table.Column<string>(nullable: true)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_T_PERSONA", x => x.ID_PERSONA);
                });


migrationBuilder.CreateTable(
                name: "T_PRODUCTO",
                columns: table => new
                {
                    ID_PRODUCTO = table.Column<decimal>(nullable: false),
                    CODIGO = table.Column<string>(nullable: true),
                    DENOMINACION = table.Column<string>(nullable: true),
                    ID_S_PRODUCTO = table.Column<decimal>(nullable: true),
                    ESTADO = table.Column<bool>(nullable: true)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_T_PRODUCTO", x => x.ID_PRODUCTO);
                });
```

Tabla N° 4.25: configuración realizada por el componente para realizar el control de versiones del esquema de base de datos relacional (Elaboración propia)

La tabla N° 4.26, muestra la adición de la columna ESTADO tipo de datos BIT, FECHA_MODIF tipo de dato DATETIME, FECHA_NACIMIENTO tipo de dato DATETIME Y ID_ESCALA_REMUNERATIVA tipo de dato NUMERIC (32,0) a la tabla T_PERSONA.

```
migrationBuilder.AddColumn<bool>(
                name: "ESTADO",
                table: "T_PERSONA",
                type: "BIT",
                nullable: true);


            migrationBuilder.AddColumn<DateTime>(
                name: "FECHA_MODIF",
                table: "T_PERSONA",
                type: "DATETIME",
                nullable: true);

            migrationBuilder.AddColumn<DateTime>(
                name: "FECHA_NACIMIENTO",
                table: "T_PERSONA",
                type:"DATETIME",
                nullable: true);

            migrationBuilder.AddColumn<decimal>(
                name: "ID_ESCALA_REMUNERATIVA",
                table: "T_PERSONA",
                type:"NUMERIC(32,0)",
                nullable: true);

            migrationBuilder.AddColumn<decimal>(
                name: "IDC_CARGO",
                table: "T_PERSONA",
                type:"NUMERIC(32,0)",
                nullable: true);
```

Tabla N° 4.26: Configuración de nuevos cambios en el modelo de datos (Elaboración propia)

La tabla N° 4.27, muestra la creación de una nueva tabla T_PERSONA_HISTORIAL con su respectiva llave primaria PK_T_PERSONA_HISTORIAL.

```
  migrationBuilder.CreateTable(
                name: "T_PERSONA_HISTORIAL",
                columns: table => new
                {
                    ID_PERSONA_HISTORIAL = table.Column<decimal>(nullable: false,
type: "NUMERIC(32,0)"),
                    CALIFICACION = table.Column<decimal>(nullable: true, type:
"NUMERIC(32,0)"),
                    DESCRIPCION = table.Column<string>(nullable: true, type:
"VARCHAR(800)"),
                    ESTADO = table.Column<bool>(nullable: false, type: "BIT"),
                    FECHA_CREACION = table.Column<DateTime>(nullable: true, type:
"DATETIME"),
                    FECHA_EVENTO = table.Column<DateTime>(nullable: true, type:
"DATETIME"),
                    FECHA_MODIFICACION = table.Column<DateTime>(nullable: true,
type: "DATETIME"),
                    ID_EXPEDIENTE = table.Column<decimal>(nullable: true, type:
"NUMERIC(32,0)"),
                    ID_PERSONA = table.Column<decimal>(nullable: true, type:
"NUMERIC(32,0)"),
                    ID_PERSONA_HISTORIAL_SIACPI = table.Column<decimal>(nullable:
true, type: "NUMERIC(32,0)"),
                    LUGAR_EVENTO = table.Column<string>(nullable: true, type:
"VARCHAR(200)"),
                    OBSERVACION = table.Column<string>(nullable: true, type:
"VARCHAR(500)")
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_T_PERSONA_HISTORIAL", x =>
x.ID_PERSONA_HISTORIAL);
                });
```

Tabla N° 4.27: Configuración para agregar nueva tabla al modelo de base de datos relacional (Elaboración propia)

## 4.3 SIMULACION DE RESULTADOS DEL COMPONENTE SOFTWARE

Configuración de un proyecto piloto para ver el comportamiento en los diferentes escenarios del desarrollo del software en el control de versiones del esquema de base de datos relacional.

En la Figura N° 4.25 se muestra una base de datos inicial sin ninguna tabla (base de datos vacía).

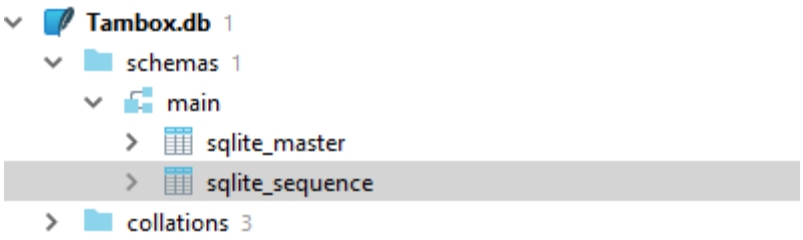



Figura N° 4.25: Base de datos inicial vacía (elaboración propia)

En la Figura N° 4.26 se muestra la tabla Sqlite_Master, esta tabla no cambia desde que se crea y la creación de la misma es por defecto, en esta tabla se definen las secuencias a usar de las llaves primarias.

| | type | name | tbl_name | rootpage | sql |
|---|---|---|---|---|---|
| 1 | table | sqlite_sequence | sqlite_sequence | 3 | CREATE TABLE sqlite_sequence(name,seq) |

Figura N° 4.26: Tabla Sqlite_Master (elaboración propia)

La tabla N° 4.28, muestra la clase o entidad (Persona) a ser creada en la base de datos como (T_PERSONA) usando el componente software de control de versiones del esquema de base de datos relacional.

```
[Table("T_PERSONA")]
    public class Persona
    {
        [Key]
        [Column("ID_PERSONA")]
        public int PersonaId { get; set; }

        [Column("NUMERO_DOCUMENTO")]
        public int NumeroDocumento { get; set; }

        [Column("NOMBRES")]
        public string Nombres { get; set; }

        [Column("APELLIDOS")]
        public string Apellidos { get; set; }

        [Column("ESTADO")]
        public bool Estado { get; set; }

        public ICollection<Usuario> Usuarios { get; set; }
    }
```

Tabla N° 4.28: Clase Modelo Persona (Elaboración propia)

La tabla N° 4.29, muestra a la clase o entidad (Usuario) a ser creada en la base de datos como (T_USUARIO) usando el componente software de control de versiones del esquema de base de datos relacional.

```
[Table("T_USUARIO")]
    public class Usuario
    {
        [Key]
        [Column("ID_USUARIO")]
        public int UsuarioId { get; set; }

        [Column("NOMBRE_USUARIO")]
        public string UserName { get; set; }

        [Column("PASSWORD")]
        public string Password { get; set; }

        [Column("ID_PERSONA")]
        public int PersonaId { get; set; }

        public Persona Persona { get; set; }
    }
```

Tabla N° 4.29: Clase Modelo Usuario (Elaboración propia)

Clase que se encarga de configurar la base de datos y generar la versión del esquema de base de datos relacional, en esta clase se muestra todos los atributos y modelos que se van crear en la base de datos tales como se muestran en la configuración, adicionalmente se podrá adicionar algunos cambios como agregar columnas, relaciones, otras tablas, entre otras operaciones soportadas por cada gestor de base de datos relacional, ver Tabla N° 4.30.

```
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "T_PERSONA",
                columns: table => new
                {
                    ID_PERSONA = table.Column<int>(nullable: false)
                        .Annotation("Sqlite:Autoincrement", true),
                    NUMERO_DOCUMENTO = table.Column<int>(nullable: false),
                    NOMBRES = table.Column<string>(nullable: true),
                    APELLIDOS = table.Column<string>(nullable: true),
                    ESTADO = table.Column<bool>(nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_T_PERSONA", x => x.ID_PERSONA);
                });

            migrationBuilder.CreateTable(
                name: "T_USUARIO",
                columns: table => new
                {
                    ID_USUARIO = table.Column<int>(nullable: false)
                        .Annotation("Sqlite:Autoincrement", true),
                    NOMBRE_USUARIO = table.Column<string>(nullable: true),
                    PASSWORD = table.Column<string>(nullable: true),
                    ID_PERSONA = table.Column<int>(nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_T_USUARIO", x => x.ID_USUARIO);
                    table.ForeignKey(
                        name: "FK_T_USUARIO_T_PERSONA_ID_PERSONA",
                        column: x => x.ID_PERSONA,
                        principalTable: "T_PERSONA",
                        principalColumn: "ID_PERSONA",
                        onDelete: ReferentialAction.Cascade);
                });

