Desarrollo web para la gestión del proceso de loterías de la empresa Dvalor S.A.S. de la ciudad de Cali

Erwin Oquendo Villada, ✉ erwin51522@hotmail.com
Edgar Fabián Velásquez Ahumada, ✉ fabian_vel@hotmail.com

Trabajo de Grado presentado para optar al título de Ingeniero en Sistemas

Asesor: Nilson Mossos Vivas, Ingeniero (Ing.) en Sistemas

Institución Universitaria Antonio José Camacho
Facultad de Ingenierías
Ingeniería en Sistemas
Cali - Colombia
2018

Nota de aceptación:

Aprobado por el Comité de Grado en cumplimiento de los requisitos exigidos por la Institución Universitaria Antonio José Camacho para optar al título de Ingeniero en Sistemas

Jurado

Jurado

Santiago de Cali, 13 de agosto de 2018

Agradecimientos

Inicialmente queremos agradecer a nuestra familia por todo el apoyo que nos brindaron durante todo el trascurso del proyecto de grado al igual que durante toda nuestra etapa de crecimiento profesional, además de inculcarnos esos valores que nos ha permitido mantenernos firmes con nuestros sueños y metas.

A nuestras esposas por todo el entendimiento, paciencia y apoyo que tuvieron con nosotros todo este tiempo.

Así mismo a nuestros amigos, compañeros y asesores, los cuales, gracias a su conocimiento, consejos y experiencia nos han permitido llegar hasta este punto de nuestra vida académica y profesional.

Finalmente queremos agradecerle a Dios por darnos salud y vitalidad para llegar hasta donde estamos y poder cumplir esas metas que tenemos en mente.

Tabla de contenido

Lista de tablas

Lista de figuras

Resumen

El presente proyecto propone una solución tecnológica para la configuración, procesamiento, generación de insumos para producción y el control de los estados que se necesitan para el procesamiento de los diferentes productos que se producen en la compañía Cadena (Dvalor S.A.S.) en la UEN de protección contra el fraude, tales como:

- Liberación procesamiento (cargue de archivos de mezcla y distribuidores).
- Liberación diseño (notificación de cambios en la variable del diseño).
- Liberación diseño layout (nombre de nuevo diseño, si el producto así lo requiere).
- Liberación producción (generación de archivos de impresión).

Garantizando que cada área que esté involucrada en el proceso se haga responsable de la configuración del producto que le corresponda, de esta manera se mitigan que se generen problemas de calidad o reprocesos en la línea de producción, también se pretende con el proyecto de que al momento del ingreso de un nuevo producto o que alguno de los existentes tengan cambio no sea tan traumático para el área de TI ya que se está planteando una aplicación configurable y extensible para que sea lo más configurable el ingreso de una nueva solicitud.

Palabras clave: procesamiento, control, loterías, producción.

Abstract

The present project proposes a technical solution for the configuration, the processing, the generation of inputs for the production and the control of the states that are shown for the processing of the different products that are produced in the company Cadena (Dvalor S.A.S.) in the UEN protection against fraud, such as:

- Processing release.
- Design release.
- Release design layout (name of new design, if the product requires it).
- Production release.

Ensuring that each area that is involved in the process is responsible for the configuration of the product that corresponds to it, in this way they mitigate that quality problems or reprocesses are generated in the production line, it is also intended with the project that The moment of the entry of a new product or that any of the existing ones have change is not so traumatic for the TI area since a configurable and extensible application is being proposed so that the entry of a new application is the most configurable.

Keywords: processing, control, lloteries, production.

Introducción

En el presente proyecto se desarrolla una solución tecnológica para la configuración, el control y procesamiento de los diferentes productos de valor que se producen en el planta de valores de la UEN (Unidad Estratégica de Negocio) de protección contra el fraude de la compañía Cadena (Dvalor S.A.S.), Actualmente la empresa cuenta con un sistema llamado SISLOT (Sistema De Información Para Loterías) para el procesamiento de la información de los productos, pero se observó que dicha aplicación tiene muchos inconvenientes al momento de hacerla extensiva para nuevos productos puesto que por cada producto se tiene que modificar el código fuente, también se observa que realizar un cambio a una característica de un producto es muy complicado puesto que este cambio puede afectar a otros productos debido a la forma en que está construida el aplicativo.

Por otra parte se observó que toda la responsabilidad de la configuración y procesamiento del producto esta delegada a una sola área que es la de TI producción Cali, quienes en estos momento son los responsables de recolectar toda la información necesaria para la producción de los productos con las diferentes áreas involucradas, ocasionando en muchos casos inconvenientes de calidad puesto que la información recogida por el personal de TI en muchas ocasiones tienen cambios que afecta la configuración del producto y estas no son notificadas al área de TI provocando en muchas situaciones problemas de calidad o reproceso en la impresión de dicho producto.

Lo que se pretende con el proyecto es brindar una herramienta donde cada área sea responsable de configurar el producto de una manera fácil y ordenada manejando estados por cada acción realizada por los usuarios y controlando que toda la información que se requiere para la producción de los productos se encuentre actualizada antes de liberar el producto a producción. En dicha liberación a producción se generan los insumos necesarios para producción, tales como, archivos de impresión, papelería, reportes, entre otros insumos requeridos en producción. Cada rol de usuario podrá realizar las acciones específicas de cada rol, las cuales interactúan con los estados presentes en el flujo de procesos del área de juegos, entre los cuales se encuentra el procesamiento de los productos, liberación por parte de diseño, liberación por parte de TI y liberación por parte de planeación.

1 Planteamiento del problema

El principal problema detectado en la línea de juegos de la empresa Dvalor S.A.S. es la falta de un flujo claramente definido entre los diferentes procesos por donde se elabora el producto. Cada área involucrada trabaja aislada de las demás lo que ocasiona reprocesos e incluso daños en la impresión del producto, como consecuencia a esta falta de comunicación, el área de tecnología tiene que soportar y corregir la falla ocasionada, generando además pérdidas materiales e incluso un posible incumplimiento en la entrega del producto al cliente, o peor un reclamo por parte del cliente.

Otra problemática que enfrenta principalmente el área de tecnología, es que existen diferentes aplicativos que realizan una tarea específica para el mismo producto, lo cual implica que si se requiere realizar alguna modificación en una de sus características, toca realizar dicha modificación en todos los diferentes aplicativos existentes, generando demora e inconsistencias durante el desarrollo y afectando los tiempos en producción.

El área de tecnología constantemente tiene dificultades con el soporte y modificación de los aplicativos, ya que el código fuente de las aplicaciones no cuentan con buenas prácticas de programación por lo tanto cuando se requiere realizar un nuevo cambio o adicionar una nueva característica esta tarea resulta muy tediosa, toma más tiempo del estimado, y se corre el riesgo de no realizar un cambio no contemplado ocasionando reproceso y en el peor de los casos algún daño o reclamo en el producto.

La información no está centralizada, por lo tanto, cuando se necesita realizar alguna trazabilidad, dicha tarea toma mucho tiempo y en ocasiones los datos encontrados no son consistentes.

Las aplicaciones no son escalables, lo que dificulta la labor de implementación de nuevos requerimientos.

Eso sin contar la redundancia de datos tanto almacenados en bases de datos como implementados directamente en el código fuente de los aplicativos.

Por otra parte, en el procesamiento de la información, esta tarea solamente puede ser realizada por el área de tecnología, debido a su complejidad, lo que no permite que varios de los analistas del área puedan emplear su tiempo en otros desarrollos, proceso que, si se implementa de

manera correcta, lo podría realizar cualquier área a la cual se le asigne dicha labor sin tener que recurrir al área de TI.

1.1 Antecedentes

Se han consultado en diferentes tipos de fuentes, información y documentos cuya problemática, objetivos o resultados tengan relación con el presente proyecto y que puedan contribuir a la creación del sistema propuesto.

A continuación, se da un breve resumen de los proyectos que fueron consultados, que debido a la naturaleza de los productos a los cuales tienen relación el proyecto, es un producto específico que no se encuentra a la disposición libre al público, por lo tanto, se centró en los antecedentes internos de la compañía Dvalor S.A.S., la cual ya tiene una trayectoria en este campo (Dvalor S.A.S., 2018).

El primer antecedente que se encontró es el que lleva como título SISTEMA DE INFORMACIÓN PARA LOTERÍAS (SISLOT), el cual fue desarrollado por el ingeniero Andrés Ayala. Dicho proyecto correspondía en mejorar el proceso y tiempo de procesamiento de las loterías, que en algunos casos podría tomar hasta 5 días para procesar un solo sorteo, y al haber algún error, se debía ejecutar el proceso de nuevo. Con este sistema, se pasó de demorar horas o hasta 5 días el procesamiento, a 5 a 10 minutos por procesamiento.

El segundo antecedente es el que lleva por nombre SISTEMA DE INFORMACIÓN PARA FORMATEO DE ARCHIVOS LOTERÍAS (SIFAL). Elaborado por Erwin Oquendo, y actualizado por los ingenieros Fabián Velásquez, Darío Agudelo y Duvan Velásquez, la problemática que se planteaba era que los insumos o archivos que enviaban las diferentes loterías pudieran cargarse por el aplicativo, darle formato dependiendo de las estructuras de estos archivos y lógica especial a aplicar, y generar archivos de salida los cuales tuvieran una estructura predefinida que luego debían cargarse por la aplicación SISLOT para realizar el procesamiento de las loterías.

El siguiente antecedente trata de un SISTEMA DE EMPAQUE DE LOTERÍAS, elaborado por el ingeniero Julián Borrero, la problemática que se plantea es que la empresa Dvalor S.A.S. debía garantizar que el armado y empaque de los paquetes y cajas de las loterías se realizaran de la manera correcta, y permitir un control o detección oportuna frente a errores humanos realizados

por los operarios de la planta de juegos, con el fin de evitar a su vez reclamos o incidentes reportados al interior de la compañía o por los clientes.

El siguiente lleva como nombre UTILITARIOS LOTERÍAS, elaborado por Cristian Martínez, el cual realizaba procesos de cargue y formateo de archivos planos con el fin de generar archivos de salida con formatos específicos. Este aplicativo es útil para generar los archivos de respuesta de las loterías.

El antecedente a continuación es MAPPER, un software creado por Jorge Arias, este software brinda ayuda para el formateo de archivos de cualquier fin, especialmente para archivos de impresión en donde permitiría adicionar campos con lógicas particulares que eran necesarias en las máquinas de impresión de la empresa Dvalor S.A.S. o campos que eran necesarios para el control de impresión.

Por último, en la empresa Dvalor S.A.S. se había comenzado con un proyecto llamado APOLO, con el ingeniero Darío Agudelo, pero debido a su salida de la empresa y a un enfoque errado que le estaban dando al proyecto, este se descartó. El proyecto pretendía darle solución a muchos de los problemas que hoy existe a nivel de sistemas con el procesamiento de las loterías, es por eso que este proyecto nos es muy importante ya que tendremos la oportunidad de conocer puntos fuertes que podamos reutilizar y los puntos en los cuales no hicieron que este proyecto tuviera éxito, para replantear su idea inicial y realizar algo que satisfaga la necesidad real.

2 Justificación

Las aplicaciones web son usadas para gestionar información de diferentes modelos de negocio, por lo tanto, a través de este proyecto se buscó que la empresa Dvalor S.A.S. pudiera tener un mejor control de las producciones que ingresan a la planta de juegos, brindar una herramienta con la que puedan administrar y monitorear los productos y así conocer el estado de cada una de las ordenes que estén pendientes, en ejecución e incluso las finalizadas.

Con la implementación de esta aplicación web le permite a la empresa Dvalor S.A.S tener trazabilidad del proceso que realizan los productos de loterías, los datos del producto, los insumos requeridos, las áreas que interactúan con este, para de esta forma minimizar los riesgos a los cuales están expuestos los productos durante su proceso de producción.

También brindar una herramienta que ayude al área de Tecnología de la empresa, entregar soluciones en menor tiempo, con menor índice de riesgo en las entregas, mejorando los tiempos de respuestas frente a nuevas órdenes o productos que ingresen a producción.

Con base a los anteriores argumentos, el desarrollo propuesto pretende normalizar un proceso que hasta el día de hoy no estaba estandarizado, además le proporciona a la empresa Dvalor S.A.S., seguridad, un mayor control y responsabilidad en cuanto a los procesos e insumos que interactúan con los productos. Con la ayuda de este desarrollo, las personas interesadas podrán conocer el estado de los productos, podrán conocer si alguna área no ha entregado algún insumo necesario para el producto, si existe algún proceso pendiente, o si el producto ya está listo para entrar a producción, permitiendo tomar decisiones oportunas al interior de la empresa e incluso escalarlas a los clientes en caso de ser necesario. Adicional, podrán tener los datos centralizados y el proceso de actualización será menos tedioso.

3 Objetivos

3.1 Objetivo general

Desarrollar una aplicación web “responsive” en donde se pueda realizar la configuración, procesamiento, generación de insumos para producción, papelería, reportes, control de estados, del flujo de procesos de los productos del área de juegos de la empresa Dvalor S.A.S.

3.2 Objetivos específicos

- Explorar el flujo más adecuado que deba tener el proceso en producción de los productos de la línea de juegos.
- Implementar dentro del flujo los siguientes procesos: su creación, su configuración y procesamiento, de una manera configurable, confiable, escalable y de fácil manejo para los empleados de la compañía.
- Explorar la arquitectura adecuada que permita que la aplicación a desarrollar sea escalable, confiable y configurable.
- Facilitar por medio de parametrización la creación y/o modificación de productos asociados a la línea de juegos.
- Implementar un prototipo funcional que permita llevar a cabo el flujo del proceso de los productos del área de juegos.

4 Problema de investigación

4.1 Pregunta de formulación

¿Cómo estandarizar el flujo de procesos y centralizar las aplicaciones existentes involucradas en la producción de loterías de una forma confiable y configurable para todo el producto?

4.2 Preguntas de sistematización

- ¿Cómo unificar la información de las diferentes bases de datos?
- ¿Cómo integrar en una misma aplicación el proceso que realiza los diferentes aplicativos entre las diferentes áreas?
- ¿Cómo mejorar el flujo del proceso que realiza la aplicación a desarrollar de tal forma que pueda ser utilizada por cualquier usuario de la empresa?

5 Marco teórico

5.1 UML

UML (Unified Modeling Language) es un lenguaje de propósito general que ayuda a especificar, visualizar y documentar modelos de sistemas de software, incluida su estructura y diseño, de tal forma que se unifiquen todos sus requerimientos (Definición oficinal de UML) (Expósito, Expósito, López, Melián, & Moreno).

5.1.1 Elementos de UML

UML hace uso de una serie de elementos que ayudan a su estructura y modelado, los cuales se dividen dependiendo de su funcionamiento y enfoque.

5.1.1.1 Elementos estructurales

- Partes estáticas del modelo.
- Representan elementos conceptuales, físicos o materiales.
- Clases, objetos, interfaces, casos de uso, actor, etc.

5.1.1.2 Elementos de comportamiento

- Describen el funcionamiento de un sistema.
- Interacciones, máquinas de estado.

5.1.1.3 Elementos de agrupación

- Permiten agrupar otros elementos del modelo.
- Componentes, paquetes, nodos.

5.1.1.4 Elementos de anotación

- Sirven para realizar anotaciones extra.
- Notas.

5.1.2 Diagramas UML

Si bien, UML se compone de una serie de elementos, estos a su vez conforman artefactos que son elaborados a través de diagramas, cada uno con su función específica que permiten describir el comportamiento del software o sus componentes, ya sea lógica o física.

5.1.2.1 Diagramas Estructurales

- Diagrama de Casos de Uso
- Diagrama de Clases
- Diagrama de Objetos

5.1.2.2 Diagramas de Comportamiento

- Diagrama de Estados
- Diagrama de Actividad

5.1.2.3 Diagramas de Interacción

- Diagrama de Secuencia
- Diagrama de Colaboración

5.1.2.4 Diagramas de Implementación

- Diagrama de Componentes

- Diagrama de Despliegue/Distribución

Varios de estos elementos y artefactos se emplearon durante el desarrollo del proyecto ya que estos son ampliamente utilizados en la metodología tradicional RUP la cual fue aplicada para este proyecto, sin embargo, no todos los elementos fueron incluidos ya que se seleccionaron solo los que aportarán valor junto a la convivencia con la metodología ágil Scrum. En los capítulos posteriores se ampliará información al respecto.