            migrationBuilder.CreateIndex(
                name: "IX_T_USUARIO_ID_PERSONA",
                table: "T_USUARIO",
                column: "ID_PERSONA");
        }
```

Tabla N° 4.30: Clase de configuración para crear el modelo de base de datos relacional (Elaboración propia)

Al ejecutar el aplicativo piloto, se muestra los modelos creados y que serán usado para realizar operaciones CRUD sobre la base de datos relacional (Figura N° 4.27), Adicionalmente se puede ver los puntos de Acceso (End

points) para realizar las pruebas necesarias sobre la entidad Persona (Figura N° 4.28).

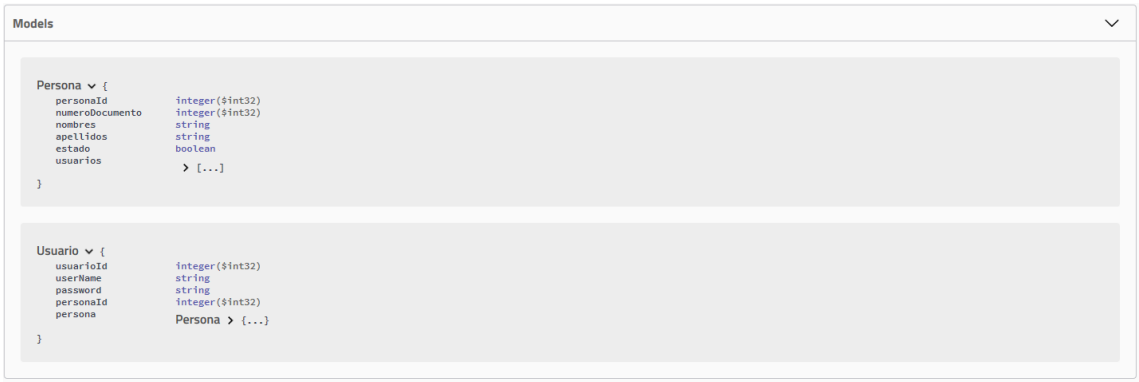



Figura N° 4.27: Modelo de datos (elaboración propia)

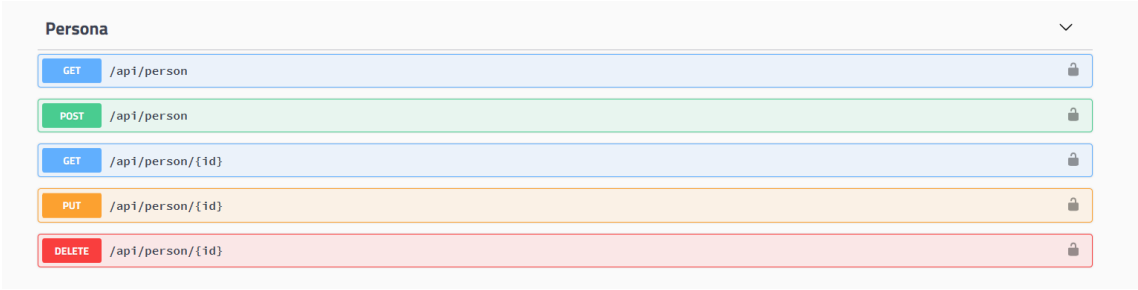



Figura N° 4.28: End points del aplicativo (elaboración propia)

El método Get es para consultar la lista de los registros de personas que existen en la base de datos relacional tal como se puede observar en la (Figura N° 4.29), para este punto aun no existes registros.

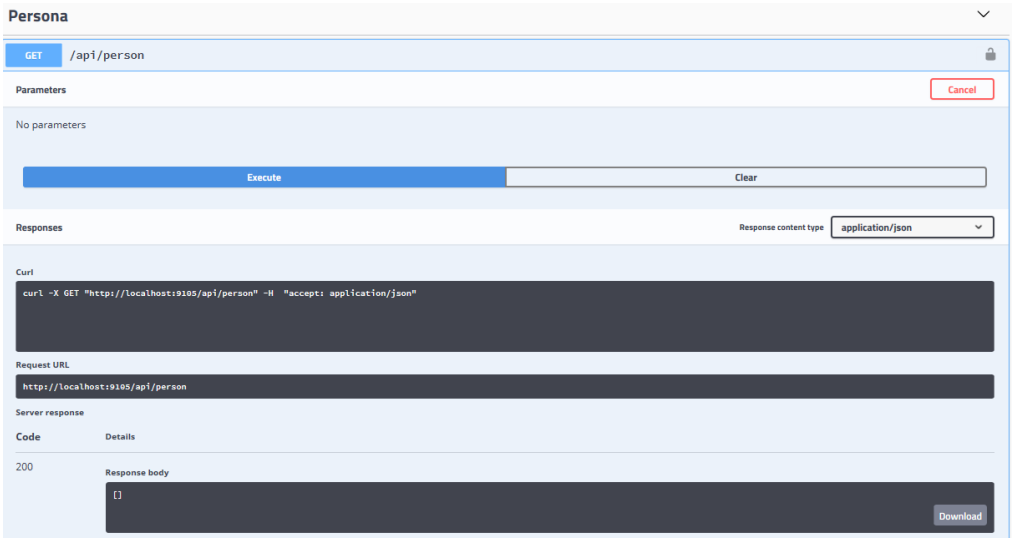



Figura N° 4.29: End Point Get del aplicativo (elaboración propia)

Base de datos después de ejecutar el proyecto piloto (Figura N° 4.30), en donde podemos ver que el esquema de base de datos cambio, mostrando los dos modelos reflejados en la base de datos, adicionalmente se agregó de forma automática una tabla para el control de cambios realizados a la base de datos.

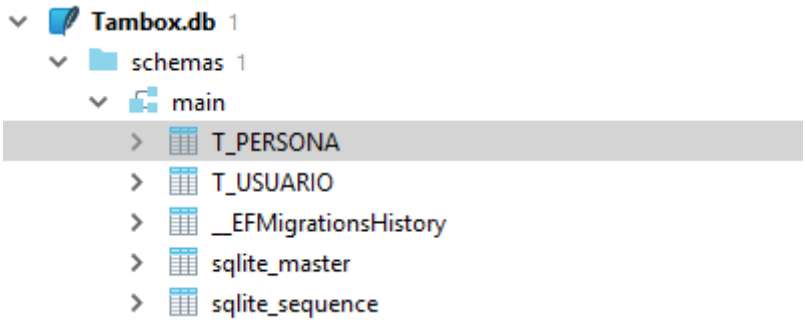



Figura N° 4.30: Base de datos inicial (elaboración propia)

En la Figura N° 4.31 se muestran los atributos de la base de datos tal como se configuró en el modelo de datos, donde podemos observar que tiene llaves primarias, y sus respectivas relaciones FK, adicionalmente a esto podemos observar que existe otra tabla llamada "_EFMigrationsHistory" en

donde almacena el historial de cambios del esquema de base de datos.

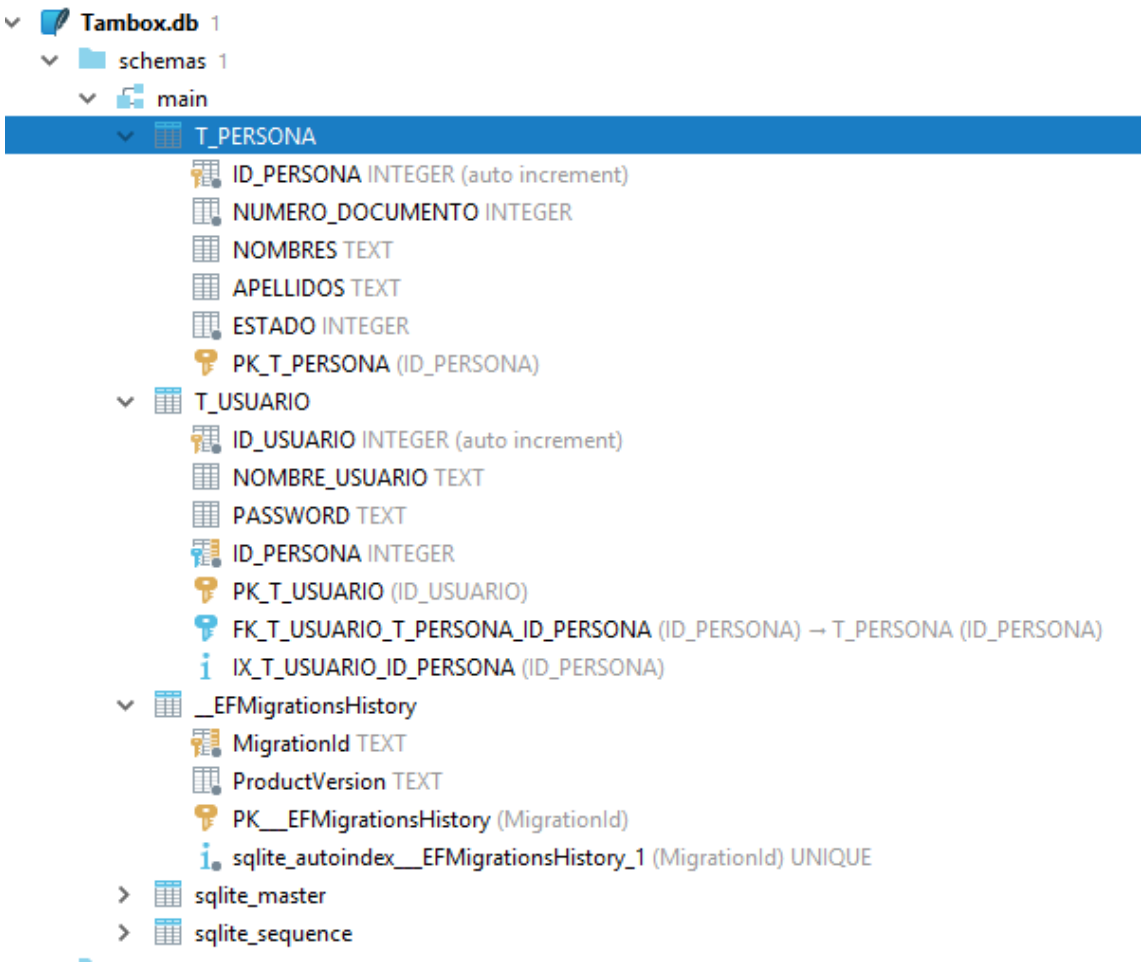



Figura N° 4.31: Base de datos con los atributos configurados en la aplicación (elaboración propia)

En la Figura N° 4.32, se muestra el historial de cambios, actualmente se encuentra un registro, esto indica que el proyecto está en la versión inicial.

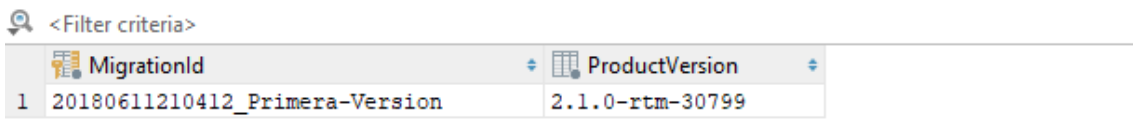



Figura N° 4.32: Tabla con historial de cambios del esquema de base de datos relacional (elaboración propia)

La Figura N° 4.33, muestra la creación de una persona desde el interfaz del proyecto piloto de prueba del componente.

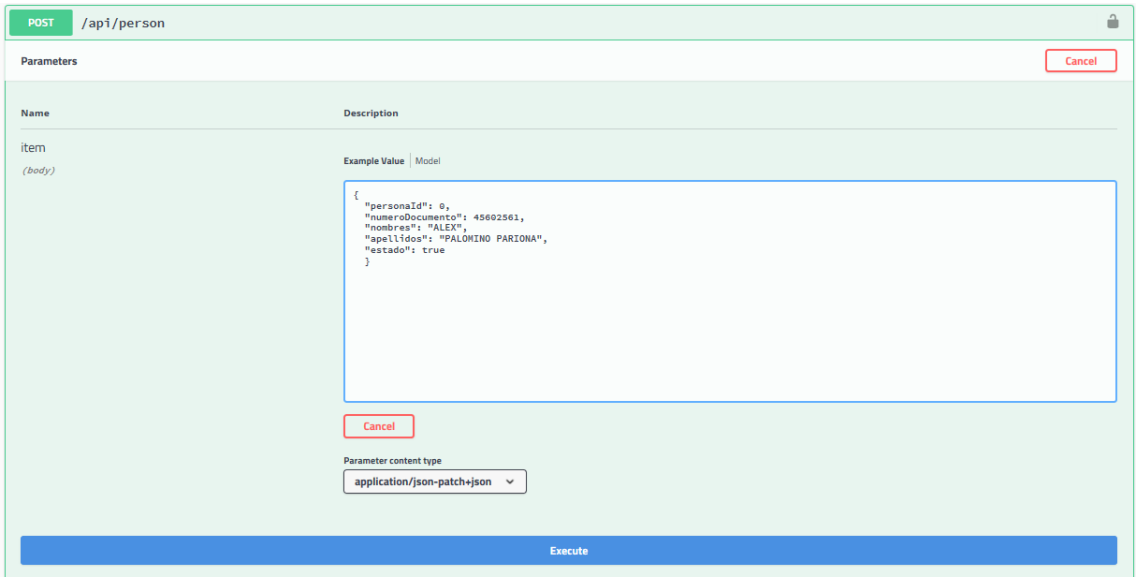



Figura N° 4.33: End Point de creación de una persona (elaboración propia)

La Figura N° 4.34, muestra el resultado de haber insertado el primer registro en la tabla T_PERSONA usando la clase modelo Persona.

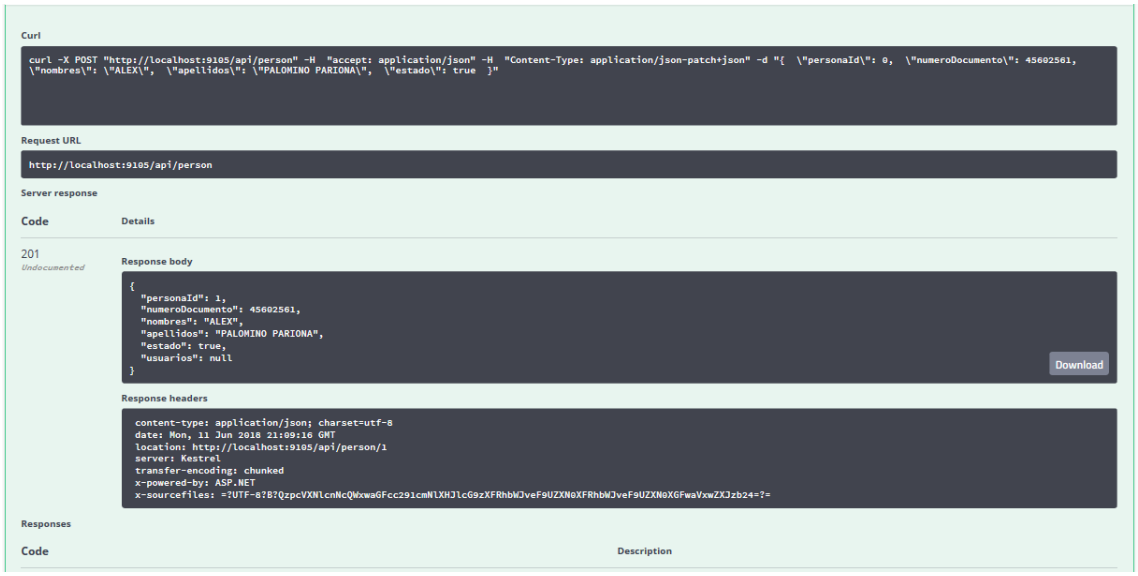



Figura N° 4.34: Resultado del primer registro a nivel del aplicativo (elaboración propia)

La Figura 4.35, muestra el primer registro realizado en la base de datos.

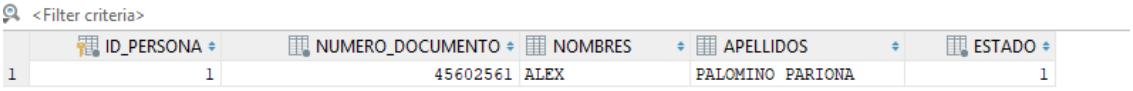



Figura N° 4.35: Tabla con registro de persona insertada desde el aplicativo (elaboración propia)

La tabla N° 4.31, muestra cómo crear o agregar una columna llamada Fecha nacimiento a la Tabla T_PERSONA, usando el modelo de datos Persona para ellos agregamos el siguiente código en el modelo Persona.