5.2 Metodologías tradicionales

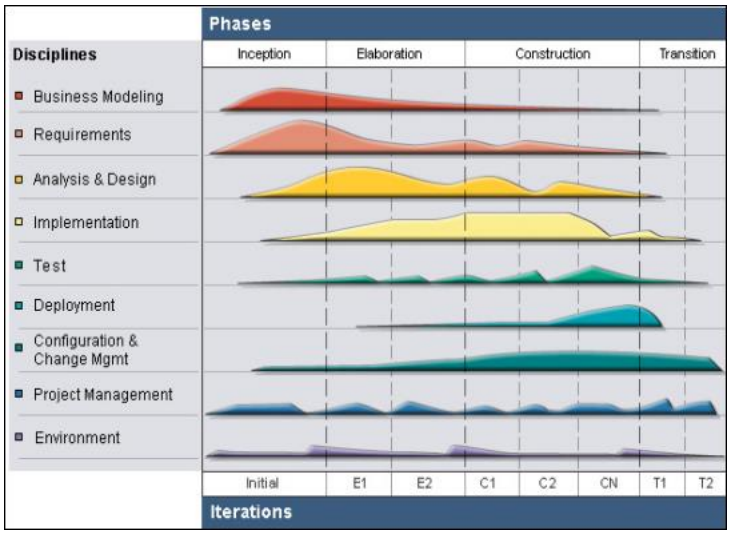
También llamadas metodologías pesadas. Son aquellas propuestas metodológicas tradicionales que se centran en mayor proporción al control del proceso, a las actividades y artefactos que se deben producir (Penadés & Letelier Torres, 2006). Inicialmente fueron muy utilizadas para el desarrollo de productos de software, ya que brindaban herramientas y procesos muy completos que permitían entregar mayor control durante el ciclo de vida del proyecto, sin embargo, en muchas ocasiones dependiendo del tipo de proyecto, se iba complicando tanto para el equipo de desarrollo como para el cliente ya que, debido a tantos procesos y documentación, el proyecto se podría tornar complejo, y no muy amigable a los cambios. Hoy en día se siguen utilizando, pero en menor proporción debido a la implementación de otra propuesta metodológica (metodología ágil) la cual explicaremos más adelante.

5.2.1 RUP

La metodología RUP, abreviatura de Rational Unified Process o en español Proceso Unificado Racional, es un marco de proceso de ingeniería de software. Proporciona mejores prácticas orientadas al desarrollo de software exitoso y una disciplina de enfoque para asignar tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es garantizar la producción de software de alta calidad que satisfaga las necesidades de sus usuarios con un tiempo estimado y presupuesto predecibles (Péraire, Edwards, Fernandes, Mancin, & Carroll, 2007).

A continuación, se podrá apreciar una figura con la arquitectura general de RUP.

Figura 1. Arquitectura RUP



Nota: Fuente http://www.redbooks.ibm.com/redbooks/pdfs/sg247362.pdf (The IBM Rational Unified Process for System z, 2007).

Como muestra la Figura anterior, RUP tiene dos dimensiones:

- El eje horizontal representa el tiempo y muestra los aspectos del ciclo de vida del proceso, ya que este se divide en cuatro fases: Inicio, Elaboración, Construcción y Transición. Cada fase se divide en una o más iteraciones.
- El eje vertical representa disciplinas, como requisitos, análisis y diseño, o implementación, que lógicamente agrupa actividades por naturaleza

5.2.1.1 Filosofías y principios

Estas filosofías y principios son la base sobre la cual se ha desarrollado RUP. A continuación, se mostrarán los principios clave para el desarrollo de software exitoso, los cuales caracterizan las mejores prácticas de la industria en creación, despliegue y evolución de sistemas de información (Péraire, Edwards, Fernandes, Mancin, & Carroll, 2007):

- Adapta al proceso.
- Equilibra las prioridades de las partes interesadas que compiten.
- Colaba en todos los equipos.
- Demuestra valor iterativamente.

- Eleva el nivel de abstracción.
- Céntrate continuamente en la calidad.

5.2.1.2 Ciclo de vida

5.2.1.2.1 Dirigido por casos de uso

En los casos de uso se plasman los requerimientos que los usuarios necesitan y desean que estén en el software que se va a realizar, permiten dimensionar el alcance de la solución, además sirve de guía para el proceso de desarrollo ya que los modelos que se obtienen, como resultados de los diferentes flujos de trabajo, representan la realización de los casos de uso (E.V.A. UCI, s.f.).

5.2.1.2.2 Centrado en la arquitectura

La arquitectura de un sistema de software surge a raíz de las necesidades de la organización, de cómo la perciben los usuarios y de cómo están plasmadas en los casos de uso, necesario para comprenderlos. Se deben tener además en cuenta varios factores los cuales influyen en la arquitectura, como la plataforma en la que funcionará el software, consideraciones de implementación, sistemas heredados y requisitos no funcionales, todo esto se refleja a través de vistas del sistema en proceso de desarrollo (Gil, 2008).

5.2.1.2.3 Iterativo e incremental

Una iteración involucra actividades de todos los flujos de trabajo, aunque se centra en algunos más que otros.

Se recomienda que cada iteración no contenga gran cantidad de actividades, por el contrario, la idea es dividir el trabajo en partes más pequeñas o miniproyectos, en donde cada una de ellas es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en el flujo de trabajo, y los incrementos, al crecimiento del producto (Gil, 2008).

Figura 2. Proceso de desarrollo Iterativo e Incremental de RUP

Nota: Fuente http://www.usmp.edu.pe/publicaciones/boletin/fia/info49/articulos/RUP%20vs.%20XP.pdf (RUP vs. XP, 2015)

5.2.1.3 Fases

Cada fase representa un ciclo de desarrollo en la vida de un producto de software.

5.2.1.3.1 Fase Inicio

Aquí se definen los elementos principales para poder iniciar con cualquier proyecto de software, tales como la visión, los objetivos y el alcance del proyecto, tanto a nivel funcional como técnico. En esta fase se da como resultado principal la lista de casos de uso y la lista de factores de riesgo del proyecto (E.V.A. UCI, s.f.).

5.2.1.3.2 Fase Elaboración

El objetivo de la fase de Elaboración es definir la arquitectura del sistema para proporcionar una base estable para la mayor parte del esfuerzo de diseño e implementación en la Fase de Construcción. La arquitectura evoluciona teniendo en cuenta los requisitos más importantes (aquellos que tienen un gran impacto en la arquitectura del sistema) y una evaluación de los riesgos.

Por medio de uno o más prototipos arquitectónicos se evalúa la estabilidad de la arquitectura (Péraire, Edwards, Fernandes, Mancin, & Carroll, 2007).

5.2.1.3.3 Fase Construcción

El objetivo de esta fase es aclarar los requisitos restantes y completar el desarrollo del sistema en función de la arquitectura base definida. Esta fase es en cierto sentido un proceso de fabricación, donde se hace hincapié en la gestión de los recursos y el control de las operaciones para optimizar los costes, los horarios y la calidad. Permite además poder contar con entregables tempranos de versiones del sistema que satisfacen los casos de uso de mayor valor (Péraire, Edwards, Fernandes, Mancin, & Carroll, 2007).

5.2.1.3.4 Fase Transición

En esta fase se entrega una versión de prueba para ser utilizado por el usuario, para de esta forma probar el producto antes de su lanzamiento a producción, y así realizar ajustes menores en función de los comentarios de los usuarios, los cuales deben centrarse principalmente en ajustar el producto, configurarlo y corrigiendo problemas de usabilidad. Después de esta iteración se espera contar con una versión final para ser puesto en producción (Péraire, Edwards, Fernandes, Mancin, & Carroll, 2007).

5.3 Metodologías ágiles

El modelo iterativo e incremental el cual ha sido utilizado durante hace muchos años para el diseño y desarrollo de proyectos de ingeniería fue fundamental para la creación de las metodologías ágiles. La metodología propulsora fue eXtreme Programming (XP), la cual fue la base de este tipo de metodologías en la industria del software. Luego de una reunión realizada en febrero del 2001, nace el termino ágil aplicado al desarrollo de software, en donde estuvieron participes un grupo de expertos, incluyendo algunos de los creadores e impulsores de las metodologías de software. El objetivo de dicha reunión fue diseñar los valores y principios que deberían permitir a los equipos desarrollar productos de software de manera rápida y con calidad,

con la capacidad de adaptarse y responder de manera ágil y segura, ante las necesidades de cambios que puedan surgir por parte del cliente durante el desarrollo de los proyectos. Esto con la necesidad de proponer alternativas ante las ya conocidas metodologías tradicionales las cuales eran muy rigorosas con una documentación extensa lo que en ocasiones volvía complejo llevar un proyecto y esto se reflejaba aún más cuando el proyecto era cambiante y tenía novedades en sus requerimientos.

Tras dicha reunión, se creó The Agile Alliance, una organización sin ánimo de lucro dedicada a diseñar y promover conceptos relacionados con el desarrollo ágil de software y apoyar a las organizaciones para que comiencen a implementar dichos conceptos. El punto de partida fue el Manifiesto Ágil, un documento que resume la filosofía ágil (Bioul, Escobar, Alvarez, Nardin, & Ricci, 2010).

5.3.1 Manifiesto Ágil

Con el Manifiesto Ágil (Agile Manifesto) los impulsadores y creadores del agilismo en el software, establecieron una serie de valores y principios generales que deben ser comunes para todas las metodologías agiles. Este menciona cuatro valores base para proporcionar equipos de alto rendimiento. A continuación, se describirá cada uno de ellos.

- **Individuos e interacciones** sobre procesos y herramientas.

  Los individuos y las interacciones son fundamentales para crear equipos de trabajo de alto rendimiento. Estudios han comprobado que cuando no existen problemas de comunicación dentro del equipo, estos pueden tener un rendimiento 50 veces mejor que el promedio en la industria. Para lograr esta mejora y facilitar la comunicación, las metodologías ágiles se basan en ciclos frecuentes de inspección y adaptación. Estos ciclos pueden variar en su tiempo de duración, pueden ser reuniones diarias con una duración de minutos, o algunas horas con integraciones continuas, en cada iteración se debe realizar su correspondiente revisión y retrospectiva (Sutherland, 2013).
- **Software funcionando** sobre documentación extensiva.

Este es uno de los puntos diferenciadores que aporta el desarrollo ágil. Estas hacen hincapié en realizar entregas pequeñas del software al cliente que generen valor, esto se hace en intervalos establecidos. Estas funcionalidades y entregables deben estar completas y se debe garantizar que superen todas las pruebas y pueda ser utilizada por el usuario final (Sutherland, 2013).

- **Colaboración con el cliente** sobre negociación contractual.

  Se crea un perfil cuyo rol será el encargado de ser el intermediario entre el equipo de trabajo y los clientes, este perfil tiene el nombre de Product Owner, o propietario del producto. El Product Owner es el representante no solo de los clientes o usuarios interesados en el proyecto, sino también de áreas o dependencias de las empresas de los clientes, como por ejemplo ventas, soporte, administración, servicio al cliente, entre otras. Es el responsable de actualizar la lista de requisitos cada que el equipo de trabajo lanza un entregable funcional, teniendo en cuenta la situación real, los comentarios de los clientes y las partes interesadas. La idea principal, es estar en total comunicación con el cliente (Sutherland, 2013).

- **Respuesta ante el cambio** sobre seguir un plan.

  Para que los equipos de trabajo creen productos que agraden a los clientes y les brinde valor comercial, deben responder al cambio. Esto se debe principalmente, a que, en la mayoría de los casos, el cliente no sabe lo que está pidiendo hasta que ve el software funcionando, ya que lo que encuentran no es exactamente lo que querían. Las metodologías ágiles buscan retroalimentación de los clientes a lo largo del proyecto para que los equipos puedan incorporar información nueva a medida que se desarrolla el producto.

  Es por eso que han establecido procesos, como revisiones y retrospectivas, que están diseñados específicamente para cambiar las prioridades regularmente en función de los comentarios de los clientes y su valor comercial (Sutherland, 2013).

"Esto es, aunque valoramos los elementos de la derecha, valoramos más los de la izquierda" (Beck, y otros, Manifiesto por el Desarrollo Ágil de Software, 2001).

5.3.2 Principios del Manifiesto Ágil

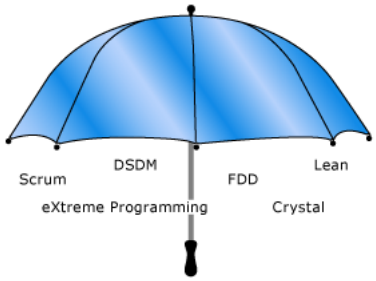
Los siguientes son los principios tomados del Manifiesto oficial (Beck, y otros, Principios del Manifiesto Ágil, 2001).

- Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
- Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
- Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
- Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
- Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
- El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
- El software funcionando es la medida principal de progreso.
- Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
- La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
- La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
- Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
- A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

5.3.3 Implementaciones

El desarrollo ágil no es una metodología en sí misma, es un término general que describe varias metodologías ágiles. En la firma del Manifiesto Ágil en 2001, estas metodologías incluyeron Scrum, XP, Crystal, FDD y DSDM (Sutherland, 2013).

Figura 3. Marco incluyente de desarrollo ágil



Nota: Fuente https://docs.microsoft.com/es-es/previous-versions/visualstudio/visual-studio-2012/dd997578(v=vs.110) (Principios y valores de Agile, por Jeff Sutherland, 2013)

Cada metodología ágil tiene un enfoque ligeramente diferente para implementar los valores centrales del Manifiesto Ágil. Según las encuestas muestran que alrededor del 50 por ciento utilizan Scrum, otro 20 por cierto dice que está utilizando Scrum con componentes de XP, un 12 por ciento adicional dice que solo están haciendo XP (Sutherland, 2013).

5.3.4 Scrum

Scrum es un marco de trabajo utilizado para desarrollar y gestionar productos de software, tanto livianos como complejos. Con este marco de trabajo se pueden emplear diferentes procesos y técnicas en donde el equipo de trabajo puede abordar problemas complejos adaptativos, y a su vez entregar productos con un alto grado de valor productivo. En esta se aplican un conjunto de buenas prácticas para trabajar colaborativamente en equipo y obtener el mejor resultado posible en un proyecto.

En Scrum se realizan entregas constantes, parciales, y priorizadas dependiendo del valor que les generen a los clientes. Es por eso que Scrum es ampliamente utilizado hoy en día por tener la capacidad de trabajar en entornos complejos, con la posibilidad de obtener resultados tempranos,

donde los requisitos pueden ser cambiantes o poco definidos, y siempre con un enfoque de servicio, innovación, competitividad, flexibilidad y productividad (proyectosagiles.org, s.f.).

5.3.4.1 Pilares de la implementación

La metodología Scrum es soportada bajo tres pilares fundamentales que garantizan que su ejecución e implementación se realice como debe ser (Dirección general, 2018), estos son:

- Transparencia

  Los puntos relevantes y significativos del proceso deben ser accesibles y visibles para todos aquellos que están implicados en su desarrollo y ejecución, por lo tanto, se requiere contar con unos estándares y protocolos comunes que faciliten el entendimiento entre los participantes y compartan un concepto común de lo que están viendo.
- Inspección

  Una de las claves para generar esa flexibilidad y adaptabilidad a los cambios presente en la metodología Scrum, es la realización de revisiones constantes a los artefactos y progresos con el fin de detectar de manera temprana alteraciones no esperadas en el proceso.
- Adaptación

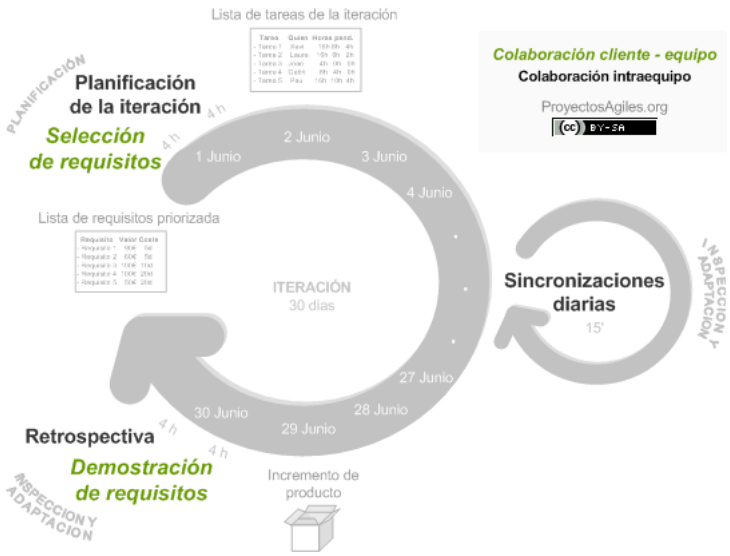
  Si algún responsable detecta algún aspecto en el proceso que genera alguna desviación inesperada frente a los términos aceptables, el proceso o el material se debe ajustar cuanto antes para minimizar desviaciones mayores.

5.3.4.2 ¿Cómo funciona Scrum?

En el proceso general en Scrum un proyecto se ejecuta en ciclos temporales cortos y con una duración cuyas iteraciones normalmente son de 2 semanas, aunque dependiendo de cómo lo defina el equipo pueden durar 3 semanas y máximo 4 semanas. En cada iteración se debe contar como resultado un entregable que corresponde a un incremento del producto final, este entregable

debe estar disponible ante el cliente si así lo solicita con el mínimo esfuerzo (proyectosagiles.org, s.f.).

Figura 4. Proceso en Scrum



Nota: Fuente https://proyectosagiles.org/como-funciona-scrum/ (proyectosagiles.org)

El insumo principal es la lista de requisitos, la cual es priorizada por el cliente con base al valor que le aportan respecto al costo y estos objetivos quedan repartidos en iteraciones y entregas (proyectosagiles.org, s.f.).

5.3.4.3 Equipo Scrum

El equipo Scrum está conformado por el dueño del producto (Product Owner), el equipo de desarrollo (Development Team) y el Scrum Master.

Los miembros del equipo deben estar capacitados para ser auto-organizados y auto-gestionables, con las competencias necesarias para realizar el trabajo sin depender de otros. También deben estar preparados para desempeñar cualquier función, cuyo modelo de equipo se enfoca en contar con un equipo de trabajo diseñado para optimizar la flexibilidad, creatividad y la productividad (Schwaber & Sutherland, 2013).

A continuación, se explicarán las funcionales principales de cada rol del equipo:

- Product Owner

  El Product Owner o dueño del producto es el responsable de maximizar el valor del producto y del trabajo del equipo de desarrollo. Es alguien del cliente con poder de decisión que asegura que el equipo conozca la perspectiva del negocio. Gestiona y define las historias de usuario, las prioriza, mantiene el Product Backlog, gestiona todos los interesados del proyecto, debe expresar claramente las necesidades, asegura que el Development Team entiende los elementos del Product Backlog (Schwaber & Sutherland, 2013).

- Development Team

  Es el grupo de personas que de manera conjunta desarrollan el producto del proyecto. Tienen un objetivo en común, comparten la responsabilidad del trabajo que realizan (así como su calidad) en cada Sprint y en el proyecto. El tamaño del equipo está entre 3 y 9 personas (Schwaber & Sutherland, 2013) (Dirección general, 2018).

  En general el equipo de desarrollo debe contar con las siguientes características:

  - Auto-organizados: el mismo equipo de trabajo define como convertir el Product Backlog en incrementos potencialmente liberables.
  - Multi-funcionales: todo el equipo de desarrollo tiene las habilidades y las capacidades necesarias para crear un incremento del producto final.
  - No se reconocen títulos para los miembros del equipo, independientemente del trabajo que realice la persona.
  - No se reconocen sub-equipos dentro del equipo de desarrollo, independientemente de la labor que ejecute la persona dentro del equipo, es decir que todos los miembros deben trabajar conjuntamente.
  - Los miembros del equipo pueden tener habilidades especializadas o áreas de enfoque, sin embargo, la responsabilidad pertenece al equipo de desarrollo.

- Scrum Master

  El Scrum Master es el Coach del equipo y es quien lo ayuda a alcanzar su máximo nivel de productividad posible. Tomando algunas referencias de Leonardo Wolk podemos decir que el Scrum Master, en tanto que coach, es el líder, facilitador, provocador, detective y soplador de brasas (Alaimo, 2013).

  Las responsabilidades principales del Scrum Master son:

  - Velar por el correcto empleo y evolución de Scrum.
  - Facilitar el uso de Scrum a medida que avanza el tiempo. Esto incluye la responsabilidad de que todos asistan a tiempo a las daily meetings, reviews y retrospectivas.
  - Asegurar que el equipo de desarrollo sea multifuncional y eficiente.
  - Proteger al equipo de desarrollo de distracciones y trabas externas al proyecto.
  - Detectar, monitorear y facilitar la remoción de los impedimentos que puedan surgir con respecto al proyecto y a la metodología.
  - Asegurar la cooperación y comunicación dentro del equipo.
- Stakeholders

  Si bien este tipo de usuario no pertenece como tal al Scrum Team, si está presente indirectamente, ya que corresponden a las partes interesadas que son responsables de comunicar sus necesidades, y proporcionar información sobre el producto al Product Owner. En resumen, es cualquier persona que tenga interés o participación en el proyecto. Esto puede ser los gestores directos de los miembros del equipo, las personas que proporcionan financiación para el proyecto, el director de proyecto, entre otros.

5.3.4.4 Actividades o ceremonias

5.3.4.4.1 Sprint Planning

Ceremonia donde el equipo Scrum realiza la planeación del Sprint a ejecutar (también llamada Planning), es decir, adquiere un compromiso de trabajo para ese Sprint (Alaimo, 2013) (Schwaber & Sutherland, 2013).

La Planning se divide en dos grandes momentos:

- El QUÉ: a qué nos vamos a comprometer a entregar en el Sprint. Es una negociación donde se deciden las funcionalidades a realizar.
- El CÓMO: cuáles actividades tenemos que realizar para llevar a feliz término las funcionalidades con las que se comprometió el equipo. Esta descomposición se debe hacer en pequeñas tareas, que al estimar no tengan un esfuerzo mayor a 8 horas.

5.3.4.4.2 Sprint

El Sprint es el corazón de Scrum, un periodo de tiempo de máximo un mes, en donde el equipo de trabajo se encarga de desarrollar un incremento del producto final, potencialmente entregable, si el Product Owner lo solicita se debe requerir un mínimo esfuerzo para que pueda estar disponible al cliente. Después de que culmine un Sprint se debe iniciar con el siguiente.

En cada Sprint se realizan eventos tales como el Sprint Planning, el Daily Scrum, el trabajo de desarrollo, el Sprint Review y la retrospectiva (Schwaber & Sutherland, 2013).

Durante cada Sprint:

- No se realizan cambios que podría poder en riesgos la meta del Sprint.
- Las metas en cuanto a calidad no disminuyen.
- El alcance se puede ajustar y renegociar entre el Product Owner y el equipo de desarrollo.

5.3.4.4.3 Daily Meeting o Reunión diaria

La reunión diaria o Daily, es una ceremonia que tiene como objetivo principal, que el equipo de desarrollo sincronice sus actividades y cree un plan para las siguientes 24 horas. Esto se lleva a cabo inspeccionando el trabajo avanzado desde la última reunión diaria, y haciendo una proyección acerca del trabajo que podría completarse antes de la siguiente reunión (Schwaber & Sutherland, 2013) (Alaimo, 2013).

En este espacio de tiempo se proponen responder tres preguntas básicas:

- ¿Qué he hecho?
- ¿Qué voy a hacer?
- ¿Qué dificultades/impedimentos tengo?

Se recomienda que esta reunión tenga una duración de 15 minutos, no debe ser prolongada para no afectar los tiempos del equipo de desarrollo.

5.3.4.4.4 Sprint Review

Ceremonia que se realiza al final de cada Sprint, es llamada Revisión o Review. El objetivo de esta ceremonia es presentar al cliente las funcionalidades con las que se comprometió el equipo para el Sprint, haciendo una revisión y aceptación del incremento del producto entregado al Product Owner y a los Stakeholders.

Se recomienda que la duración de la Review, sea de 1 hora por cada semana de duración del Sprint (Schwaber & Sutherland, 2013).

5.3.4.4.5 Sprint Retrospective o Retrospectiva

En esta ceremonia el equipo de trabajo se reúne para analizar cómo ha sido la manera de trabajar del equipo durante el Sprint, por qué está consiguiendo o no los objetivos a los que se comprometió al inicio del Sprint, y por qué el incremento de producto que acaba de demostrarse al cliente era lo que él esperaba o no. En resumen, el objetivo de esta reunión es mejorar de manera continua la productividad del equipo y la calidad del producto que se está desarrollando, ver qué lecciones pueden obtener de esa experiencia para que pueda ser aplicado en los próximos trabajos. Esta ceremonia se realiza al final de cada Sprint, después de la Review (Schwaber & Sutherland, 2013).

5.3.4.4.6 Refinar historias de usuario

Ceremonia en la cual todo el equipo Scrum, analiza las historias de usuario incluidas en el Product Backlog que se trabajarán en los próximos Sprints, de acuerdo a su priorización, para lograr tenerlas totalmente claras a la hora de realizar la planeación de los siguientes Sprints. El objetivo es realizar ajustes a las historias de usuario del Product Backlog, añadiendo requisitos, modificándolos, eliminándolos, re-priorizándolos y detallando las funcionalidades conforme se acerca el momento de su desarrollo (Schwaber & Sutherland, 2013).

5.3.4.5 Herramientas

5.3.4.5.1 Product Backlog

El Product Backlog es el conjunto de requisitos funcionales y no funcionales que debe cumplir el producto una vez entregado. Por lo general, se suelen organizar los requisitos en Historias de Usuario. Las Historias de Usuario son utilizadas para la especificación de un requisito, son una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos, también permiten responder rápidamente a los requisitos cambiantes. Están escritas en lenguaje coloquial al ser, simplemente, el recordatorio de la conversación con el cliente, y un acuerdo formal de mínimos para dar por buena la funcionalidad describa y esperada (Schwaber & Sutherland, 2013) (Florez & Ardila, 2014).

5.3.4.5.2 Sprint Backlog

Es el listado de las funcionalidades que se van a construir durante el Sprint y cada una de las actividades para lograr el objetivo. El Sprint Backlog hace visible todo el trabajo que el equipo de desarrollo identifica como necesario para cumplir la meta del Sprint. Es un plan con suficiente detalle para que los cambios en el progreso se puedan entender en el Daily.

Cada que se requiera un nuevo trabajo, el equipo de desarrollo lo agrega a la lista de espera. A medida que se realiza o se completa el trabajo, se actualiza el trabajo restante estimado (Schwaber & Sutherland, 2013).

5.4 Comparativo entre metodologías tradicionales y ágiles

Tabla 1. Comparativo entre metodologías tradicionales y ágiles

Metodología Ágil	Metodología Tradicional
Menos artefactos	Más artefactos
Pocos roles	Más roles
Contratos no son tradicionales (son más flexibles)	Contratos predefinidos (camisa de fuerza)
El cliente es parte activa durante el proceso	Cliente participa de formas más mesurada
Grupos pequeños < 10. Si un equipo es superior a esa cantidad de personas lo que se hace es formar subgrupos y utilizar técnicas que permitan la colaboración y comunicación entre estos.	Grupos grandes

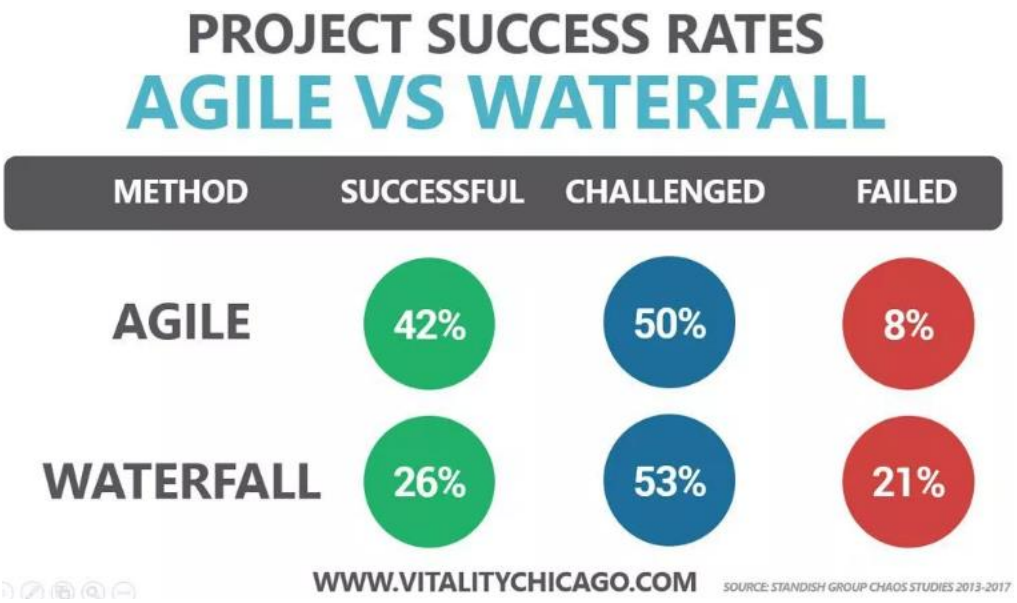
Arquitectura no es esencial.	Arquitectura es un factor fundamental
Requerimientos cambiantes	Requerimientos estables
Proyectos dinámicos	Gran alcance

5.4.1 Reporte del caos (CHAOS Report)

Según los últimos resultados del estudio realizado el 2018 por The Standish Group, que muestran los índices de éxito y de fracaso de los proyectos en cascada y ágiles, se observa, al igual que en los últimos años, que los proyectos agiles estadísticamente tienen más probabilidad de éxito en comparación a los proyectos en cascada (tradicionales).

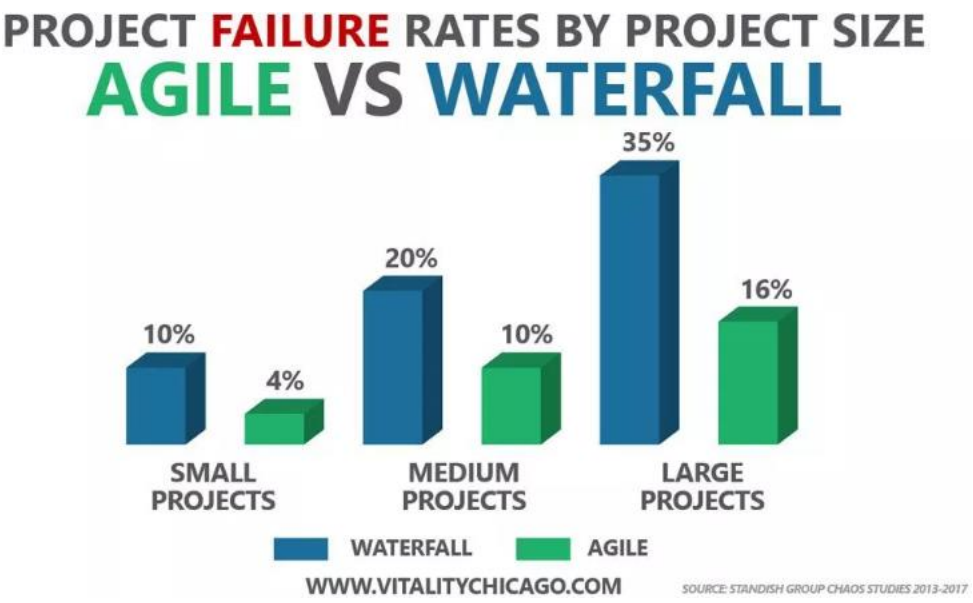
A continuación, los resultados de dicho estudio:

Figura 5. Tasa de éxito Ágil vs Tradicional



Nota: Fuente https://vitalitychicago.com/blog/agile-projects-are-more-successful-traditional-projects/ (Agile Projects are More Successful than Traditional Projects, 2018)

Figura 6. Tasa de fracaso por tamaño del proyecto Ágil vs Tradicional



Nota: Fuente https://vitalitychicago.com/blog/agile-projects-are-more-successful-traditional-projects/ (Agile Projects are More Successful than Traditional Projects, 2018)

5.4.2 Resultado

Después de la labor de investigación con respecto a las metodologías de desarrollo que se utilizaron para la realización del proyecto CUBIC, se determinó realizar la toma de ciertos elementos de dos metodologías, una ágil (Scrum) y la otra robusta (RUP), con el fin de realizar un hibrido de estas dos metodologías, esto se decidió ya que el tiempo para el desarrollo de este proyecto no da para implementar todos los requisitos que tiene que emplearse para la elaboración del mismo utilizando una metodología robusta como es RUP.