```
[Column("FECHA_NACIMIENTO")]
      [DataType(DataType.Date)]
      public DateTime FechaNacimiento { get; set; }
```

Tabla N° 4.31: Atributo de la persona que se va crear (Elaboración propia).

La tabla N° 4.32, muestra la configuración que realiza el componente software para poder agregar el atributo del modelo Persona en la base de datos relacional.

```
public partial class Add_FechaNacimiento_To_Persona
    {
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.AddColumn<DateTime>(
                name: "FECHA_NACIMIENTO",
                table: "T_PERSONA",
                nullable: false,
                defaultValue:    new    DateTime(1,    1,    1,    0,    0,    0,    0,
DateTimeKind.Unspecified));
        }
        protected override void Down(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.DropColumn(
                name: "FECHA_NACIMIENTO",
                table: "T_PERSONA");
        }
    }
```

Tabla N° 4.32: Clase configuración del componente (Elaboración propia)

Al ejecutar el proyecto el modelo de datos cambia automáticamente a como se ve en la Figura N° 4.36.

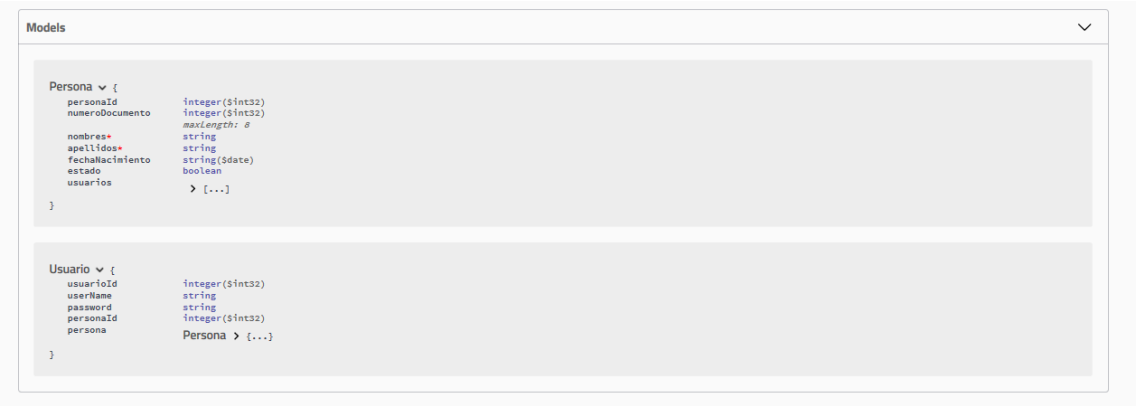



Figura N° 4.36: Modelo de datos en la vista (elaboración propia).

Al ingresar a la base de datos relacional podemos observar que el esquema de base de datos cambió, y se puede apreciar que se adicionó la columna FECHA_NACIMIENTO, ver (Figura N° 4.37).

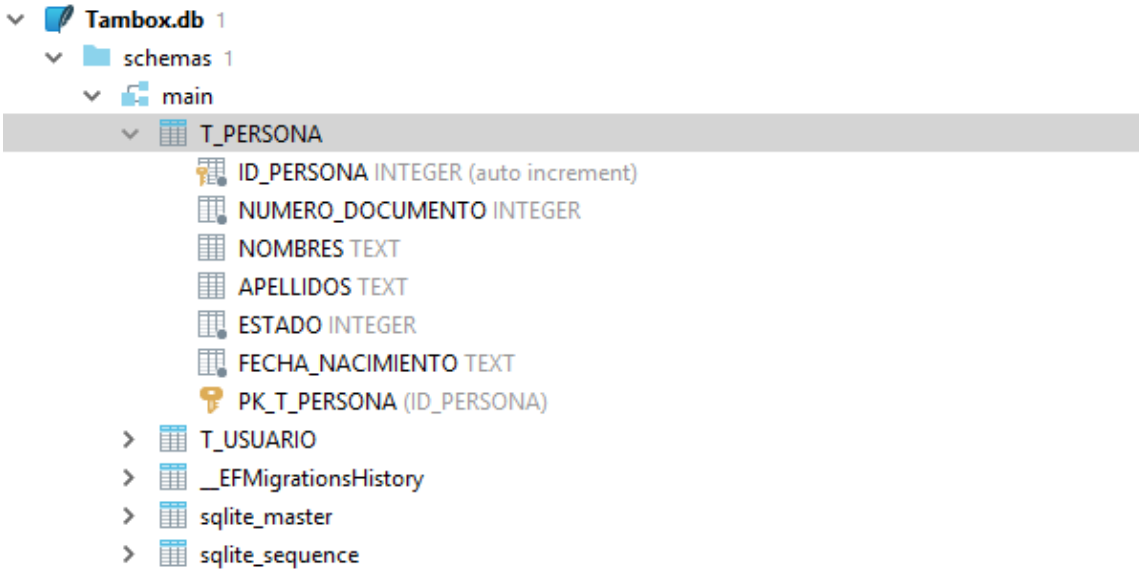



Figura N° 4.37: Modelo de base de datos alterado desde la aplicación (elaboración propia).

Consecuentemente podemos observar la tabla encargada de controlar las versiones del esquema de base de datos relacional cambió, ahora hay dos versiones de la base de datos como se ve en la (Figura N° 4.38).

<Filter criteria>

| | MigrationId | ProductVersion |
|---|---|---|
| 1 | 20180611210412_Primera-Version | 2.1.0-rtm-30799 |
| 2 | 20180611211826_Add_FechaNacimiento_To_Persona | 2.1.0-rtm-30799 |

Figura N° 4.38: Tabla que controla los cambios del esquema de base de datos relacional (elaboración propia)

Adicionalmente en la clase Persona procedemos agregar el atributo Teléfono ver Tabla N° 4.33.