Para ello se utilizaron artefactos y procesos de Scrum, el cual ayudó a realizar un desarrollo con alta calidad y de forma ágil, apoyando ciertos procesos de documentación con estándares de RUP.

En la sección de Metodologías del presente trabajo de grado, se explicarán cada uno de los procesos y artefactos que se emplearon de cada metodología, análisis que se tomó con el apoyo de algunos artículos que trataban exactamente de la convivencia entre estas dos metodologías (Cadelli & Fernández, 2015); (Villanueva & Siachoque, 2014); (Castillo, Barrios, Montilva, & Rivero, 2010).

5.5 .NET Framework

.NET Framework es una tecnología de código abierto, multiplataforma y gratuita que permite crear diferentes tipos de aplicaciones. Está disponible en diferentes idiomas, cuenta con múltiples editores y bibliotecas para compilar diferentes tipos de aplicaciones, como por ejemplo: aplicaciones de escritorio, aplicaciones web, dispositivos móviles, gaming y IoT (internet de las cosas) (Microsoft, s.f.).

.NET Framework está diseñado para cumplir con los siguientes objetivos (Latham, Wenzel, Wagner, tompratt, & Jones, 2017):

- Proporcionar un entorno ideal para la programación orientada a objetos.
- Proporcionar un entorno de ejecución de código que minimice la implementación del software y los conflictos en el control de versiones.
- Proporcionar un entorno de ejecución que promueva la ejecución segura de código, incluido el código creado por terceros.
- Proporcionar un entorno de ejecución que elimine los problemas de rendimiento.
- Para hacer que la experiencia del desarrollador sea coherente en diferentes tipos de aplicaciones, como las basadas en Windows y las aplicaciones web.
- Desarrollar todo sobre estándares de la industria para garantizar que el código basado en .NET Framework se integre con cualquier otro código.

Con este Framework se desarrolló el prototipo propuesto en este proyecto de grado, principalmente porque es el entorno de desarrollo utilizado por la empresa Dvalor S.A.S., y se requiere que la plataforma en la que sea desarrollada la aplicación pueda ser conocida y soportada por los analistas que actualmente cuenta la empresa, la empresa además cuenta con las herramientas para emplear dicha tecnología.

5.5.1 Versión del .NET Framework

Existen diferentes versiones del .NET Framework, la versión influye en diferentes cosas, como, por ejemplo, la versión del IDE (entorno de desarrollo integrado) que se esté usando para

desarrollar software, las características a las cuales deseas acceder, y por prerrequisitos de algunas herramientas y plugins adicionales que se integran con el IDE.

La versión que se utilizó en este proyecto es la versión 4.6.2. ya que es una versión que es compatible con el IDE que se empleó y además se quiere contar con las mejoras de rendimiento y correcciones de posibles bugs que normalmente vienen integradas en las actualizaciones de las versiones. Otra razón es que la plataforma llamada SonarQube dentro de sus prerrequisitos exige que esa sea la versión del .NET Framework.

5.5.2 IDE

El IDE (entorno de desarrollo integrado) que es la aplicación que proporciona las herramientas y los servicios para facilitar el desarrollo de software, que se utilizó es el Visual Studio 2015 Community, que es una edición gratuita para estudiantes, desarrolladores independientes o empresas pequeñas (Microsoft, s.f.), sin embargo, en caso tal, la empresa Dvalor S.A.S. cuenta con la edición del IDE Professional el cual está licenciado y las aplicaciones son totalmente compatibles sin importar las ediciones del IDE en que se desarrollen.

5.5.3 Lenguaje de programación

El lenguaje predominante en .NET Framework es C#. C# es un lenguaje elegante, con seguridad de tipos y orientado a objetos, que permite a los desarrolladores crear una gran variedad de aplicaciones seguras y sólidas que se ejecutan en .NET Framework. Se puede usar para crear aplicaciones cliente de Windows, servicios web, XML, componentes distribuidos, aplicaciones cliente-servidor, aplicaciones de bases de datos, entre otros (Cai, 2015).

5.5.4 ASP.NET

ASP.NET es un Framework web de código abierto que contiene todas las herramientas y librerías necesarias para permitir crear y desarrollar aplicaciones y servicios web en .NET Framework. ASP.NET crea sitios web basados en HTML5, CSS y JavaScript.

5.5.4.1 Scaffolding

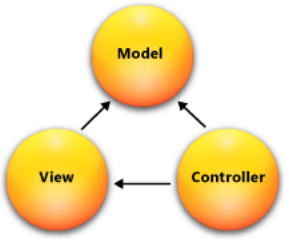
Para la creación de la gran parte de las vistas y los controladores de las vistas de la aplicación CUBIC se realizó con Scaffolding. Scaffolding lo que hace es que, gracias a una serie de elementos de datos dinámicos, permite la creación de aplicaciones web, la cual se encarga de generar automáticamente las páginas web para cada tabla de la base de datos (Microsoft, s.f.), creando el CRUD del modelo de la base de datos (crear, editar, eliminar, listar).

Esta herramienta se utilizó ya que permitía agilidad en el desarrollo de la aplicación, creando la base de la cual se partiría para continuar con el desarrollo.

5.5.5 MVC

ASP.NET MVC facilita la implementación del patrón arquitectónico MVC (Model-View-Controller), ya que permite la creación de aplicaciones web basadas en esta arquitectura, separando la aplicación en esos tres componentes principales: el modelo, la vista y el controlador. El Framework ASP.NET MVC tiene una presentación liviana el cual se integra con las características existentes de ASP.NET, como las master pages y la autenticación basada en la membresía utilizadas en el modelo tradicional de ASP.NET el cual está basado en formularios Web Forms y postbacks (devolución de datos) (Microsoft, 2013).

Figura 7. Patrón de diseño MVC



Nota: Fuente https://docs.microsoft.com/en-us/previous-versions/aspnet/web-frameworks/dd381412(v=vs.108) (ASP.NET MVC Overview, 2013)

Se decidió utilizar este patrón de diseño debido a que va alineado con el patrón arquitectónico (Model-View-Controller) que se empleó en la aplicación, además de facilitar la

implementación del código aprovechando todas las ventajas que ya vienen en si incorporadas en este modelo, tanto, por ser un modelo ampliamente utilizado en el desarrollo de las aplicaciones en la empresa Dvalor S.A.S. por lo tanto debemos estar alineados con la empresa, de igual forma, el uso de MVC es la mejor opción para llevar a cabo el desarrollo de la aplicación desde el IDE Visual Studio teniendo en cuenta las características de la aplicación a desarrollar, la arquitectura que se va a usar, y la empresa que lo va a utilizar.

5.5.6 ADO.NET

ADO.NET es un conjunto de clases que exponen servicios de acceso a datos para .NET Framework. Proporciona acceso a origines de datos como SQL Server y XML, así como orígenes de datos expuestos mediante OLE DB y ODBC. Las aplicaciones pueden utilizar ADO.NET para conectarse a estos orígenes de datos y recuperar, controlar y actualizar datos.

ADO.NET separa el acceso de datos de la manipulación de datos y crea componentes discretos que se pueden utilizar por separado o conjuntamente, brindando una capa de seguridad adicional a la aplicación ya que no estaría permitiendo el acceso directo a los datos en el lenguaje nativo de la fuente de datos que se esté empleando (Cai, ADO.NET Overview, 2017).

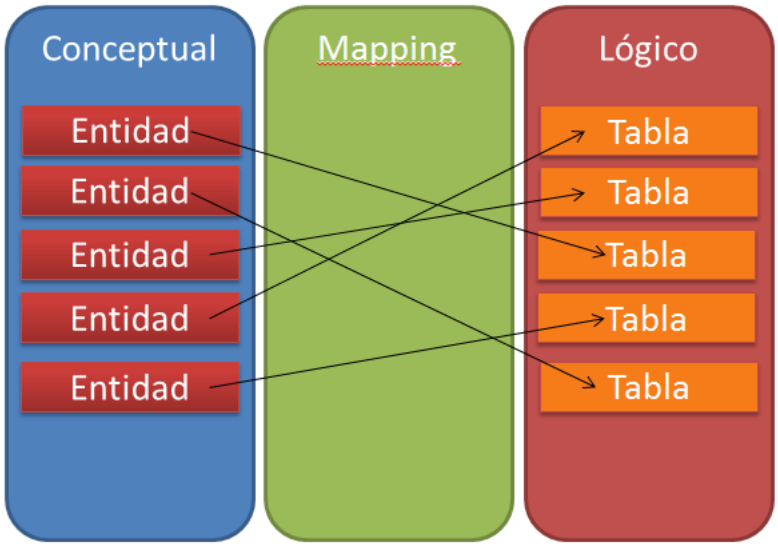
5.5.7 ORM

ORM (Object-Relational Mapping) en el desarrollo de capas de acceso de datos es una técnica de programación que mapea una base de datos relacional a una serie de objetos que puede ser manipulados desde los lenguajes de programación orientados a objetos utilizados tradicionalmente hoy en día para el desarrollo. Lo que se pretende con ORM es facilitar y automatizar el proceso de desarrollo manual permitiendo tener a la disposición modelos y métodos de acceso a datos ya predefinidos con lo cual, el desarrollador ya no tendría que invertir tiempo en crear y diseñar la capa de acceso a datos por completo, agilizando a su vez el desarrollo de la aplicación (Catalá & Puig, 2017).

5.5.7.1 ADO.NET Entity Framework

Entity Framework (EF) es el ORM más popular de Microsoft. Entity Framework es un conjunto de librerías de clases y herramientas que permite a los desarrolladores crear aplicaciones de acceso de datos, pero en vez de programar contra modelos relacionales para acceder a estos datos, se pasa a programar contra esquemas o modelos conceptuales, de modo que el desarrollador puede trabajar sobre el modelo conceptual de la aplicación y centrarse en "lo que quiere conseguir", y no perder tiempo en "como lo quiere conseguir" (Catalá & Puig, 2017).

Figura 8. Mapeo entre el modelo conceptual y el modelo lógico



Nota: Fuente https://msdn.microsoft.com/es-es/communitydocs/net-dev/dev/buenas-practicas-parte3 (Buenas prácticas de codificación para capas de acceso a datos de aplicaciones (III), 2017)

Dicho esto, EF es utilizado ampliamente en el desarrollo de la aplicación, para el acceso de datos tanto del CRUD de las vistas, como de la lógica de los controladores, debido a su versatilidad y facilidad de uso.

5.5.7.1.1 Lazy Loading, Eager Loading

Lazy Loading y Eager Loading son técnicas para la recuperación de datos a través de Entity Framework (EF), que consiste básicamente en la carga de las entidades relacionadas. El uso de cada técnica depende de la necesidad que se requiera en el proyecto.

Lazy Loading viene habilitado por defecto en EF, cuando se accede al contexto de la base de datos con EF, si se requiere acceder a los datos de una entidad, se hace un llamado a la base de datos.

Lazy Loading viene habilitado por defecto en EF, sin embargo, para el proyecto se deshabilitó la carga diferida (Lazy Loading) ya que se pretende disminuir la cantidad de roundtrips de peticiones realizadas a la base de datos. Cuando Lazy Loading está activo, si se quiere acceder a los datos de las entidades relacionadas en alguna entidad padre, se debe volver a consultar la base de datos para hacer el llamado de dichos datos, este esquema se vuelve critico cuando la cantidad de datos en las entidades relacionadas son grandes, lo que percute a un alto consumo de memoria en la aplicación afectando su rendimiento.

Al realizar dicha des-habilitación, se pasa de estar con la técnica Lazy Loading a Eager Loading o carga previa, lo cual permite que si se necesita obtener una entidad junto a sus entidades relacionas se puede hacer realizando un solo llamado a la base de datos incluyendo el método Include al realizar el llamado de las entidades desde el contexto de la base de datos.

Con esto se logra disminuir la cantidad de datos cargados en el contexto y en memoria, y solo se cargarán más datos solo cuando el proceso lo requiera (Lerman, 2011).

5.5.7.1.2 Data Annotations

Los Data Annotation permiten configurar los modelos o entidades de la aplicación web, agregándole propiedades o validaciones a los componentes de las entidades por medio de atributos. ASP.NET MVC aprovecha estas anotaciones para realizar validaciones del lado del cliente. Una de las ventajas de usar esta característica de Data Annotation es que, al aplicar atributos de datos y validaciones, se puede administrar la definición de datos en un solo lugar sin necesidad de volver a escribir las mismas reglas en todas las clases y vistas que lo utilicen (tutorialspoint, s.f.) (Soji, 2014).

En el desarrollo del proyecto se hace uso de Data Annotation para incluir atributos a las entidades para realizar validaciones de tipos de datos, de características y propiedades de los campos, entre otras.

5.5.7.1.3 Modos de trabajo principales de EF

EF puede funcionar de varias maneras diferentes a la hora de mapear las clases de la aplicación orientado a objetos y las tablas en la base de datos (Alarcón, 2018).

Existen tres modos principales de trabajo con Entity Framework.

- Database First

  Database First funciona teniendo una base de datos ya creada y diseñada, es decir con una base de datos existente, y lo que se va a hacer es que EF se va a encargar de generar las clases necesarias juntos a sus componentes para poder trabajar con las entidades relacionadas a la base de datos seleccionada.

- Model First

  En este modo las entidades con EF se crean a partir de modelos creados con el diseñador de modelos de Visual Studio, no hay necesidad de escribir código ni de tener una base de datos ya existente.

- Code First

  En este modo, definimos las clases mediante código, y EF se encarga de generar la base de datos y todo lo necesario para encajar las clases en ellas.

Evaluando las ventajas y desventajas de cada modo de mapeo de entidades, y teniendo en cuenta la experiencia en este tipo de aplicaciones, se decidió emplear Code First, ya que permite mantener sincronizado el código con la base de datos, no se eliminarían los atributos configurados con Data Annotations en las entidades para las validaciones de sus propiedades, el proceso de ajuste y cambios hacia la base de datos es muy sencillo de realizar, y además no se genera tanto código permitiendo mantener una aplicación más limpia en cuanto al manejo de las entidades y el contexto de la base de datos con EF se refiere.

5.5.7.1.4 EF Migrations

Migrations es una herramienta de EF empleada principalmente cuando se hace uso de Code First, ya que esta herramienta es la que permite sincronizar la base de datos con respecto a los cambios que se hayan realizado en las clases de entidades o el contexto.

Su manejo y funcionamiento es muy sencillo, a través de comandos en la consola del NuGet del Visual Studio se ejecutan los comandos para habilitar las migraciones, crear migraciones, realizar rollbacks de las migraciones, actualizar la base de datos con respecto a la última migración creada o con relación a una en particular, entre otras operaciones (Microsoft, 2016).

5.6 Team Foundation Server (TFS)

TFS proporciona un conjunto de herramientas de desarrollo de software de colaboración que se integran fácilmente con el IDE Visual Studio, lo que permite al equipo de desarrollo trabajar de manera eficiente en proyectos de software de cualquier tamaño (Microsoft, s.f.).

Es el repositorio de código fuente utilizado para el proyecto, en donde cada miembro del equipo de desarrollo tendrá acceso a las aplicaciones que se creen en el TFS, permitiendo trabajar simultáneamente y respaldando siempre el código fuente en la nube que dispone Microsoft. Se hace uso del TFS como repositorio de código fuente y como administrador de aplicaciones ya que es el utilizado por la empresa Dvalor S.A.S. para gestionar sus aplicaciones y llevar un control de las versiones de las mismas.

5.7 Internet Information Services (IIS)

El servidor de aplicaciones web utilizado es IIS, en donde se publicó la página web que se desarrolló en el proyecto para que este pueda quedar disponible para los usuarios dentro de la intranet de la empresa Dvalor S.A.S. Este es el servidor web utilizado por Dvalor S.A.S. para las demás aplicaciones web que tiene actualmente en ejecución.

5.8 Motor de bases de datos

El sistema de gestión de bases de datos utilizado es SQL Server de Microsoft, que emplea el lenguaje de base de datos Transact-SQL o T-SQL, que es un lenguaje potente que incorpora programación procedimental y además permite definir numerosas tareas y procesos que se requieran efectuar sobre una base de datos, es una extensión del lenguaje estándar SQL (Khan, 2012),

SQL Server es el gestor de bases de datos definido por la empresa Dvalor S.A.S. al ser el único motor de base de datos que dispone la compañía, no obstante, no existe ninguna limitación o inconveniente para emplear este gestor de bases de datos para el desarrollo del proyecto de grado.