```
[Column("TELEFONO")]
      public string Telefono { get; set; }
```

Tabla N° 4.33: Agregación del atributo al Modelo Persona (Elaboración propia)

Al ejecutar el componente software podemos observar que el control de versiones del esquema de base de datos sigue creciendo, ver (Figura N° 4.39).

<Filter criteria>

| | MigrationId | ProductVersion |
|---|---|---|
| 1 | 20180611210412_Primera-Version | 2.1.0-rtm-30799 |
| 2 | 20180611211826_Add_FechaNacimiento_To_Persona | 2.1.0-rtm-30799 |
| 3 | 20180611212528_Add_Telefono_To_Persona | 2.1.0-rtm-30799 |

Figura N° 4.39: Tabla que controla el esquema de base de datos relacional (elaboración propia)

Observamos que la Figura N° 4.40, muestra la base de datos relacional igual al modelo de datos que creamos en la aplicación, en este caso se agregó otro atributo "CORREO_ELECTRONICO", así mismo podemos observar que el resultado del componente en la tabla de control de versiones está actualizado como se ve en la (Figura N° 4.41), de tal modo el modelo de datos está actualizada en la aplicación, ver (Figura N° 4.42).


Figura N° 4.40: Esquema de base de datos final (elaboración propia)

<Filter criteria>

| | MigrationId | ProductVersion |
|---|---|---|
| 1 | 20180611213018_Primera-Version | 2.1.0-rtm-30799 |
| 2 | 20180611213045_FechaNacimiento_To_Persona | 2.1.0-rtm-30799 |
| 3 | 20180611213105_Add_Telefono_To_Persona | 2.1.0-rtm-30799 |
| 4 | 20180611213222_Add_Email_To_Persona | 2.1.0-rtm-30799 |

Figura N° 4.41: Tabla Final con el historial de cambios en el esquema de base de datos relacional (elaboración propia)

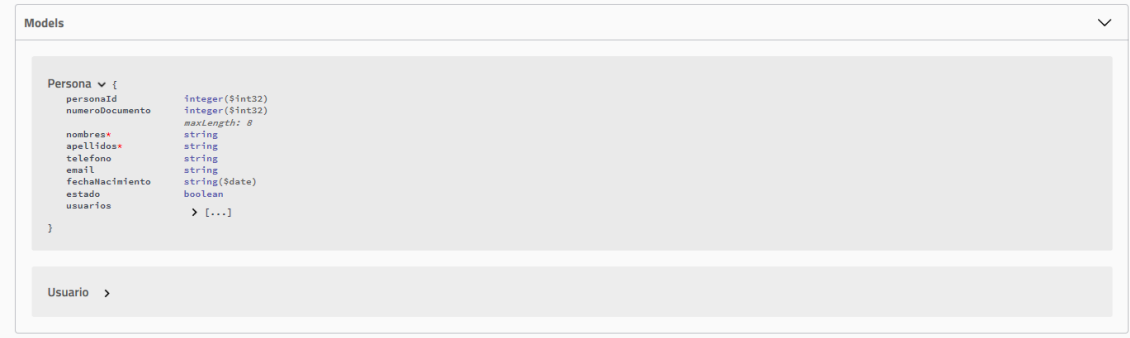



Figura N° 4.42: Resultado final del modelo de datos (elaboración propia)

## 4.4 ELABORACIÓN DE LAS TABLAS COMPARATIVAS DE EFICIENCIA

Se procede a formalizar las medidas comparativas en dos tablas elaboradas con los indicadores de eficiencia definida en la configuración y creación de esquema de base de datos relacional.

| N° | Indicador | Abreviatura | Unidad | Abreviatura |
|---|---|---|---|---|
| 1 | Tiempo en creación y configuración del esquema de base de datos. | TC | Milisegundos | ms |
| 2 | Líneas de código implementado | LC | Numero | # |
| 3 | Memoria usada por el proceso | MU | Megabyte | mb |

| | | | | |
|---|---|---|---|---|
| 4 | CPU usado por el proceso | CP | Porcentaje | % |
| 5 | Incompatibilidad de tipos de datos | ID | Si/No | s/n |
| 6 | Multiplataforma y transparencia del componente | MT | Si/No | s/n |

Tabla N° 4.34: Indicadores de eficiencia en creación y configuración de la base de datos relaciona (Elaboración propia).

| CREACIÓN Y CONFIGURACIÓN | | INDICADORES | | | | | |
|---|---|---|---|---|---|---|---|
| N° | Operaciones a la base de datos relacional | TC (ms) | LC (#) | MU (mb) | CP (%) | ID (s/n) | MT (s/n) |
| 1 | Creación de tablas | 150 | 30 | 5 | 5 | n | s |
| 2 | Creación de llaves primarias | 10 | 1 | 2 | 0.5 | n | s |
| 3 | Creación de llaves foráneas | 20 | 2 | 2 | 0.5 | n | s |
| 4 | Creación de tipos de datos | 30 | 1 | 3 | 0.5 | n | s |
| 5 | Creación de columnas | 40 | 3 | 1 | 0.8 | n | s |
| 6 | Eliminación de columnas | 60 | 1 | 2 | 1 | n | s |

Tabla N° 4.35: Medias de indicadores del componente software desarrollado para controlar las versiones del esquema de base de datos relacional (Elaboración propia).

| CREACIÓN Y CONFIGURACIÓN | | INDICADORES | | | | | |
|---|---|---|---|---|---|---|---|
| N° | Operaciones a la base de datos relacional | TC (ms) | LC (#) | MU (mb) | CP (%) | ID (s/n) | MT (s/n) |
| 1 | Creación de tablas | 800 | 30 | 10 | 10 | s | n |
| 2 | Creación de llaves primarias | 100 | 3 | 5 | 5 | n | n |
| 3 | Creación de llaves foráneas | 200 | 3 | 5 | 5 | s | n |
| 4 | Creación de tipos de datos | 80 | 2 | 5 | 5 | s | n |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 5 | Creación de columnas | 50 | 2 | 5 | 5 | s | n |
| 6 | Eliminación de columnas | 80 | 1 | 5 | 5 | n | n |

Tabla N° 4.36: Medias de indicadores del delta scripts para controlar las versiones del esquema de base de datos relacional (Elaboración propia).

## 4.5 ANÁLISIS ESTADÍSTICO EN EL DESPLIEGUE DEL COMPONENTE SOFTWARE

### 4.5.1 PRUEBA PARA EL TIEMPO EN CREACIÓN Y CONFIGURACIÓN DEL ESQUEMA DE BASE DE DATOS RELACIONAL (TC)

Para las muestras del indicador tiempo en creación y configuración del esquema de base de datos relacional, del componente software denominado control de versiones del esquema de base de datos relacional y la más usada denominada delta scripts o script de cambios de base de datos se aplica la diferencia de medias y el estadístico t-stutent puesto que las muestras son menores o iguales 30 y no se conoce la varianza poblacional.

Datos tomados de las tablas N° 4.34 y 4.35.

Tamaño de la muestra:

$n_1 = 6$ Tiempo en creación del esquema de base de datos relacional con componente software.

$n_2 = 6$ Tiempo en creación del esquema de base de datos relacional con delta scripts.

Para el cálculo se utilizó la herramienta análisis de datos de Excel con un nivel de confianza de 95% y 5 grados de libertad de un total de 6 muestras.

| | Componente software | delta scripts |
|---|---|---|
| Media | 51.66666667 | 218.3333333 |
| Varianza | 2616.666667 | 83856.66667 |
| Observaciones | 6 | 6 |

| | | |
|---|---|---|
| Diferencia hipotética de las medias | 0 | |
| Grados de libertad | 5 | |
| **Estadístico t** | **-1.388299843** | |
| P(T<=t) una cola | 0.111860282 | |
| Valor crítico de t (una cola) | 2.015048373 | |
| P(T<=t) dos colas | 0.223720563 | |
| Valor crítico de t (dos colas) | 2.570581836 | |

Tabla N° 4.37: Cálculo de t student para la creación y configuración del esquema de base de datos relacional (Elaboración propia).

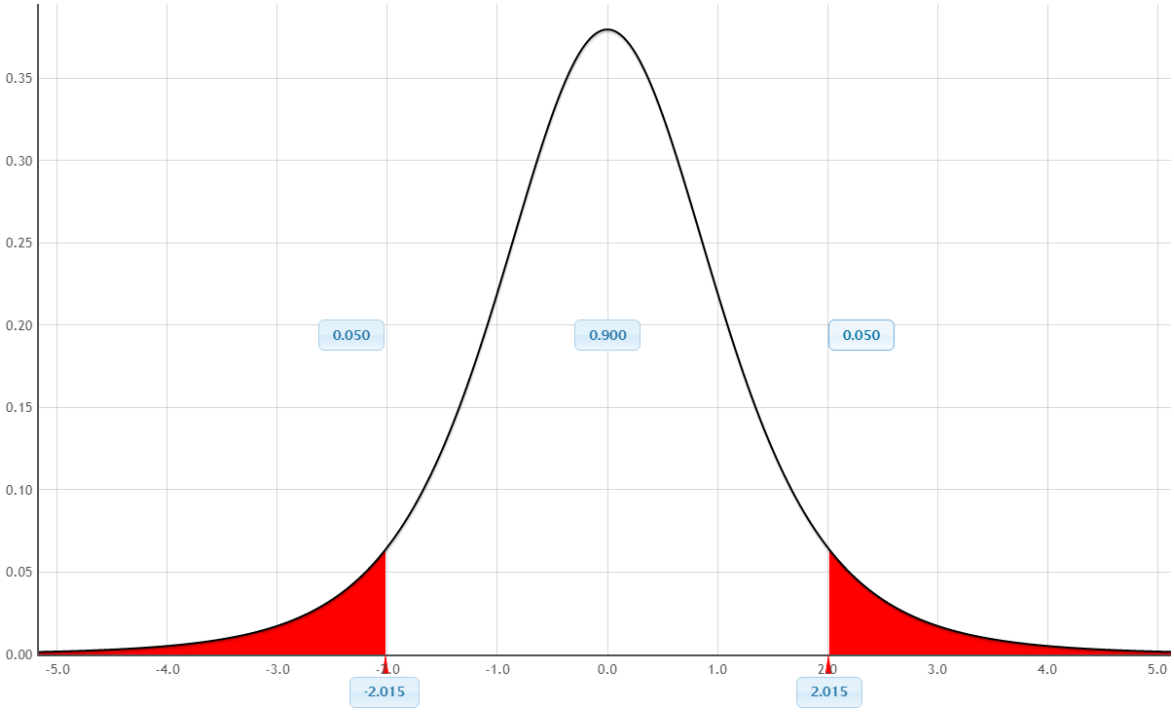



Figura N° 4.43: Curva t-student con 5 grados de libertad para TC.

Según la muestra, se usa el estadístico t con 5 grados de libertad y un nivel de confianza del 95%, $\mu_m \leq \mu_h$, los tiempos de creación y configuración del esquema de base de datos relacional de las aplicaciones que utiliza el componente software para controlar las versiones del esquema de base de

datos relacional es significativamente inferior a los que utilizan los deltas scripts.

### 4.5.2 PRUEBA DE LAS LÍNEAS DE CÓDIGO IMPLEMENTADO (LC)

Para las muestras del indicador líneas de código implementado en creación y configuración del esquema de base de datos relacional para el componente software denominada control de versiones del esquema de base de datos relacional, se aplica la diferencia de medias y el estadístico t-stutent, puesto que las muestras son memores o iguales 30 y no se conoce la varianza poblacional.

Datos tomados de las tablas N° 4.34 y 4.35.

Tamaño de la muestra:

$n_1 = 6$ Tiempo en creación del esquema de base de datos relacional con componente software.

$n_2 = 6$ Tiempo en creación del esquema de base de datos relacional con delta scripts.

Para el cálculo se utilizó la herramienta análisis de datos de Excel con un nivel de confianza de 95% y 10 grados de libertad de un total de 6 muestras.

| | Componente software | delta scripts |
|---|---|---|
| Media | 6.333333333 | 6.833333333 |
| Varianza | 135.0666667 | 129.3666667 |
| Observaciones | 6 | 6 |
| Diferencia hipotética de las medias | 0 | |
| Grados de libertad | 10 | |
| **Estadístico t** | **-0.075316049** | |
| P(T<=t) una cola | 0.470724337 | |