5.8.1 Stored Procedures (Procedimiento almacenado)

Un procedimiento almacenado es conjunto de código programado en SQL que se puede guardar y reutilizar las veces que sean necesarias. Este procedimiento almacenado se almacena dentro de la base de datos.

Si tiene consultas que son usadas frecuentemente puede llamar este procedimiento almacenado para que se ejecute.

Los procedimientos almacenados también pueden recibir parámetros o valores de entrada, los cuales se pueden manipular dentro del procedimiento almacenado para que se realicen acciones dependiendo de los datos contenidos en ellos, o como se requiera en el proceso (w3schools, s.f.).

5.9 Pruebas de cargues masivas a la base de datos

En el proyecto de grado, el acceso a los datos de la base de datos se hace mayormente a través del ORM Entity Framework, debido a su gran facilidad y agilidad al momento de requerir realizar alguna operación de consulta, creación, edición y eliminación, además de la seguridad que ofrece al implantar una capa adicional propiamente de los ORM’s, sin embargo, existe un proceso de cargue de archivos a la base de datos que debido a la gran cantidad de registros que contienen los archivos (en promedio 250.000 líneas), realizar el cargue usando EF para insertarlos a la base de datos no resulta optimo ya que la aplicación consume muchos recursos al ejecutar dicha operación, no solo por la cantidad de datos que se está cargando en memoria sino por las validaciones que se deben realizar a nivel de negocio sobre estos archivos.

Como alternativa, se crearon procedimiento almacenados los cuales reciben un DataTable desde la aplicación con todos los datos contenidos en el archivo, ya validados, y es el procedimiento almacenado quien se encarga de insertar los datos a la base de datos.

Esta operación se necesitó realizar para cargar los archivos de la mezcla y de los distribuidores.

A continuación, se reflejará cada una de las pruebas que se realizaron para tomar la decisión que más se ajustara a lo que se necesitaba en cuanto a rendimiento y confiabilidad de la información:

Tabla 2. Comparación alternativa de cargues masivas en BD

Tipo de prueba	Cantidad registros	Tiempo de ejecución	Resultado
Cargando archivo a un DataTable y luego pasándolo a una lista tipada para después guardarlos en la BD usando EF Batch Operation	250.000	35 segs aprox.	Tiempo de ejecución aceptable sin embargo tiene una desventaja, al usar EF y poder cargar esa cantidad de datos se deshabilitan una serie de configuraciones y parámetros al EF para evitar realizar validaciones de nivel de constraints y demás lo cual disminuye la confiabilidad de la información. ⚠
Cargando archivo directamente a una lista tipada para luego guardarlos en la BD usando EF Batch Operation	250.000	43 segs aprox.	Toma más tiempo de ejecución. ✖
Cargando archivo usando la clase Parallel (bucles paralelos) para luego guardarlos en la BD	250.000	35 segs aprox.	Tiempo de ejecución aceptable sin embargo al realizar la operación en paralelo podría insertar los datos en desorden (es importante para el negocio que se guarden en el mismo orden del archivo) además generaba excepciones en algunos casos, por lo tanto, se descartó. ⚠
Cargando archivo a un DataTable y luego pasándolos a un DataTable con la estructura de la tabla destino para luego guardarlos en la BD usando un SP TVP (procedimiento almacenado que usa Table-Valued Parameters –datos con un tipo creado en BD)	250.000	36 segs aprox.	Tiempo de ejecución aceptable. Fue la opción escogida ya que realizar la inserción de datos directamente desde la base de datos se distribuye la carga de consumo de memoria de la aplicación además se mantienen las validaciones a nivel de base de datos como los constraints y demás. Esta opción también es óptima en caso de requerir cargar más datos del promedio. ✔

5.10 Patrones de arquitectura

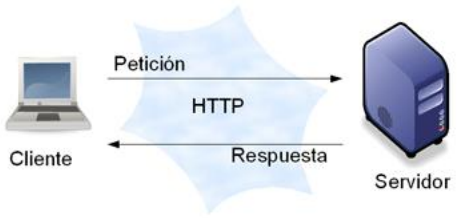
Los patrones de arquitectura definen básicamente como se comunicarán los componentes de la aplicación, los aspectos fundamentales de la estructura de un sistema de software.

Existen diferentes tipos de patrones de arquitectura, todos ellos especifican un conjunto predefinido de subsistemas con sus responsabilidades, y una serie de recomendaciones para organizar los distintos componentes (etsii, s.f.).

5.10.1 Cliente-Servidor

Define una relación entre dos aplicaciones en las cuales una de ellas (cliente) envía peticiones a la otra (servidor y fuente de datos) (Cuéllar, 2010).

Figura 9. Arquitectura Cliente/Servidor



Nota: Fuente http://josecuellar.net/estilos-patrones-basicos-arquitectura-software/ (Estilos y patrones básicos en arquitectura de software, 2010)

En el caso del proyecto, existen varios usuarios que realizan peticiones del lado del cliente, las cuales irán al servidor en donde está publicada la aplicación en el servidor de aplicaciones web IIS (Internet Information Services).

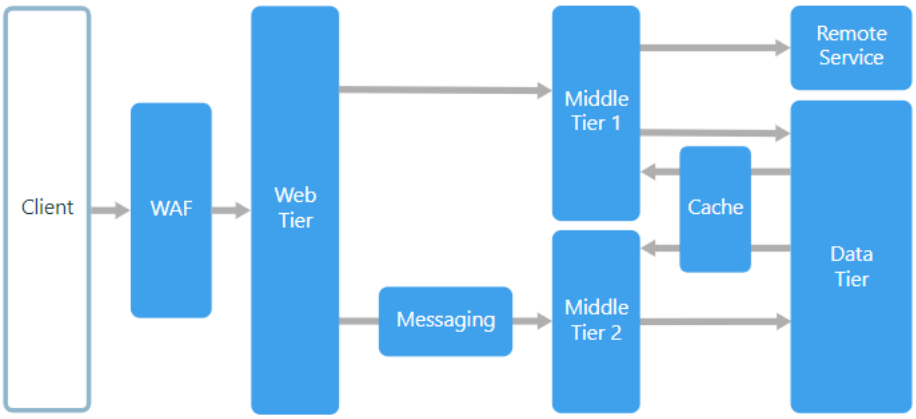
5.10.2 Arquitectura en capas (N-Tier)

Una arquitectura N-Tier divide una aplicación en capas lógicas y niveles físicos.

Las capas son una forma de separar responsabilidades y administrar dependencias. Cada capa tiene una responsabilidad especifica. Una capa más alta puede usar servicios de una capa inferior, pero no al revés.

Los niveles están separados físicamente, funcionando en máquinas separadas. Un nivel puede llamar a otro nivel directamente o usar mensajes asincrónicos. Aunque cada capa puede estar alojada en su propio nivel eso no es obligatorio, varias capas pueden estar alojadas en el mismo nivel. La separación física de los niveles mejora la escalabilidad y la flexibilidad, pero también agrega latencia en la comunicación de red adicional (Wasson, 2017).

Figura 10. Arquitectura N-Tier



Nota: Fuente http://josecuellar.net/estilos-patrones-basicos-arquitectura-software/ (Estilos y patrones básicos en arquitectura de software, 2010)

En el caso del proyecto, la empresa Dvalor S.A.S. cuenta con diferentes servidores, uno donde está alojada la base de datos, otro servidor en donde está instalado el servidor de aplicaciones web IIS, lugar donde se publicará la aplicación, y los usuarios accederán a la aplicación a través de sus equipos de cómputo por medio de la intranet de la compañía. Este modelo es exigido por la empresa ya que de esa forma está definido desde las áreas de seguridad informática e infraestructura.

5.11 Patrones de diseño

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software, en resumen, lo que hace es ofrecer alternativas ya probadas y documentadas, para solventar problemas o riesgos en el desarrollo de software, no hay que inventar una solución que puede que no sea optima o genere nuevos inconvenientes, ya existe una serie de

patrones de diseños con sus ventajas y consecuencias, solo hay que seleccionar la que más se ajuste al proyecto (Tedeschi, s.f.).

5.11.1 Strategy

El patrón de diseño Strategy (Estrategia) decide como interactuar dentro de la aplicación e intercambiar mensajes entre diferentes objetos de clases para realizar un proceso especifico, con el uso de Interfaces se delega a la clase padre decidir qué comportamiento realizar e incluso permite cambiar su comportamiento de manera dinámica, en tiempo de ejecución (Ruiz Pacheco, 2017).

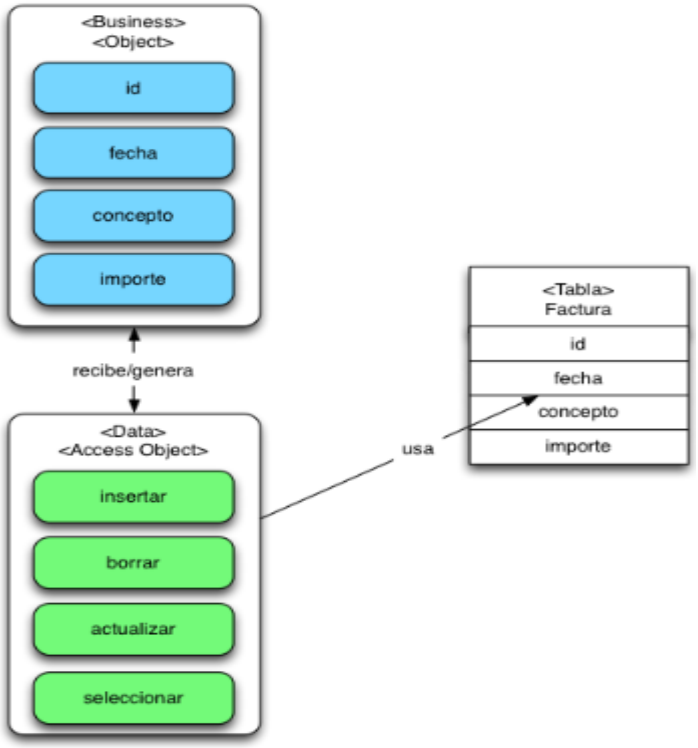
Este patrón de diseño es ampliamente utilizado en la aplicación, debido a que permite darle solución a uno de los problemas más complicados que ha tenido al área de tecnología de la empresa Dvalor S,A,S, con las aplicaciones que actualmente tienen desarrolladas, ya que cada cliente puede tener lógicas o comportamientos diferentes pero todas estas no están separadas en objetos individuales lo que ocasiona que el mantenimiento del código sea más exhaustivo y delicado, con la posibilidad de afectar la lógica de otros clientes mientras se agrega nuevas funcionalidades o se ajustan las ya existentes solo para un cliente especifico. Además, existen procesos en donde su comportamiento varía dependiendo de cómo se tendrán que producir en producción, como es el caso de las formas de impresión de los productos, en donde los productos pueden variar su forma de impresión cada vez que entra uno nuevo a producción.

5.11.2 DAO

DAO (Data Access Object) es un patrón de diseño que consiste en separar las responsabilidades de la aplicación en donde se tendrán las clases que se encargarán de la lógica de negocio y otras clases que se encargarán de la persistencia de datos (Álvarez, 2014).

De esta forma se aislará completamente la aplicación sin necesidad de hacer uso de SQL puro dentro de la aplicación ya las clases de persistencia serán las encargadas de acceder a los datos de la base de datos por medio del ORM Entity Framework, dándole seguridad a la aplicación y mayor flexibilidad.

Figura 11. Distribución Patrón de diseño DAO



Nota: Fuente https://www.genbeta.com/desarrollo/patrones-de-diseno-active-record-vs-dao (Patrones de Diseño (Active Record vs DAO), 2014)

5.11.3 MVC

MVC o Modelo-Vista-Controlador (Model-View-Controller) es un patrón de diseño enfocada en sistemas donde requiere uso de interfaces de usuario, las cuales pueden ser tanto formularios en páginas web, como alguna aplicación de comandos, entre otras, es decir, cualquier medio en donde el usuario tenga que interactuar directamente con el sistema (Alvarez, 2014) (Universidad de Alicante, s.f.).

Su base es la separación de la aplicación en tres componentes principales: los datos de la aplicación, la interfaz de usuario y la lógica de control.

- Modelo

  Contiene la representación de los datos que maneja el sistema.

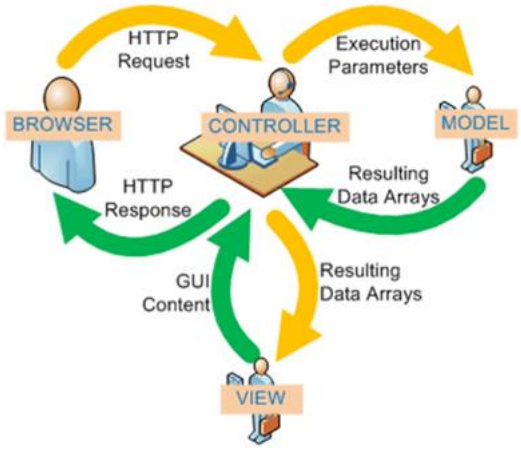
- Vista

Contiene los componentes con la información que se envía al cliente y los mecanismos de interacción con este, es decir, contiene la interfaz de usuario.

- Controlador

  Es el intermediario entre el Modelo y la Vista, gestionando el flujo de información que viaja entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

Figura 12. Diagrama MVC



Nota: Fuente https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html (Modelo vista controlador (MVC))

Este patrón de diseño se implementó en el proyecto debido a las características importantes en la separación de componentes dentro de la aplicación que permitía un mayor control e implementación durante el desarrollo, además MVC está totalmente integrado con Visual Studio, y este proyecto se creó desde el Visual Studio usando la plantilla MVC que viene por defecto en el IDE, aprovechando todas sus características.

5.11.4 Inyección de dependencias

La inyección de dependencias (DI) es un patrón de diseño que consiste en extraer responsabilidades a un componente para delegarlas en otro, estableciendo un mecanismo a través del cual el nuevo componente pueda ser cambiado en tiempo de ejecución (Ruiz, 2017).

Este patrón de diseño comúnmente es utilizado por otros patrones de diseño, en este caso es empleado en el patrón de diseño Estrategia que se está empleando en este proyecto de grado.

5.12 Pruebas unitarias (Unit Test)

Las pruebas unitarias o Unit Testing son trozos de código diseñados para probar que el código de la aplicación está funcionando correctamente. En estos test se espera que superen las pruebas en la ejecución y que el resultado obtenido sean los esperados (Apiumhub, 2017).

Características de las pruebas unitarias:

- Automatizable; Aunque los resultados deben ser específicos de cada test unitario desarrollado, los resultados se pueden automatizar, de forma que podemos hacer las pruebas de forma individual o en grupos.
- Completas; El proceso consta de pequeños test sobre parte del código, pero al final, se debe comprobar su totalidad.
- Repetibles; En el caso de repetir las pruebas de forma individual o grupal, el resultado debe ser siempre el mismo dando igual el orden en que se realicen los test, los tests se almacenan para poder realizar estas repeticiones o poder usarlos en otras ocasiones.
- Independientes; Es un código aislado que se ha creado con la misión de comprobar otro código muy concreto, no interfiere en el trabajo de otros desarrolladores.
- Rápidos de crear; a pesar de lo que muchos desarrolladores opinen, el código de los tests unitarios no debe llevar más de 5 minutos en ser creado, están diseñados para hacer que el trabajo sea más rápido.

Estás pruebas unitarias se crearon desde el Visual Studio, ya que proporciona una forma flexible y eficaz de ejecutar las pruebas unitarias y ver los resultados en Visual Studio (Microsoft, 2015).

5.13 Visual Studio Team Services (VSTS)

Visual Studio Team Services es un conjunto de herramientas de colaboración basado en la nube que sirve para planear, desarrollar y administrar proyectos de software de cualquier tamaño,

en cualquier lenguaje de programación de software. Usando las capacidades de Team Foundation Server, junto con servicios en la nube adicionales, Team Services ayuda a administrar el código fuente, elementos de trabajo, compilaciones, pruebas y otros recursos (Microsoft, s.f.).

5.14 Integración continua

Integración continua es una práctica en la que miembros de un equipo integran su trabajo frecuentemente, típicamente cada persona íntegra al menos una vez al día, generando varias versiones al día. Cada versión ejecutable es verificada por un sistema automático de integración y pruebas para detectar errores de integración lo más rápido posible (Galván, 2018).