| Valor crítico de t (una cola) | 1.812461123 | |

| | | |
|---|---|---|
| P(T<=t) dos colas | 0.941448675 | |
| Valor crítico de t (dos colas) | 2.228138852 | |

Tabla N° 4.38: Cálculo de t-student para líneas de código implementado (Elaboración propia).

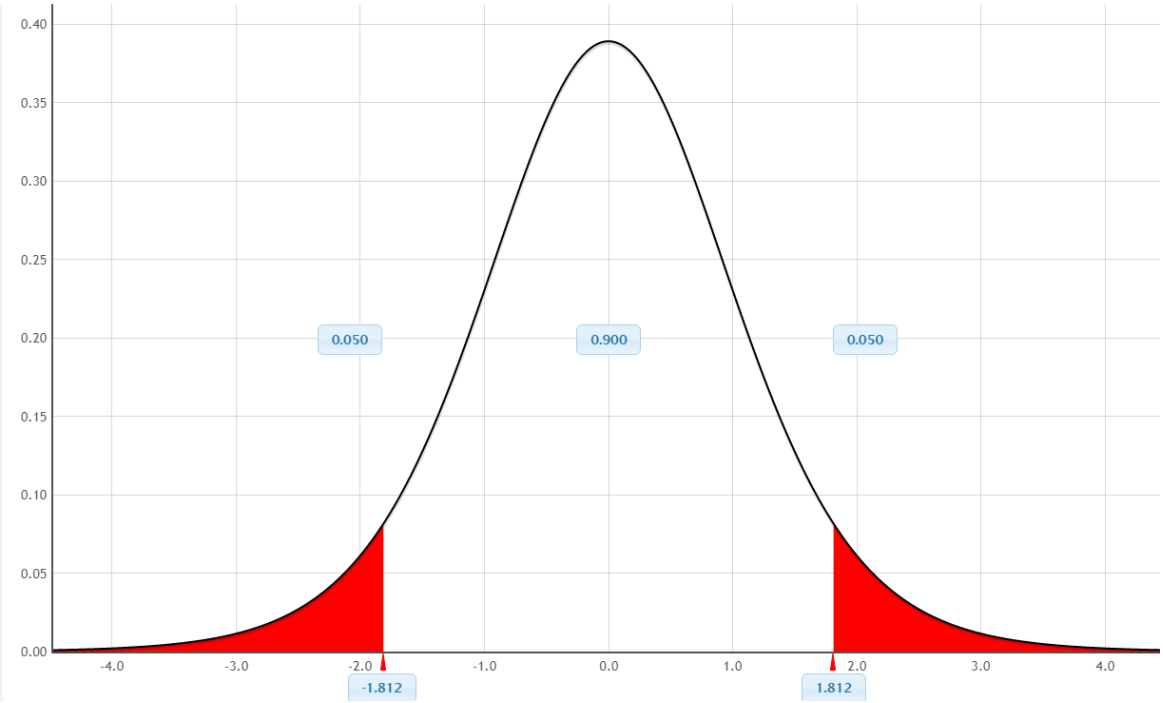



Figura N° 4.44: Curva t-student con 10 grados de libertad para LC.

Según la muestra, se usa el estadístico t con 10 grados de libertad y un nivel de confianza del 95%, $\mu_m \leq \mu_h$, las líneas de código implementado para la creación y configuración del esquema de base de datos relacional de las aplicaciones que utiliza el componente software para controlar las versiones del esquema de base de datos relacional es significativamente inferior a los que utilizan los deltas scripts.

### 4.5.3 PRUEBA PARA MEMORIA USADO POR EL PROCESO (MU)

Para las muestras del indicador memoria utilizada del componente software denominado control de versiones del esquema de base de datos

relacional, se aplica la diferencia de medias y el estadístico t-stutent puesto que las muestras son memores o iguales a 30 y no se conoce la varianza poblacional.

Datos tomados de las tablas N° 4.34 y 4.35.

Tamaño de la muestra:

$n_1 = 6$ Tiempo en creación del esquema de base de datos relacional con componente software.

$n_2 = 6$ Tiempo en creación del esquema de base de datos relacional con delta scripts.

Para el cálculo se utilizó la herramienta análisis de datos de Excel con un nivel de confianza de 95% y 9 grados de libertad de un total de 6 muestras.

| | Componente software | delta scripts |
|---|---|---|
| Media | 2.5 | 5.833333333 |
| Varianza | 1.9 | 4.166666667 |
| Observaciones | 6 | 6 |
| Diferencia hipotética de las medias | 0 | |
| Grados de libertad | 9 | |
| **Estadístico t** | **-3.314967721** | |
| P(T<=t) una cola | 0.00450614 | |
| Valor crítico de t (una cola) | 1.833112933 | |
| P(T<=t) dos colas | 0.00901228 | |
| Valor crítico de t (dos colas) | 2.262157163 | |

Tabla N° 4.39: Cálculo de t-student para memoria usado por el proceso (Elaboración propia).

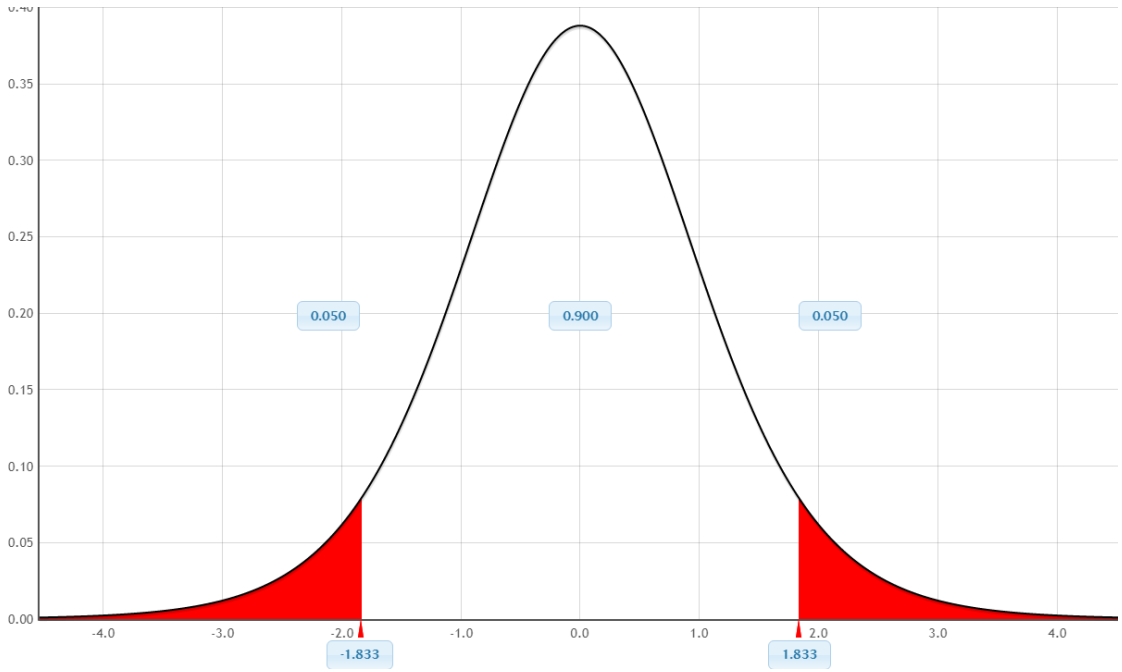



Figura N° 4.45: Curva t-student con 9 grados de libertad para MU.

Según la muestra, se usa el estadístico t con 9 grados de libertad y un nivel de confianza del 95%, $\mu_m \leq \mu_h$, la memoria usada por el proceso para la creación del esquema de base de datos relacional de las aplicaciones que utiliza el componente software para controlar las versiones del esquema de base de datos relacional es significativamente inferior a los que utilizan los deltas scripts.

### 4.5.4 PRUEBA PARA EL CPU USADO POR EL PROCESO (CP)

Para las muestras del indicador CPU utilizada por componente software denominado control de versiones del esquema de base de datos relacional, se aplica la diferencia de medias y el estadístico t-stutent puesto que las muestras son memores o iguales a 30 y no se conoce la varianza poblacional.

Datos tomados de las tablas N° 4.34 y 4.35.

Tamaño de la muestra:

$n_1 = 6$ Tiempo en creación del esquema de base de datos relacional con componente software.
$n_2 = 6$ Tiempo en creación del esquema de base de datos relacional con delta scripts.

Para el cálculo se utilizó la herramienta análisis de datos de Excel con un nivel de confianza de 95% y 10 grados de libertad de un total de 6 muestras.

| | Componente software | delta scripts |
|---|---|---|
| Media | 1.383333333 | 5.833333333 |
| Varianza | 3.181666667 | 4.166666667 |
| Observaciones | 6 | 6 |
| Diferencia hipotética de las medias | 0 | |
| Grados de libertad | 10 | |
| **Estadístico t** | **-4.021066097** | |
| P(T<=t) una cola | 0.001217134 | |
| Valor crítico de t (una cola) | 1.812461123 | |
| P(T<=t) dos colas | 0.002434267 | |
| Valor crítico de t (dos colas) | 2.228138852 | |

Tabla N° 4.40: Cálculo de t-student para el CPU usado por el proceso (Elaboración propia).

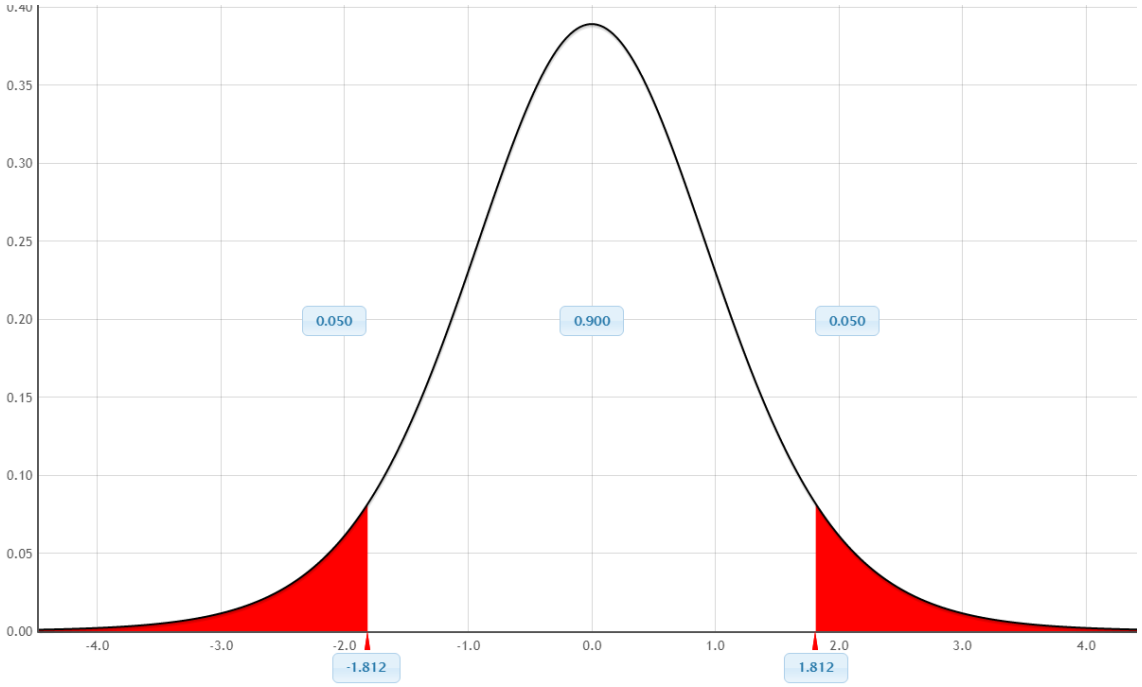



Figura N° 4.46: Curva t-student con 9 grados de libertad para CP.

Según la muestra, se usa el estadístico t con 9 grados de libertad y un nivel de confianza del 95%, $\mu_m \leq \mu_h$, el CPU usado por el proceso para la creación del esquema de base de datos relacional de las aplicaciones que utiliza el componente software para controlar las versiones del esquema de base de datos relacional es significativamente inferior a los que utilizan los deltas scripts.

### 4.5.5 PRUEBA PARA INCOMPATIBILIDAD DE TIPOS DE DATOS (ID)

Para las muestras del indicador incompatibilidad de tipo de dato en la creación y configuración del esquema de base de datos relaciona del componente software denominado control de versiones del esquema de base de datos relacional. Se utiliza el estadístico ji-cuadrado, puesto que las muestras son menores o iguales a 30, no paramétricas y cualitativas.

Datos tomados de las tablas N° 4.34 y 4.35.

| Incompatibilidad de Tipo de Dato | | | |
|---|---|---|---|
| **Componente software** | Si | No | **Total** |
| Control de versiones del esquema de base de datos relacional. | 0 | 6 | **6** |
| Scripts de creación del base de datos(delta scripts). | 4 | 2 | **6** |
| **Total** | **4** | **8** | **12** |

Tabla N°4.41: Tabla de frecuencia observada de ID.

Calculamos los valores estimados:

$fe_{11} = \frac{6 \times 4}{12} = 2$ $fo_{12} = \frac{6 \times 8}{12} = 4$ $fo_{21} = \frac{6 \times 4}{12} = 2$ $fo_{22} = \frac{6 \times 8}{12} = 4$

| Incompatibilidad de Tipo de Dato | | | |
|---|---|---|---|
| **Componente software** | Si | No | **Total** |
| Control de versiones del esquema de base de datos relacional. | 2 | 4 | **6** |
| Scripts de creación del base de datos(delta scripts). | 2 | 4 | **6** |
| **Total** | **4** | **8** | **12** |

Tabla N°4.42: Tabla de frecuencia esperada de ID.

$$X^2 = \sum_{ij} \frac{(fo_{ij} - fe_{ij})^2}{fe_{ij}} = \frac{(0-2)^2}{2} + \frac{(6-4)^2}{4} + \frac{(4-2)^2}{2} + \frac{(2-4)^2}{4}$$

$X^2 = 2 + 1 + 2 + 1 = 6$

Calculamos los grados de libertad, región crítica para un nivel de confianza del 95%:

gl $= (2-1) * (2-1) = 1$ y el nivel de confianza $\alpha = 0.05$

ji-cuadrado para la región crítica es 3.8415

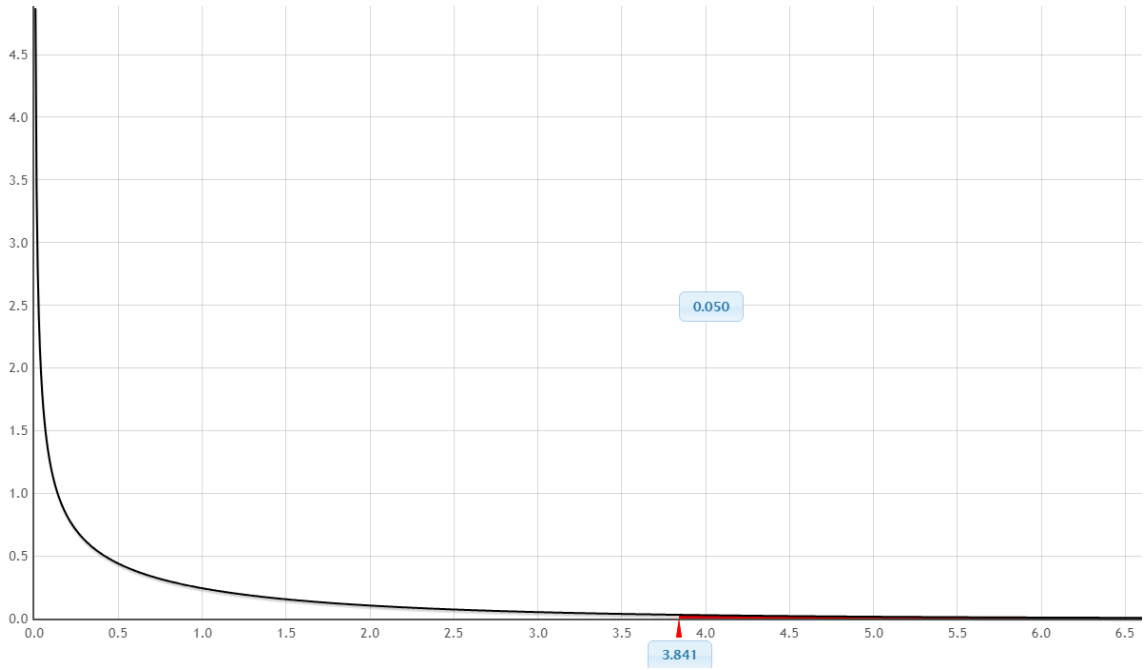



Figura N° 4.47: Curva ji-cuadrada con 1 grado de libertad para ID.

Según la muestra se usa $x^2$, ji-cuadrada, con un grado de libertad, las aplicaciones desarrolladas con el control de versiones del esquema de base de datos relacional tienen significativamente menor incompatibilidad de tipos de datos en la creación y configuración del esquema de base de datos relacional que con la utilización de delta scripts o script de creación y/o actualización de esquema de base de datos relacionales.

### 4.5.6 PRUEBA PARA MULTIPLATAFORMA Y TRANSPARENCIA DEL COMPONENTE (MT)

Para las muestras del indicador multiplataforma y transparencia del componente en la creación y configuración del esquema de base de datos relacional del componente software desarrollado denominado control de versiones del esquema de base de datos relacionales.

Se utiliza el estadístico ji-cuadrado, puesto que las muestras son menores o iguales a 30, no paramétricas y cualitativas.

Datos tomados de la tabla N° 4.34 y 4.35.

| Multiplataforma y Transparencia del Componente | | | |
|---|---|---|---|
| **Componente software** | Si | No | **Total** |
| Control de versiones del esquema de base de datos relacional. | 6 | 0 | **6** |
| Scripts de creación del base de datos(delta scripts). | 0 | 6 | **6** |
| **Total** | **6** | **6** | **12** |

Tabla N°4.43: Tabla de frecuencia observada de MT.

Planteamos las hipótesis:

Calculamos los valores estimados:

$$fe_{11} = \frac{6 \times 6}{12} = 3 \qquad fo_{12} = \frac{6 \times 6}{12} = 3 \qquad fo_{21} = \frac{6 \times 6}{12} = 3 \qquad fo_{22} = \frac{6 \times 6}{12} = 3$$

| Incompatibilidad de Tipo de Dato | | | |
|---|---|---|---|
| **Componente software** | Si | No | **Total** |
| Control de versiones del esquema de base de datos relacional. | 3 | 3 | **6** |
| Scripts de creación del base de datos(delta scripts). | 3 | 3 | **6** |
| **Total** | **6** | **6** | **12** |

Tabla N°4.44: Tabla de frecuencia esperada de MT.

$$X^2 = \sum_{ij} \frac{(fo_{ij} - fe_{ij})^2}{fe_{ij}} = \frac{(6-3)^2}{3} + \frac{(0-3)^2}{3} + \frac{(0-3)^2}{3} + \frac{(6-3)^2}{3}$$

$$X^2 = 3 + 3 + 3 + 3 = 12$$

Calculamos los grados de libertad, región crítica para un nivel de confianza del 95%:

$\mathrm{gl} = (2-1) * (2-1) = 1$ y el nivel de confianza $\alpha = 0.05$

ji-cuadrado para la región crítica es 3.8415

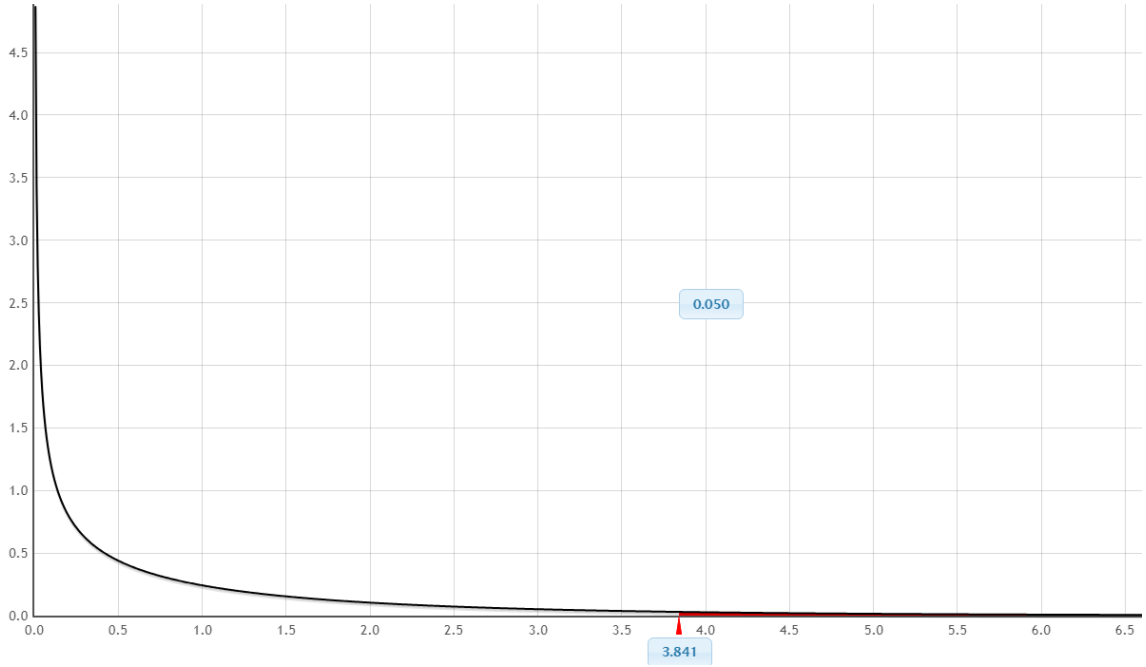



Figura N° 4.48: Curva ji-cuadrada con 1 grado de libertad para MT.

Según la muestra se usa $x^2$, ji-cuadrada, con un grado de libertad, las aplicaciones desarrolladas con el control de versiones del esquema de base de datos relacional tienen significativamente mayor adaptabilidad al entorno multiplataforma y transparencia en la creación y configuración del esquema de base de datos relacional que con la utilización de delta scripts o script de creación y/o actualización de esquema de base de datos relacionales.

## 4.6 DISCUSIÓN

Acorde al análisis del resultado y el análisis estadístico t con 5 grados de libertad y un nivel de confianza del 95%, $\mu_m \leq \mu_h$, los tiempos de creación

y configuración del esquema de base de datos relacional de las aplicaciones que utiliza el componente software para controlar las versiones del esquema de base de datos relacional es significativamente inferior a los que utilizan los deltas scripts, sin embargo Tuya (2007), afirman que las pruebas para determinar si los tiempos de respuesta del sistema tanto en condiciones normales como en condiciones especiales usando el frameworks, se encuentran dentro de los limites predefinidos. Por consiguiente, bajo este sustento afirmamos que el tiempo de creación y configuración del esquema de base de datos relacional es inferior, de esta manera mejoramos la creación del esquema de base de datos relacional.

De la misma forma el análisis del resultado y el análisis estadístico t-student con 10 grados de libertad y un nivel de confianza del 95%, $\mu_m \leq \mu_h$, las líneas de código implementado para la creación y configuración del esquema de base de datos relacional de las aplicaciones que utiliza el componente software para controlar las versiones del esquema de base de datos relacional es significativamente inferior a los que utilizan los deltas scripts, sin embargo O'Neil (2008), sostiene que los ORMs se han convertido en una práctica necesaria en la manipulación de objetos entre la memoria y su ubicación real en la base de datos, permitiendo al desarrollador definir de forma clara la gestión entre el modelo de objetos y el esquema de base de datos, además de expresar las operaciones de esta última en términos de objetos. Por consiguiente, bajo este sustento contrastado, afirmamos que la cantidad de código implementado para la creación y configuración del esquema de base de datos relacional es significativamente inferior a los que utilizan los deltas scripts.

De la misma forma el análisis de resultados y el análisis estadístico t-student con 9 grados de libertad y un nivel de confianza del 95%, $\mu_m \leq \mu_h$, la

memoria usada por el proceso para la creación del esquema de base de datos relacional de las aplicaciones que utiliza el componente software para controlar las versiones del esquema de base de datos relacional es significativamente inferior a los que utilizan los deltas scripts, así mismo Nolte (1995), sostiene que en términos generales de sistemas de información, el rendimiento es considerado como la reducción de uso de la CPU, Memoria y la búsqueda de métodos más óptimos para el sistema. Por consiguiente, estos sustentos nos ayudan a confirmar que el uso del componente software para la creación y configuración del esquema de base de datos relacional tiene un uso considerablemente menor de la memoria.

De la misma forma el análisis de resultados y el análisis estadístico t-student con 9 grados de libertad y un nivel de confianza del 95%, $\mu_m \leq \mu_h$, el CPU usado por el proceso para la creación del esquema de base de datos relacional de las aplicaciones que utiliza el componente software para controlar las versiones del esquema de base de datos relacional es significativamente inferior a los que utilizan los deltas scripts, así mismo Nolte (1995), sostiene que en términos generales de sistemas de información, el rendimiento es considerado como la reducción de uso de la CPU, Memoria y la búsqueda de métodos más óptimos para el sistema. Por consiguiente, estos sustentos y las pruebas realizadas nos ayudan a confirmar que el uso del componente software para la creación y configuración del esquema de base de datos relacional tiene un uso considerablemente menor de la CPU.

Del mismo modo, si comparamos los resultados y el análisis estadístico del indicador cualitativo con el estadístico $x^2$, ji-cuadrada, con un grado de libertad y un nivel de confianza de 95%, indicamos que las aplicaciones desarrolladas con el control de versiones del esquema de base de datos relacional tienen significativamente menor incompatibilidad de tipos de

datos en la creación y configuración del esquema de base de datos relacional que con la utilización de delta scripts o script de creación y/o actualización de esquema de base de datos relacionales. Así como afirma que existen inconvenientes en los tipos de datos de los diferentes DBMS a pesar de manejar el estándar SQL-92 como los campos numéricos y fechas generando incompatibilidad de las aplicaciones (Velasco, 2010). Bajo este sustento concluimos que con el uso del componente software de control de versiones del esquema de base de datos relacional ayudamos a mejorar la creación y configuración de los esquemas de bases de datos relacionales de forma transparente a los diversos tipos de datos en los DBMS.

Finalmente si comparamos los resultado y el análisis estadístico del indicador cualitativo con el estadístico $x^2$, ji-cuadrada, con un grado de libertad y nivel de confianza de 95%, indicamos que, las aplicaciones desarrolladas con el control de versiones del esquema de base de datos relacional tienen significativamente mayor adaptabilidad al entorno multiplataforma y transparencia en la creación y configuración del esquema de base de datos relacional que con la utilización de delta scripts o script de creación y/o actualización de esquema de base de datos relacionales. Así mismo Velasco (2010), concluye que, con una interfaz especializada es posible asegurar la trasparecía y adaptabilidad a un entorno de sistemas de base de datos. Con este sustento terminamos afirmando que con el componente software de control de versiones del esquema de base de datos relacionales aseguramos esta transparencia y adaptabilidad a entornos cambiantes y bases de datos distribuidas geográficamente.