Beneficios:

- Reducir problemas de integración
- Mejorar la visibilidad del estatus del producto de software
- Acelerar la detección de fallas
- Disminuir el tiempo dedicado a depurar errores
- Evitar la espera para averiguar si un código funciona

La integración continua en el proyecto se está realizando desde el VSTS aprovechando su valiosa integración con Team Foundation Server, repositorio de código fuente que es utilizado por la empresa Dvalor S.A.S. Dentro de los procesos automáticos que se ejecutan en la integración, se incluyen los compiladores de la solución, la detección de fallas si estas están presentes, y la ejecución automática de todas las pruebas unitarias incluidas en la solución.

6 Metodología

6.1 Tipo de investigación

El presente proyecto de desarrollo propuesto, se llevó a cabo a través de una investigación aplicada, la cual permitió alcanzar cada uno de los objetivos específicos, dando como resultado una aplicación que cumpla con las expectativas planteadas.

Este tipo de investigación se caracteriza porque busca la aplicación o utilización de los conocimientos que se adquieren, con el objetivo de resolver problemas prácticos y útiles, he aquí en donde este tipo de investigación se define claramente y se ajusta al proyecto a desarrollar.

6.2 Fuentes de recolección

Se realizaron encuestas en las diferentes áreas involucradas en el proceso, con esta información se hizo un análisis inicial de la situación, la cual posteriormente se extenderá a otros actores que interactúan en la situación.

Estos actores pueden ser el área de planeación, calidad, diseño, procesamiento, despachos, producción y por supuesto el área de tecnología, además de los directivos y líderes interesados en el proceso ejecutado.

Se realizó un estudio de conocimiento del dominio de la temática, que se alimentará de las demás fuentes de recolección que se llevaron a cabo.

Se revisó la existencia de sistemas actuales y anteriores, con el fin de detectar el enfoque que desarrollaron, la problemática que intentan solucionar, lo que permitió a su vez tener una guía importante en el desarrollo planteado, analizando las diferencias que se ofrecen con la aplicación propuesta y las ventajas que se tendrían.

6.3 Metodología de desarrollo

Para el desarrollo de la aplicación, se tomaron elementos de la metodología RUP y se combinaron con elementos de la metodología SCRUM. El objetivo de esta mezcla de elementos entre dos metodologías (ágil -SCRUM- y tradicional -RUP-), es combinar técnicas ágiles que

permita realizar los procesos de desarrollo de una manera más rápida y ágil, permitiendo además tener un contacto más frecuente con el cliente con el fin de minimizar ambigüedades durante toda la fase de desarrollo, por otro lado, los elementos de la metodología tradicional brinda herramientas de prevención de riesgos y permite tener un panorama más claro desde el inicio de lo que se requiere en el proyecto.

Las fases empleadas se sacaron de RUP, y en ellas se listará cada uno de los artefactos que se emplearon en cada fase y de cual metodología proviene:

Inicio

- Documento visión: Es el documento que se llevó a cabo durante el trascurso de la asignatura de Anteproyecto.
- Roles (tomado de Scrum):
    - Cliente: Producción juegos, planeación, calidad, despachos, procesamiento, directivos y líderes interesados en el proceso.
    - Product Owner: Ana Lorena Gutiérrez (Coordinadora de TI).
    - Scrum Master: Erwin Oquendo
    - Team Development: Fabián Velásquez y Erwin Oquendo.
- Especificación de Requerimientos: Product Backlog e Historias de Usuario (tomados de Scrum). En estos se levantaron todos los requerimientos implicados con el desarrollo, se estimaron y refinaron dependiendo del tamaño de la historia de usuario.

Elaboración

- Casos de uso nivel 0 de los procesos principales (tomado de RUP).
- Diagramas de caso de uso (tomado de RUP).

Construcción

- Documento arquitectura (tomado de RUP), en el cual se trabajaron las siguientes vistas:

- Vista física: Es la distribución física de la aplicación, una descripción del sistema y su interacción.
- Vista lógica: Diagramas de clases, Modelo Entidad Relación.

- Sprint (tomado de Scrum): Se llevó a cabo una serie de iteraciones durante el desarrollo de cada uno de los Sprint que se obtengan de las historias de usuario y de los casos de uso. Se complementarán las historias de usuario con los casos de uso debido a la dificultad que en ciertas o incluso en muchas ocasiones se presentan para tener disponible al cliente durante el proceso de desarrollo, es ahí en donde los casos de uso permitirán obtener mayor detalle en la especificación.

  En esta se realizaron las siguientes etapas:

  - Reunión de planificación de Sprint: Los involucrados en el equipo se reúnen para planificar el Sprint. Durante este evento se decide qué requerimientos o tareas se le asignará a cada uno de los elementos del equipo. Cada integrante deberá asignar el tiempo que crea prudente para llevar a cabo sus requerimientos. De esta manera se define el tiempo de duración del Sprint.
  - Reunión de Equipo de Scrum: Se harán reuniones diariamente con un máximo de 15 minutos. Los medios utilizados para realizar estas reuniones serán herramientas tecnológicas tales como: Skype, Whatsapp, entre otras. En esta reunión cada integrante del equipo responderá tres preguntas:
    - ¿Qué hiciste ayer?
    - ¿Qué tienes planeado hacer hoy?
    - ¿Qué obstáculos encontraste en el camino?

    La idea de estas reuniones es que entre todo el equipo se apoyen entre sí, con el objetivo de cumplir con las metas trazadas. Si existen puntos que tomarán más tiempo se debe programar una reunión a parte en donde se toque a detalle la problemática a resolver.
  - Refinamiento del Backlog: El Product Owner revisa cada uno de los elementos en ejecución con el fin de esclarecer cualquier duda que pueda surgir por parte del equipo de desarrolladores. También sirve para volver a estimar el tiempo y esfuerzo dedicado a cada uno de los requerimientos.

- Revisión del Sprint: Los miembros del equipo y los clientes se reúnen para mostrar el trabajo de desarrollo de software que se ha completado. Se hace una demostración de todos los requerimientos finalizados dentro del Sprint.
- Retrospectiva del Sprint: Se reúne todo el equipo del Team Scrum para hablar sobre lo ocurrido durante el Sprint. Los puntos principales a tratar son:
  - Qué se hizo mal
  - Qué se hizo bien
  - Qué inconvenientes se encontraron

Al final de las fases anteriormente mencionadas, se llevó a cabo pruebas de funcionamiento para verificar la exactitud de la solución desarrollada.

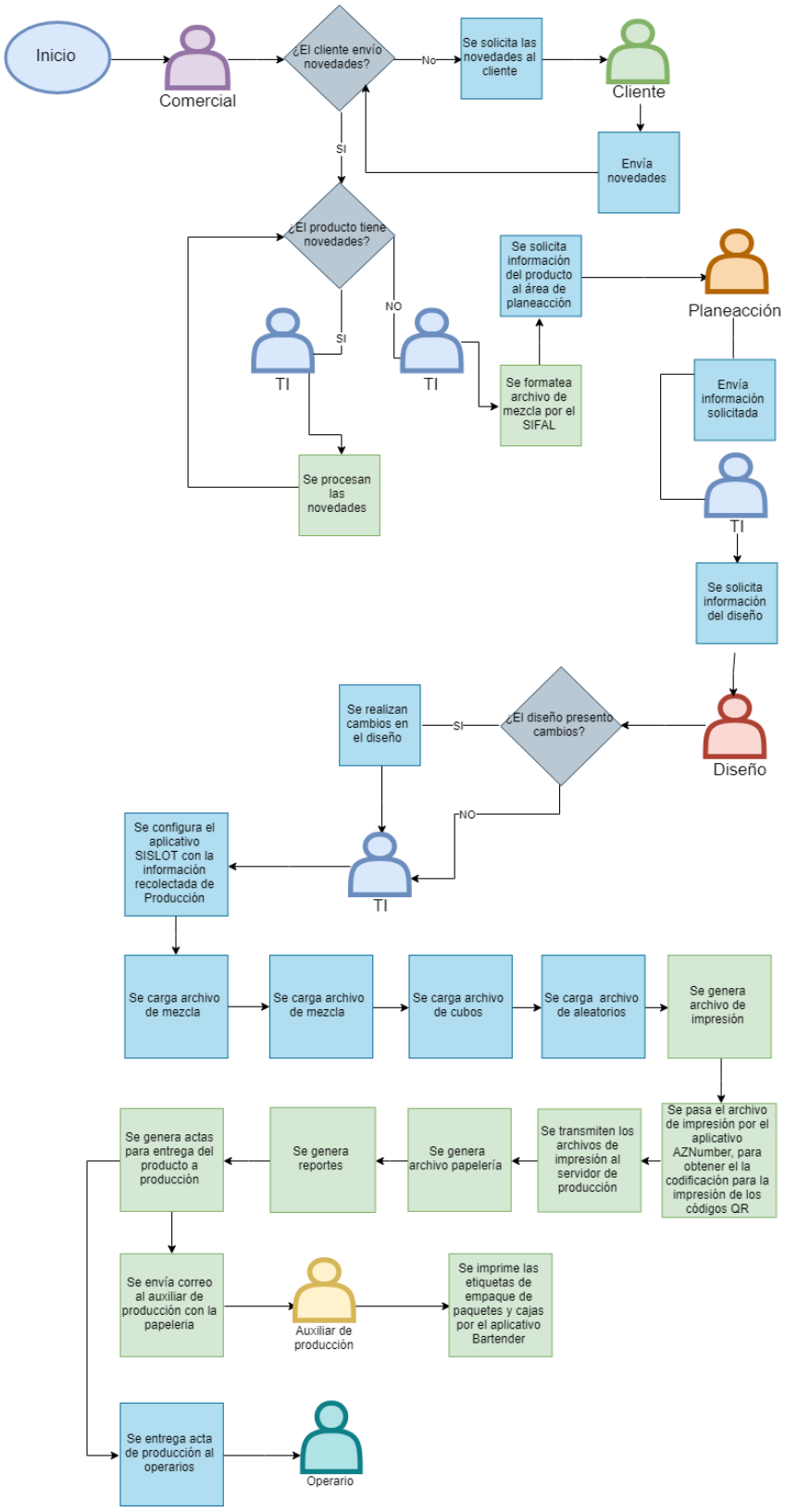
7 Resultados

A continuación, se explicará el resultado del análisis que se realizó en los procesos, hallazgos y aplicaciones que se utilizaban con el modelo anterior a implementar este proyecto, lo cual ayuda, además, a tener una visión clara de los puntos de enfoque en donde se puede mejorar el proceso.

7.1 Flujo del proceso anterior

El procesamiento de los productos de loterías dependía mucho del área de TI Producción Cali quienes eran los encargados de levantar toda la información del producto con las diferentes áreas que se veían involucradas en el proceso, esta responsabilidad era asumida por el personal de esta área ya que la arquitectura del anterior sistema estaba diseñada para que la configuración de cada producto se realizara por un usuario experto al hacer lineal y no transversal como se plantea en la solución, adicional de solicitar toda la información y realizar la configuración, el personal de TI eran los responsables de cargar los diferentes archivos que la aplicación requería ya que la mayoría se realizaban de forma manual, también eran los encargados de realizar el procesamiento de la información, realizar las actas de entrega de forma manual y liberar a producción, con la solución planteada y como se puede observar en la figura 19, el área de TI sale de la ecuación de configuración y procesamiento del producto ya que cada área es responsable de configurar la parte que le corresponda del producto por medio de la aplicación realizada. En la siguiente figura, se relaciona el anterior flujo para el proceso de la información que se realizaba por el anterior aplicativo.

Figura 13. Flujo anterior en el procesamiento de los productos



Para ampliar sobre el flujo del proceso anterior, ver Anexo 1, en donde se refleja en detalle de qué manera interactúan las diferentes áreas con el proceso actual (como funcionaba antes de implementar este proyecto).

7.2 Comunicación entre las áreas en el proceso anterior

En el proceso para que un producto saliera a producción, se requiere cierta información la cual la suministra cada área involucrada en el proceso.

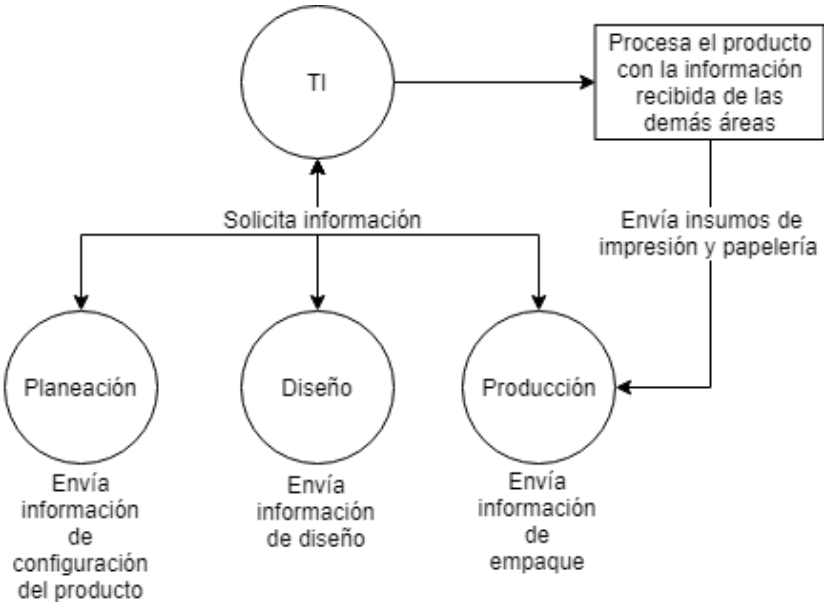
- Área de Planeación: Es la encargada de entregar la información correspondiente a la configuración del producto, en donde se conoce las características que el producto va a tener y de qué forma debe ingresar a la planta de producción.
- Área de Diseño: En esta área se conoce si al producto se le debe crear un diseño nuevo o si se debe actualizar alguno de los existentes.
- Área de TI: En el flujo, el área de TI es el que tiene la mayor carga y es la que está involucrada en casi todo el proceso. Entre las tareas que debía desempeñar están:
    - Gestionar por parte del comercial y los clientes el envío de los insumos iniciales que son cargados en las aplicaciones en donde se procesa la información.
    - Preguntar al área de Planeación qué características debía tener el producto a crear.
    - Crear el producto con su configuración en el sistema.
    - Preguntar al área de Diseño si al producto se le debe realizar un diseño nuevo o si se tiene que modificar uno existente.
        - Si hay que crear un diseño existente debe encargarse de crearlo.
    - Cargar los archivos a la aplicación SIFAL (para formatear la mezcla y los distribuidores), AZ Number (para generar los códigos QR’s) y SISLOT (para cargar los archivos formateados de la mezcla y distribuidores, más otros archivos que son realizados de manera manual).
    - Generar los archivos necesarios para iniciar el proceso de producción, tales como, los archivos de impresión, el listado de máquina, archivos de papelería, archivos para facturación, archivos de respuesta y reportes.

- Generar de manera manual el acta de entrega el cual contiene toda la información necesaria para que los operarios y auxiliar de producción conozcan qué es lo que deben producir y de donde tomar la información que requieren.
- Enviar correo electrónico a las personas pertenecientes a las diferentes áreas a las cuales hay que informarle la disponibilidad de los insumos para entrar a producción con el producto procesado.

- Área de Producción: Es el área encargada de enviar Es el área encargada de cargar los archivos de papelería que envía el área de TI por correo electrónico, para luego imprimirlo y entregar a producción.

El área de TI, además de las funciones que desempeña en su área, debe estar pendiente de que las demás áreas envíen la información que deben suministrar para darle continuidad al producto en producción, lo que ocasiona que en muchas ocasiones estas no envían la información a tiempo o incluso, la envían pero realizan cambios que no son notificadas al área de TI lo que ocasiona que cuando el producto llega a la planta de producción, esta no concuerda con lo que producción espera, generando reprocesos, inconsistencias, e incluso daños que a su vez genera demora en la entrega del producto al cliente.