# CAPÍTULO V
# CONCLUSIONES Y RECOMENDACIONES

## 5.1 CONCLUSIONES

a) Se logró desarrollar el componente software mediante técnicas e instrumentos plasmadas en el capítulo III sección 3.5, la metodología ágil scrum plasmada en el capítulo IV sección 4.2, control de versiones y patrón code first de entity framework 6 plasmada en el capítulo IV sección 4.3, para controlar las versiones del esquema de base de datos relacional.

b) Se logró analizar y desinar la capa de modelo de datos para configurar las versiones del esquema de base de datos relacional, en base a al código fuente del componente que están plasmados en las tablas N° 4.21, 4.22, 4.23 y 4.24, estas tablas muestran cómo se deberían configurar las clases modelo para que la estructura de del modelo de base de datos no tenga errores.

c) Se logró implementar y probar la capa de contexto para administrar las versiones del esquema de base de datos relacional, que está plasmada en la Tabla N° 4.28, la configuración realizada por el componente está en la Tabla N° 4.30, y el resultado se muestra en la Figura N° 4.31 y 4.41.

## 5.2 RECOMENDACIONES

a) Se debe analizar la manera de crear los comandos necesarios para trabajar de manera más aislada con el control de versiones, dicho de otra manera, buscar la forma en la que el componente software de control de versiones pueda crear, trabajar e implementar copias locales de bases de datos para eliminar la necesidad de mantener conexión directa con la base de datos.

b) Analizar la manera y el procedimiento necesario para crear una herramienta gráfica que sin disminuir la autonomía (sin necesitar la integración o intervención de otras herramientas) y funcionalidad de componente software permita la conexión y administre los cambios de los diferentes tipos de bases de datos.

# BIBLIOGRAFÍA


Bauer, C., & King, G. (2007). *Java Persistence with Hibernate*. Greenwich: Manning Publications.

Berlack, R. H. (1992). *Software configuration management*. New York: John Wiley & Sons.

Bernal, T. C. (2006). *Metodologia de la investigacion: para administracion, humanidades y ciencias sociales*. Mexico: Pearson Educacion de Mexico.

Borrell, G. (2006). *Control de versiones*. España.

Callejas, M., Peñalosa, D. I., & Alacón, A. C. (2011). *Evaluación y análisis de rendimiento de los frameworks de persistencia (Tesis de Maestría)*. Universidad de Manizales. Colombia.

Cardelli, L., & Wegner, P. (1985). *On understanding types, data abstraction, and polymorphism*. AT&T Bell Laboratories. Estados Unidos: Murray Hill.

Cobo, A. (n.d.). Base de datos relacionales: Teoría y práctica (1ª ed.). Madrid, España: s.f.

Collins Sussman, B., Fitzpatrick, B. W., & Pilato, M. (2008). *Version control with subversion*. O'Reilly Media.

Deemer, P., Benefield, G., Larman, C., & Vodde, B. (2009). *The Scrum primer. Certified Scrum Training Worldwide*. Agile-Spain.

Díaz, M., & Del Dago, S. (2008). *Educación a Distancia en el Nivel Superior: Un Análisis sobre las Prácticas de Evaluación de los Aprendizajes. Anuales del Encuentro Internacional BTM 2008, Educación, Formación y Nuevas Tecnologías*. Punta del Este, Uruguay.

Downs, K. (2008). *Database development: table structure changes*.

Elmasri, R., & Navathe, S. B. (2000). *Sistemas de bases de datos. (7a ed.)*. México: Addison Wesley.

Elmasri, R., & Navathe, S. B. (2007). *Fundamentals of database systems. (5ª ed.)*. Addison Wesley.

Escobar, J., Losavio, F., & Ortega, D. (2012). *Una revisión de Frameworks, Lenguajes de Modelado y Herramientas para Arquitecturas Empresariales.*

Evans, E. J. (2003). *Domain drive design: Tackling complexity in the heart of software. (1ª ed.).* Massachusetts: Addison Wesley.

Evans, M. (2003). *"Testing Software Patterns". Microsoft Patterns & Practices.*

Everett, G., & Raymond, M. (2007). *Software testing, testing across the entire software development life cycle.* John Wiley & Sons.

Fidias, G. A. (2006). *El Proyecto de Investigación, introducción a la metodología científica 5ta edición.* Caracas, Venezuela: Episteme.

Griffin, C. (2013). *A Comparative look at entity framework code first. Mathematics and computer science capstones.* La Salle University: Digital Commons.

Gueddari, M. (2014). *Managing DbContext the right way with Entity Framework 6: an in-depth guide.*

Hanley, J. (2015). *Scrum: Your Quick Start Guide To Adopting Scrum For Your Organization.*

Hayase, Y., Matsushita, M., & Inoue, K. (2005). *Revision control system using delta script of syntax tree. Proceedings of the 12th international workshop on Software configuration management.*

Hernández, M. (2000). *Metodología de la investigación – Tipos y Niveles de Investigación.* Maracaibo, Venezuela.

Hernández, R., Fernández, C., & Baptista, P. (2006). *Metodología de la Investigación. 4ta. ed.* McGraw-Hill.

Hummel, J. (2007). *LINQ: The future of data access in C# 3.0.* Washington: O'Reilly Media.

IBM Corporation, Relational Databases and Object Orientation (Rational Unified Process 10 15, 2017).

Kent, B. (2001). *Principios de manifiesto ágil.*

Keyes, J. (2004). *Software configuration management.* CRC Press.

Kniberg, H. (2007). *Prólogo de Jeff Sutherland y Mike Cohn. "Scrum y XP desde las trincheras".*

Korth, H. F., & Silberschatz, A. (2002). *Fundamentos de bases de datos.* Madrid: McGraw-Hill.

Krutchen, P., & Kroll, P. (2002). *Rational Unified Process. (2 ª ed.).* Addison Wesley.

Microsoft. (2014). *Entity Framework*. Obtenido de https://msdn.microsoft.com/es-es/library/gg696172(v=vs.103).aspx

Miguel, M., & Piatthini, M. (1993). *Concepción y diseño de bases de datos: del modelo E/R al modelo relacional.*

Mindy, S., & Erick, T. (2011). *Desarrollo de un sistema web basado en ADO.NET para el control de la gestión de inscripción y logística de cursos.* Caracas.

MSDN. (2014). *ADO.NET Entity Framework*. Obtenido de https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ef/overview

Münch, J. (2010). *New Modelling Concepts for Today´s Software Prosses. Paderbom: Proceedings.*

Nolte, J. (1995). *Towards a Persistence Framework for High Performance Computing Systems.* Pennsylvania, USA.

Osorio, F. (2008). *Base de datos relacionales: Teoría y práctica. (1ª ed.).* Madrid, España: Thomson.

Palacio, J. (2008). *Flexibilidad con Scrum.*

Pham, A., & Pham, P. (2010). *Scrum Rin Action: Agile Software Project Management and Development*.

Pressman, R. S. (2002). *Ingeniería de software. Un enfoque práctico. (2ª ed.)*. España: McGraw-Hill Interamericana.

Pressman, R. S. (2005). *Ingeniería de Software. Un enfoque práctico. (5ª ed)*. España: McGraw-Hill/Interamericana de España.

Pressman, R. S. (2010). *Ingeniería de Software un enfoque práctico*. México: Mc Graw Interamericana.

Pressman, R. S. (2011). *Software Engineering, a practitioner's approach España*. España: McGraw-Hill/Interamericana.

Ramos Orozco, D. (2012). *Control de versiones sobre bases de datos relacionales: Análisis de herramientas y sus características.* México.

Reyes, I. (2008). *Método de recolección de datos. Universidad de Carabobo.*

Sáez, P. (2013). *Identificación y valoración de técnicas ágiles de gestión de proyectos software*. Oviedo.

Santos, L., Galan, J., Izquierdo, L., & Del Olmo, R. (2009). *Aplicaciones de las TIC en el nuevo modelo de enseñanza del EEES.*

Schwaber, & Sutherland. (2011). *Marco de trabajo Scrum.* España: McGraw-Hill Interamericana.

Silberschatz, A., Korth, H., & Sudarshan, S. (2002). *Fundamentos de bases de datos. (4ª ed.)*. España: McGraw-Hill Interamericana.

Suárez, E. (2011). *Análisis comparativo de los Frameworks EJB3 y Entity framework para el desarrollo de aplicaciones empresariales.* Quito, Ecuador.

Szyperski, c. (2002). *Component software: Beyond object oriented programming. (2ª ed.)*. Boston, USA: Addison Wesley Longman Publishing.

Trejo, I. (2002). *Módulo XML para acceder al Sistema Administrador de Bases de Datos SQLmx a través de Internet. Tesis (Magíster en Ciencias de la Computación).* México D. F.

Tuya, G. J. (2007). *Técnicas cuantitativas para la gestión en la ingeniería el software.* Madrid.

Vanegas, C. (2013). *Funcionalidad del lenguaje integrado de consultas (linq).*

Velasco, M. (2010). *Especificación de una interfaz común SQL para la comunicación de base de datos heterogéneos en diferentes sistemas operativos.* Querétaro, México.

Zorrilla. (1993). *Investigación científica- Tipo de Investigación Aplicada.* México: Océano.

# ANEXO A

## MATRIZ DE OPERACIONALIZACIÓN DE VARIABLES

| VARIABLES | DIMENSIONES | INDICADORES | ÍTEMS | INSTRUCTIVO |
|---|---|---|---|---|
| COMPONENTE SOFTWARE | Capa Contexto | Patrón de diseño | ¿Cuáles son los patrones de diseño que permiten controlar la versión del esquema de base de datos relacional? | Análisis documental |
| | | | ¿Los patrones de diseño de las arquitecturas actuales posibilita solucionar las deficiencias en controlar la versión del esquema de base de datos relacional? | Análisis documental |
| | | Entity framework | ¿Cómo implementa entity framework el control de versiones del esquema de base de datos relacional? | Análisis documental |
| | | | ¿Entity framework mejora las deficiencias en control de versiones del esquema de base de datos relacional? | Análisis documental |

| | | | | |
|---|---|---|---|---|
| | | Orm (mapeo objeto-relación) | ¿Cómo utilizar el ORM (mapeo objeto-relación) para solucionar problemas en control de versiones del esquema de base de datos relacional? | Análisis documental |
| | Capa Modelo de Datos | Tipo de Dato | ¿Es posible mantener homogeneidad de tipo de dato en el cambio de versiones del esquema de base de datos relacional? | Análisis documental |
| | | | ¿En qué medida afecta la heterogeneidad de los tipos de datos en el cambio de versiones del esquema de base de datos relacional? | Análisis documental |
| | | Code first | ¿Cómo patrón code first de entity framework controla las versiones del esquema de base de datos relacional? | Análisis documental |
| | | Model first | ¿Es factible usar el patrón model first de entity framework para controlar las versiones del esquema de base de datos relacional? | Análisis documental |

| | | | | |
|---|---|---|---|---|
| | | Data base first | ¿Al utilizar el patrón data base first de entity framework es posible controlar las versiones del esquema de base de datos relacional? | Análisis documental |
| VERSIÓN DEL ESQUEMA DE BASE DE DATOS RELACIONAL | Identificación de los cambios | Análisis | ¿Cómo analizar el componente software para controlar las versiones del esquema de base de datos relacional?<br>¿Cómo analizar correctamente un componente software para controlar las versiones del esquema de base de datos relacional? | Ficha observacional |
| | | Diseño | ¿Cómo diseñar el componente software para controlar las versiones del esquema de base de datos relacional?<br>¿Es importante diseñar el componente software para controlar óptimamente las versiones del esquema de base de datos relacional? | Ficha observacional |

| | | | | |
|---|---|---|---|---|
| | | Codificación | ¿Cómo codificar los cambios del esquema de base de datos relacional para controlar las versiones del esquema de base de datos relacional? | Ficha observacional |
| | | Prueba | ¿De qué manera implementar las pruebas unitarias para controlar las versiones del esquema de base de datos relacional?<br>¿Las pruebas unitarias pueden controlar las versiones de esquema de base de datos relacional? | Ficha observacional |
| | Control de cambios | Versión | ¿Cómo versionar el esquema de base de datos relacional para controlar las versiones del esquema de base de datos relacional? | Ficha observacional |
| | | Revisión | ¿Cómo la revisión de los cambios del esquema de base de datos relacional controla las versiones del esquema de base de datos relacional? | Ficha observacional |

| | | | | |
|---|---|---|---|---|
| | | Reléase | ¿Cómo el reléase de una aplicación controla las versiones del esquema de base de datos relacional? | Ficha observacional |
| | Auditoria de los cambios | Verificar el cambio | ¿Cómo la verificación del cambio ayuda a controlar las versiones del esquema de base de datos relacional? | Ficha observacional |
| | | Validar el esquema | ¿Cómo validar el esquema de base de datos relacional para controlar óptimamente las versiones del esquema de base de datos relacional?<br>¿? | Ficha observacional |

Tabla N° A.1: Matriz de operacionalización de las variables.

# ANEXO B

## FICHA ANÁLISIS DOCUMENTAL

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FECHA | | | | | | | | | | | | | | | | | | | |
| TIEMPO | | | | | | | | | | | | | | | | | | | |
| HORAS | | | | | | | | | | | | | | | | | | | |
| APELLIDOS Y NOMBRES: | | | | | | | | | | | | | | | | | | | |

Tabla N° B.1: Formato de análisis documental del componente software de control de versiones del esquema de base de datos relacional.

# ANEXO C

## FICHA OBSERVACIONAL

1. ¿Cuáles son los patrones de diseño que permiten controlar la versión del esquema de base de datos relacional?

   .....

   .....

2. ¿Los patrones de diseño de las arquitecturas actuales posibilita solucionar las deficiencias en controlar la versión del esquema de base de datos relacional?

   .....

   .....

3. ¿Cómo implementa entity framework 6 el control de versiones del esquema de base de datos relacional?

   .....

   .....

4. ¿Entity framework 6 mejora las deficiencias en control de versiones del esquema de base de datos relacional?

   .....

   .....

5. ¿Cómo utilizar el ORM (mapeo objeto-relación) para solucionar problemas en control de versiones del esquema de base de datos relacional?

   .....

   .....

6. ¿Es posible mantener homogeneidad de tipo de dato en el cambio de versiones del esquema de base de datos relacional?

   .....

   .....

7. ¿En qué medida afecta la heterogeneidad de los tipos de datos en el cambio de versiones del esquema de base de datos relacional?

   .....

| |
|---|
| ……<br>8. ¿Cómo patrón code first de entity framework controla las versiones del esquema de base de datos relacional?<br>……<br>……<br>9. ¿Es factible usar el patrón model first de entity framework para controlar las versiones del esquema de base de datos relacional?<br>……<br>……<br>10. ¿Al utilizar el patrón data base first de entity framework es posible controlar las versiones del esquema de base de datos relacional?<br>……<br>…… |

Tabla N° C.1: Ficha Observacional para recolectar información del componente software de control de versiones del esquema de base de datos relacional.