Figura 14. Diagrama de la comunicación entre las áreas en el proceso anterior



7.3 Duplicidad y redundancia de datos

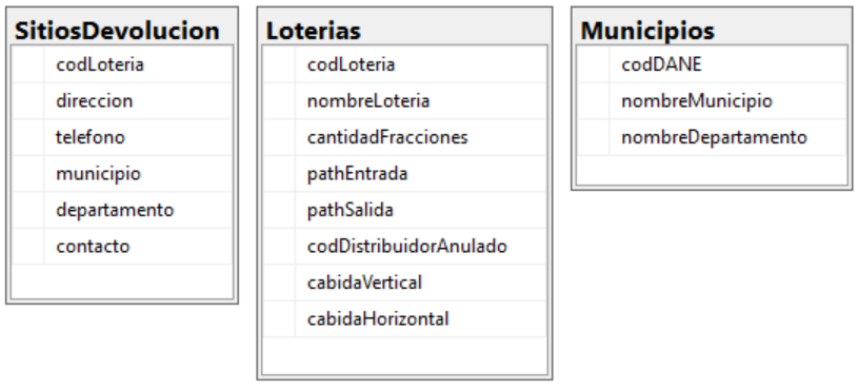
Para el procesamiento de los diferentes productos de la compañía por la anterior aplicación se requería de tres bases de datos que tenían mucha información duplicada, las bases de datos que antes se tenían eran:

- SISLOT

Figura 15. Tablas utilizadas por la aplicación SISLOT

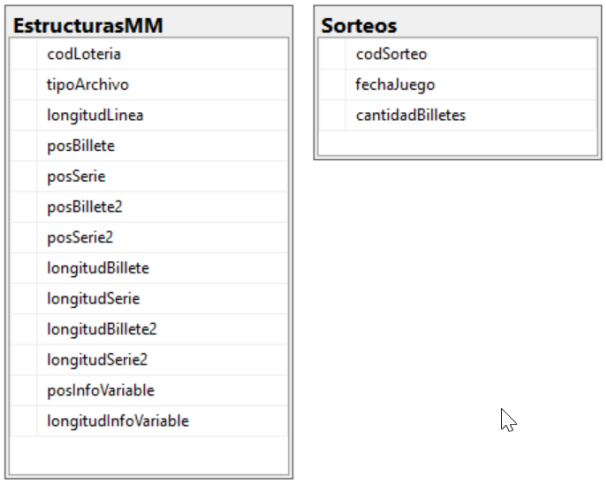
- UTILIDADES

Figura 16. Tablas utilizadas en la base de datos UTILIDADES



- LOTERIAS

Figura 17. Tablas utilizadas en la base de datos LOTERIAS



Como podrán observar, las tablas no se encontraban relacionadas y no tenían un identificador como primary key.

Muchos de los datos de las aplicaciones no estaban parametrizadas en la base de datos, estos están “quemados” en el código de la aplicación o en los controles visuales de las aplicaciones de escritorio.

7.4 Despliegue de versiones

Otro de los problemas detectados es que debido a que las aplicaciones que permiten el procesamiento de los productos están desarrolladas en Windows Forms (aplicaciones stand-alone o de escritorio), cada que se necesitaba realizar el despliegue a producción de alguna nueva versión de la aplicación, un analista del área de TI tenía que ir a cada uno de los equipos de cómputo en donde estuviera instalado la aplicación en la planta de producción, y realizar la actualización de la aplicación uno a uno. En ocasiones se presentaban problemas o llamadas de soporte en horario hábil y también en horario no hábil, debido a que el analista que realizó dicha labor de actualización, se le olvidó instalar la nueva versión en alguno de los equipos de cómputo.

7.5 Cargue de insumos manuales

Las aplicaciones que realizaban el procesamiento de los productos, deben cargar una serie de archivos que son insumos primordiales para generar y obtener cierta información que es necesaria para la generación de archivos de impresión. Aunque son necesarios, estos procesos pueden ser mejorados generando la información automáticamente desde la aplicación, sin necesidad de obtener dichos datos cargando archivos, evitando errores de cargues por archivos incorrectos, mucha operación manual, entre otros.

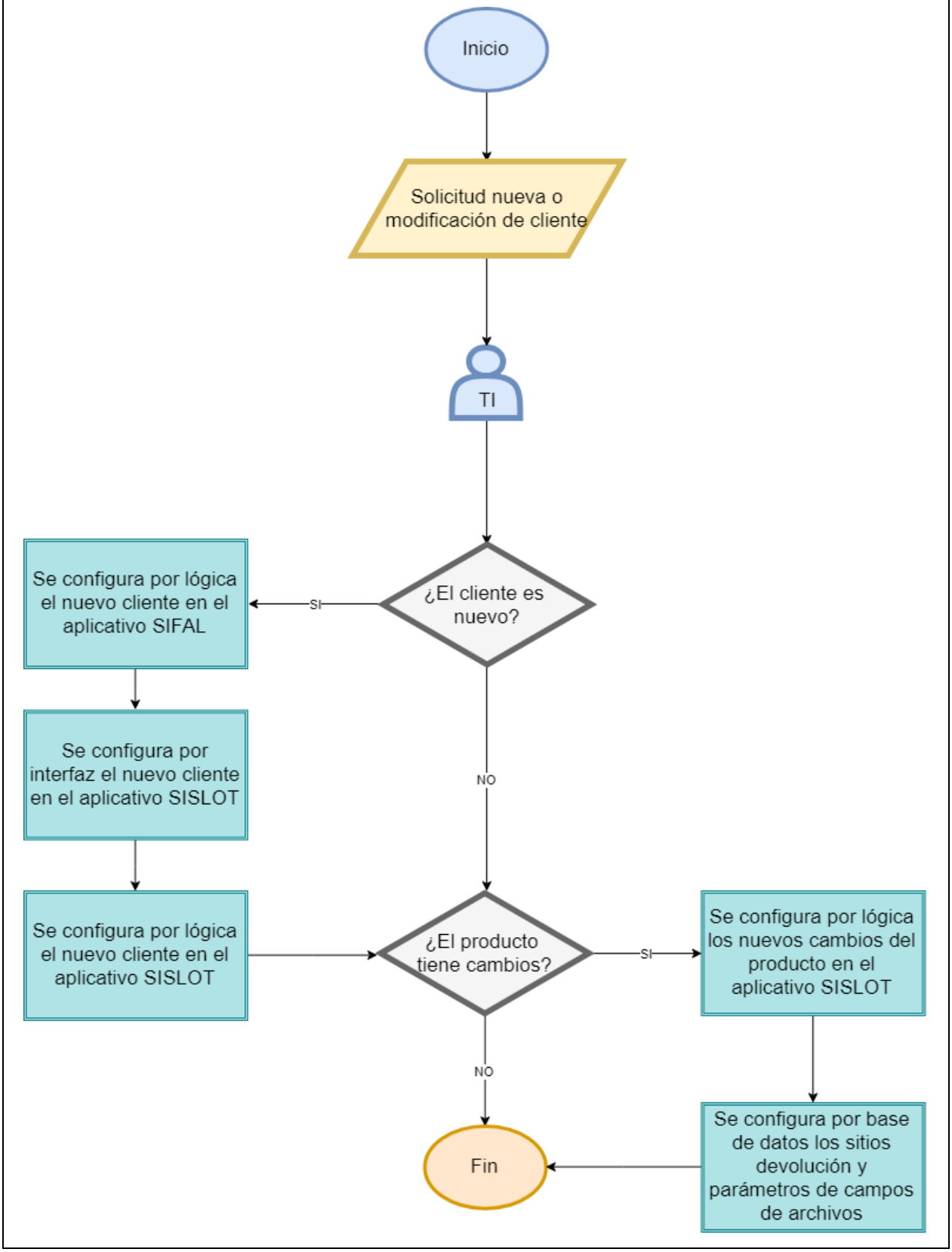
7.6 Proceso anterior en la creación y/o modificación de un cliente

Cada que se solicita la creación y/o modificación de un cliente (lotería), el área de TI debe gastar mucho tiempo para realizar dicha labor, debido a que se deben realizar muchos ajustes en las aplicaciones existentes y en las bases de datos. Esto generalmente ha incurrido a demoras en la entrega de los productos a producción, al igual que de errores en el proceso ya que debido a la complejidad y a la deuda técnica que tienen las aplicaciones, agregar alguna funcionalidad nueva o modificar una existente ocasiona que se altere las funcionalidades de otros clientes a los cuales no hay que modificarle nada, o en ocasiones no se realizan todas las modificaciones necesarias, se pasa algún componente o configuración por alto, lo que genera una salida de insumos no adecuados a lo que se esperaba.

Si bien por base de datos están algunos de los datos empleados en el proceso por cada cliente, la deuda técnica de las aplicaciones es alta, debido a que muchas configuraciones y datos requeridos en el procesamiento están “quemados” en el código fuente de las aplicaciones, y estas no son tan extensibles a nuevas funcionalidades, no cuenta con una arquitectura definida, no cuenta con buenas prácticas de programación y código limpio.

A continuación, se muestra el diagrama general de cómo era el proceso cuando se iba a crear y/o modificar un cliente.

Figura 18. Proceso anterior en la creación/modificación de un cliente



8 Discusión

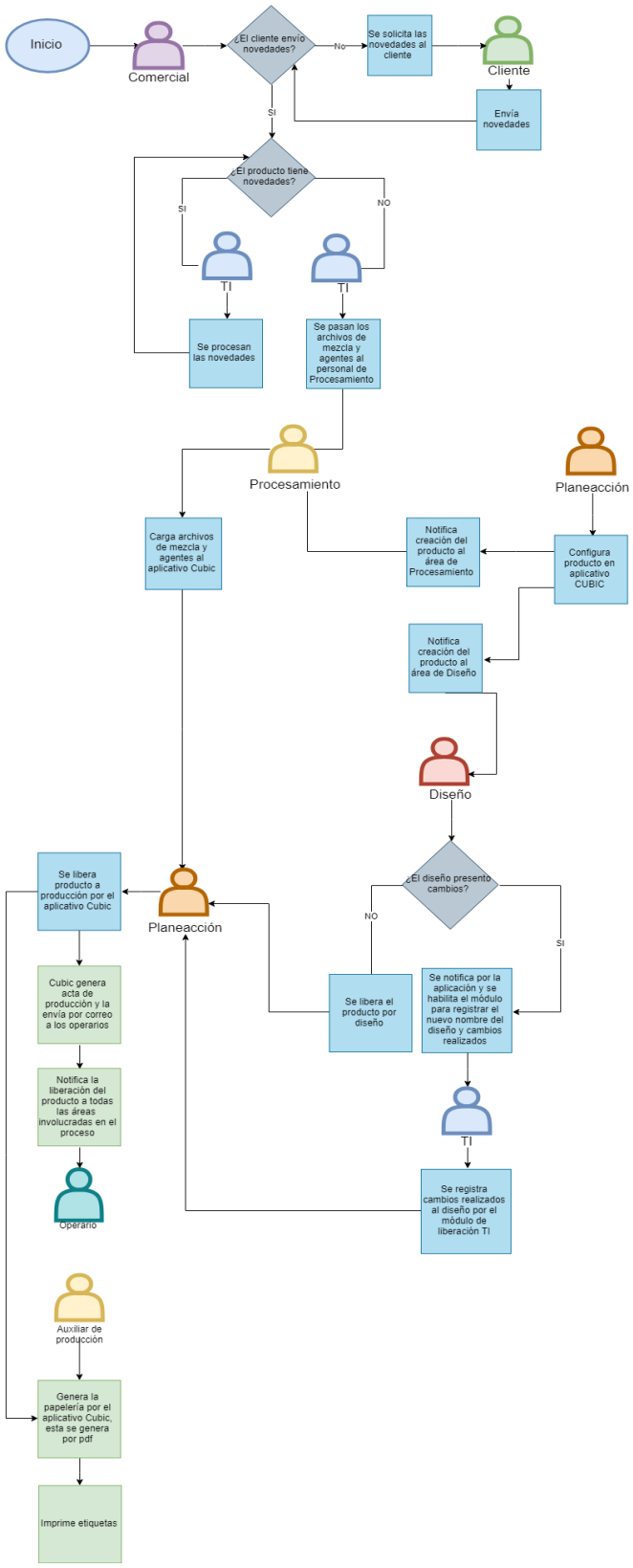
A raíz de los resultados obtenidos en el análisis del proceso como funcionaba antes, se evidenció muchas mejoras a nivel de operación y sistema, los cuales se explicarán a continuación.

8.1 Flujo propuesto para el proceso

Con la realización de la solución tecnológica se logró plantear un flujo con el cual cada área que involucra la realización de los productos sea la responsable de configurar los parámetros que les corresponden para que la aplicación pueda procesar y liberar el producto a producción.

Lo que se hace básicamente, es controlar todo el proceso a través de estados, en donde los usuarios podrán conocer el estado que tiene el producto, saber qué insumos están pendientes por realizar para tomar las medidas correspondientes para darle continuidad o averiguar porqué del estado de ese producto, saber cuáles productos ya están listos para pasar a producción, cuales ya se liberaron y comunicar todas las áreas para llegar a un objetivo en común que es la correcta realización de los productos en la planta de producción.

Figura 19. Flujo planteado para el procesamiento de los productos

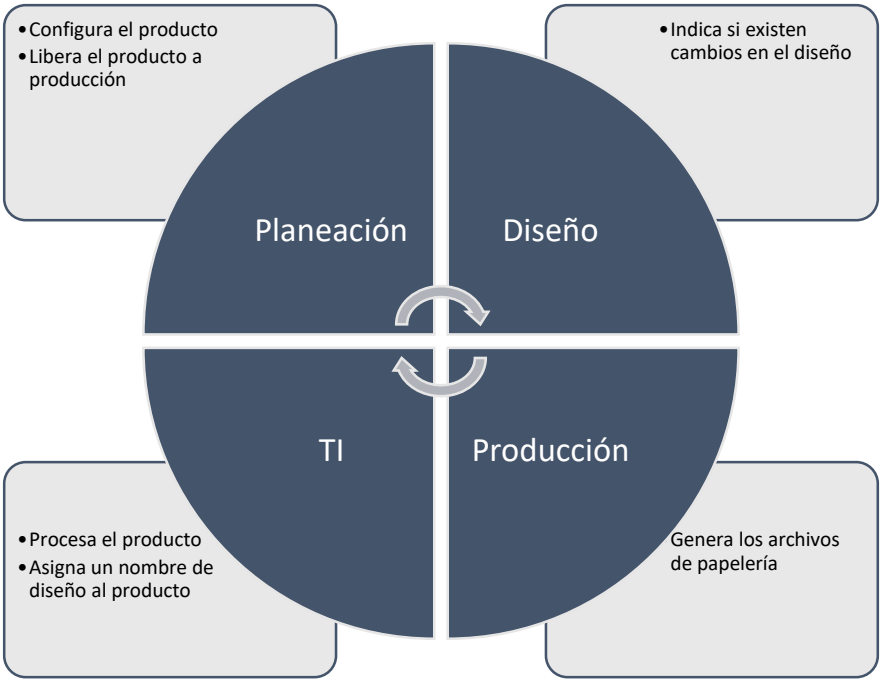


Para ampliar sobre el flujo propuesto para el proceso, ver Anexo 2, en donde se refleja en detalle de qué manera interactúan las diferentes áreas con la implementación del proyecto de desarrollo propuesto.

8.2 Comunicación entre las áreas en el proceso implementado

Para solucionar uno de los inconvenientes en la mala realización de los productos que se liberan a producción, el flujo del proceso se cambió de manera notoria con el fin de comunicar asertivamente a las diferentes áreas que se involucran con el proceso, para de esta forma garantizar que toda la información requerida se obtenga oportunamente y que sea confiable, ya que se delegan responsabilidades que cada área debe asumir, y si llega a existir algún cambio durante el proceso, todas las demás áreas se puedan enterar de dicho cambio con el fin de tomar medidas al respecto e evitar entregas a producción que no están alineadas con dichos cambios.

Figura 20. Diagrama de la comunicación entre las áreas en el proceso implementado



8.3 Unificación y homologación de datos

Se realiza el análisis de la duplicidad y redundancia de datos en las bases de datos, y se logra consolidar todas estas bases de datos en una sola base de datos normalizada, quedando así:

Figura 21. Tablas para la gestión de usuarios y permisos

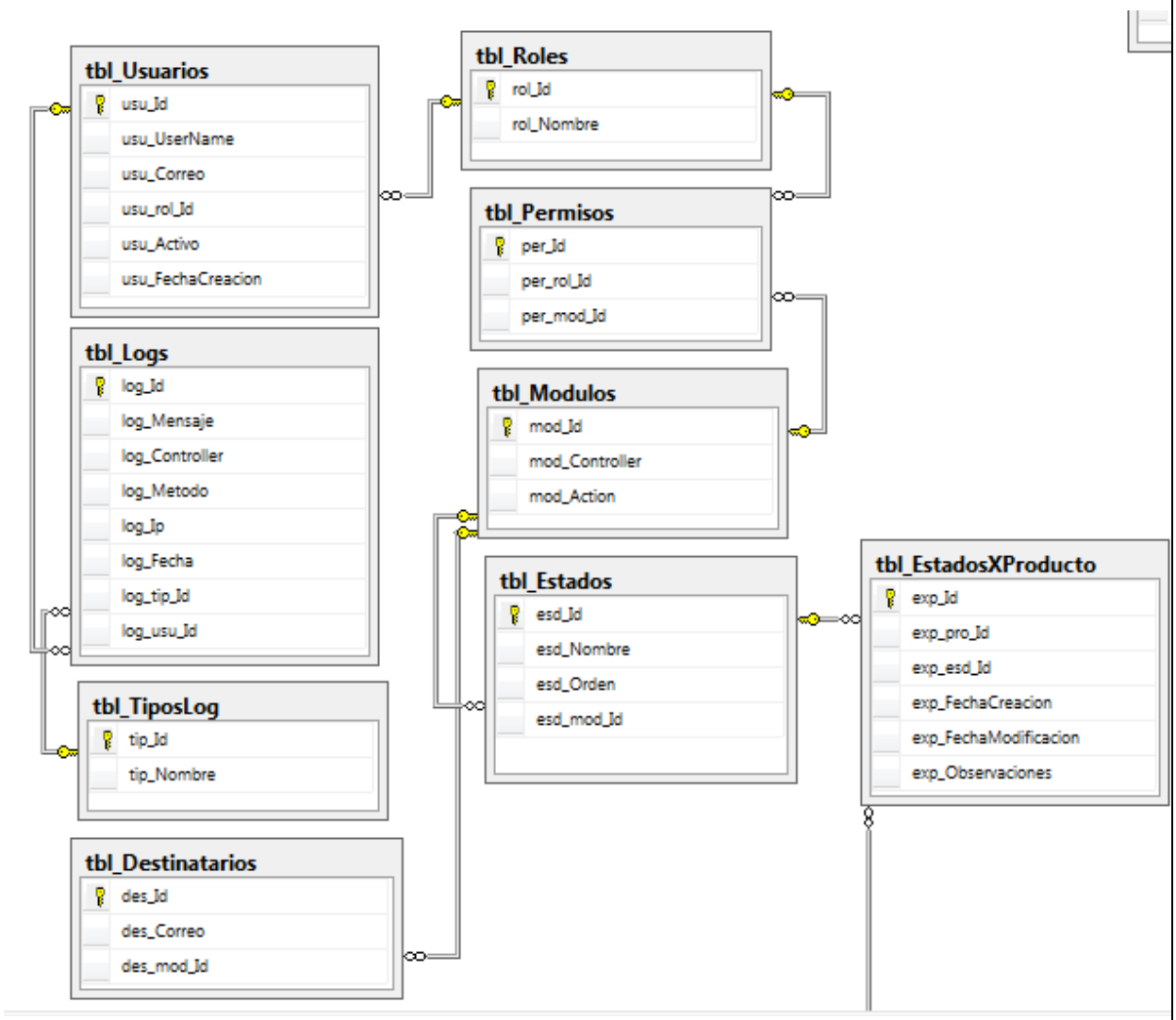


Figura 22. Tablas para la gestión de las mezclas y el control de versiones de archivos

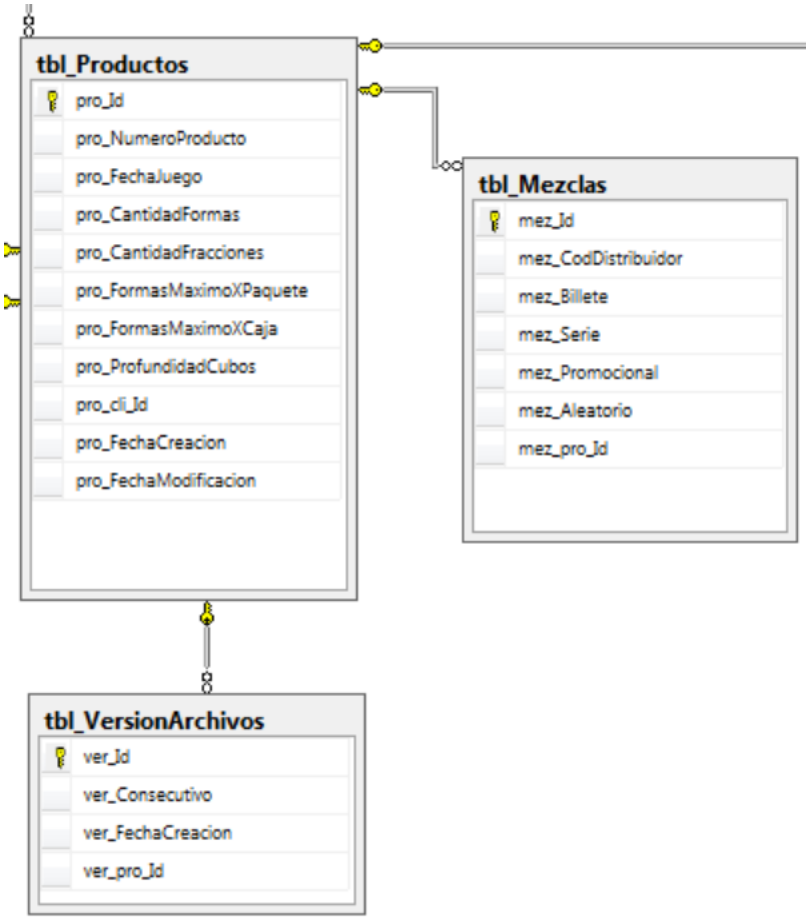


Figura 23. Tablas para la gestión de clientes y datos de distribuidores

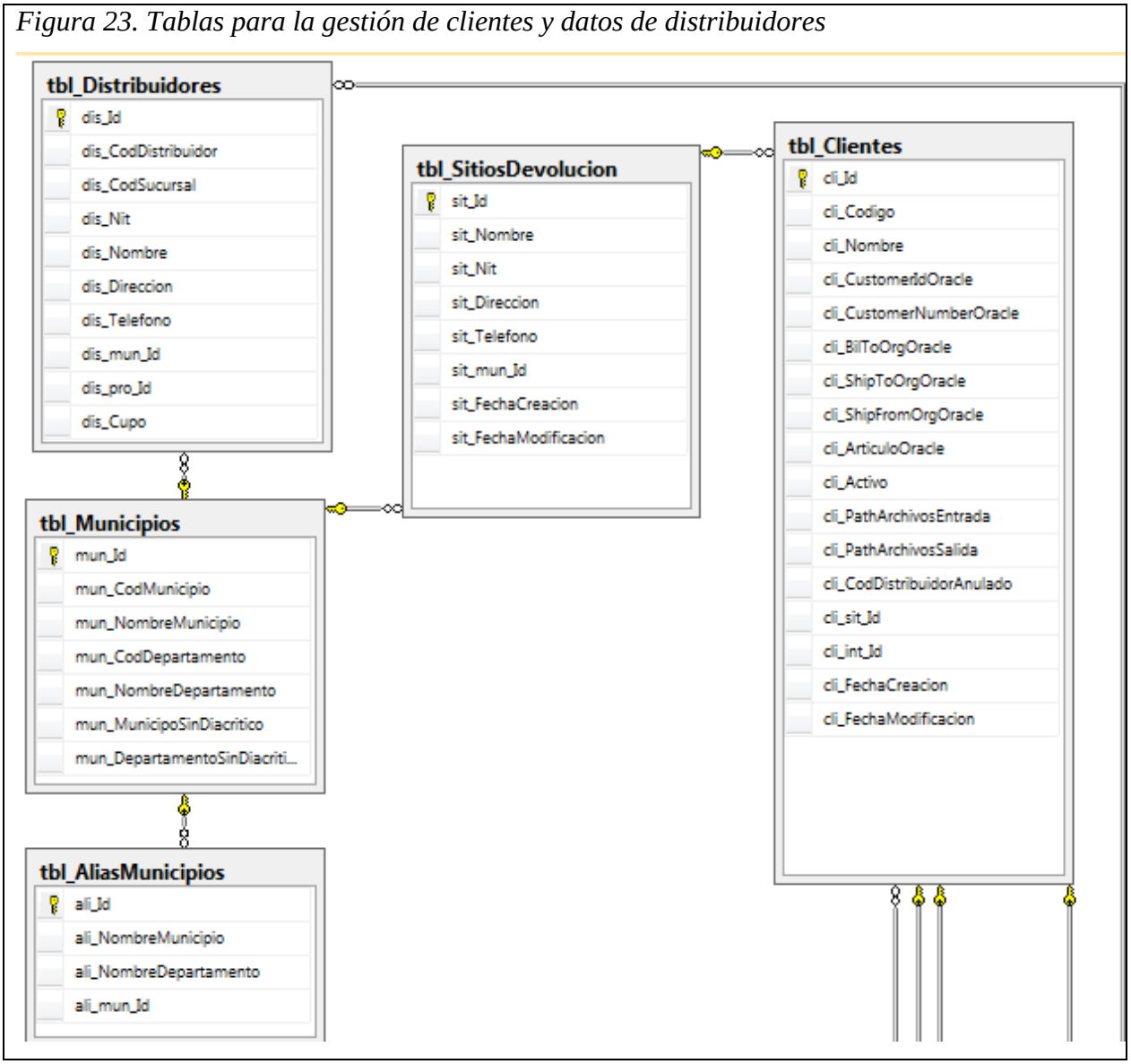


Figura 24. Tablas para la gestión de datos para producción

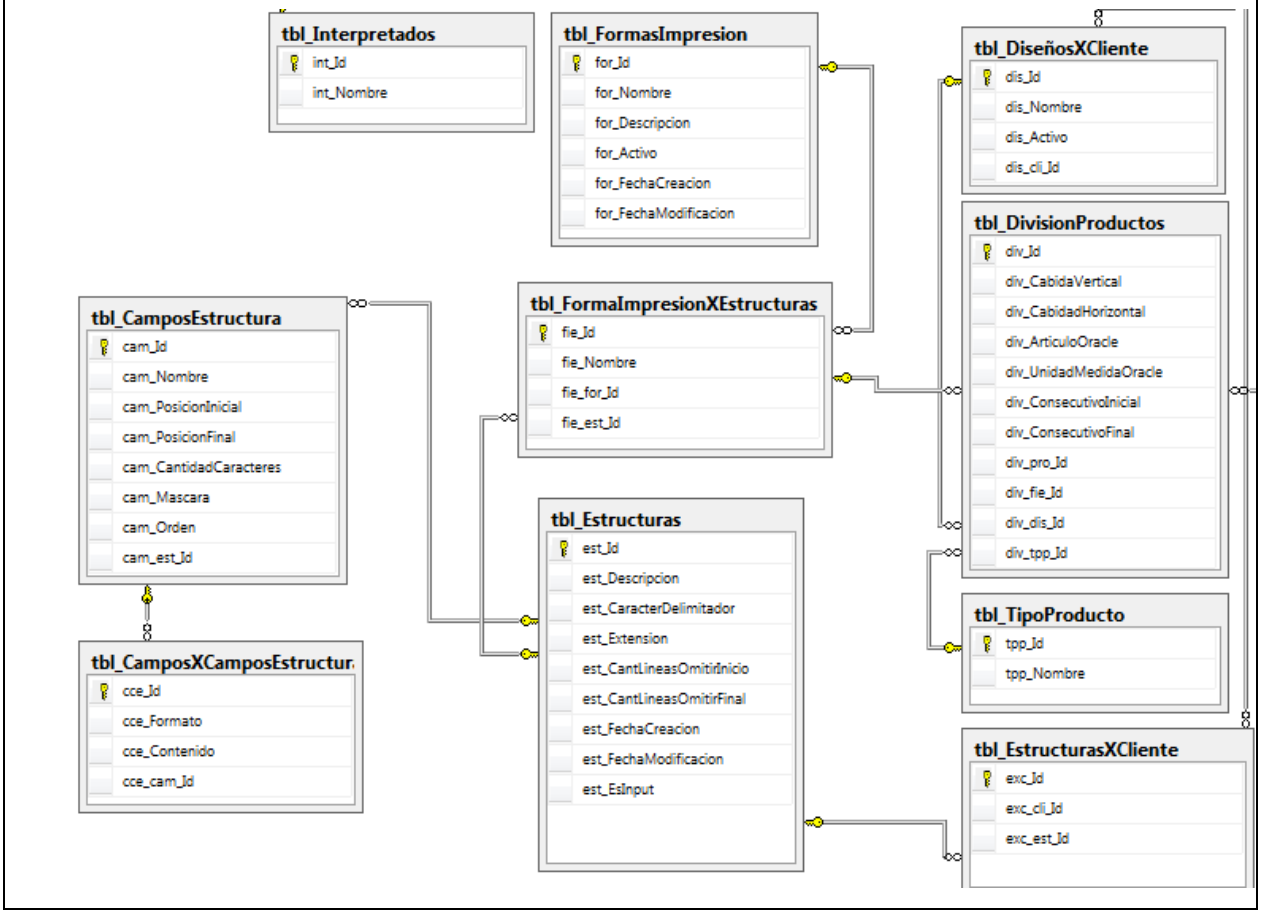
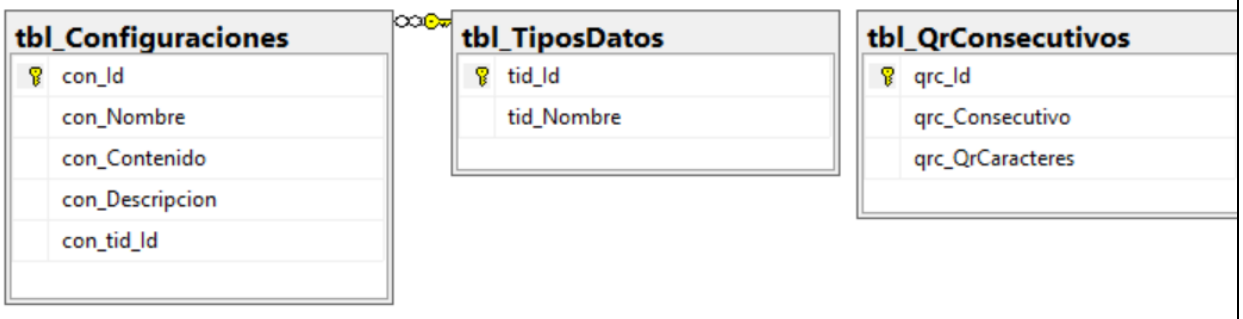


Figura 25. Tablas de configuraciones



8.4 Despliegues más limpios

La solución se hizo en una aplicación web, cuyo despliegue a producción es mucho más limpio ya que la publicación de nuevos releases se realiza en un solo lugar, en el servidor de

aplicaciones, donde se está ejecutando el IIS. Con esto, las nuevas versiones estarán disponibles para todos los usuarios y no se correrá el riesgo de que alguno de los equipos quede desactualizado con versiones anteriores del software.

Sin embargo, dependiendo del tipo de release que se vayan a ejecutar, es importante que se esté en constante comunicación con los usuarios y se le haga acompañamiento para garantizar que todo salga según lo esperado.

8.5 Mejoras a nivel de aplicación y proceso

Se lograron eliminar los siguientes procesos manuales al momento de procesar la información para liberar los productos a producción:

- Realización de archivo de cubos para organizar la caída de la información al momento de la impresión, anteriormente se realizaba este archivo con ayudad de la herramienta Excel, mediante cálculos realizados con paramentos suministrados por el área de planeación.
- Cargue de archivo de números de 7 dígitos aleatorios para el cálculo de los dígitos de control y tiras virtuales que se le asigna a cada fracción de cada uno de los billetes de producidos, estos números tienen que ser de 7 dígitos.
- Configuración en la lógica de la aplicación al momento que la estructura de archivo de impresión tuviera un cambio en posiciones o aumento de campos, este cambio se realizaba para generar el listado de máquina.
- Eliminación de aplicativo (SIFAL) quien era el encargado de realizar el formateo de los diferentes archivos de mezclas y agentes que enviaban las loterías para que se diera el formato de los archivos de entrada que recibía la aplicación (SISLOT), el aplicativo actual se configura la estructura de entrada del archivo enviado por el cliente, teniendo como parámetros obligatorios el número de billete, serie y código del distribuidor.

8.6 Proceso implementado en la creación y/o modificación de un cliente

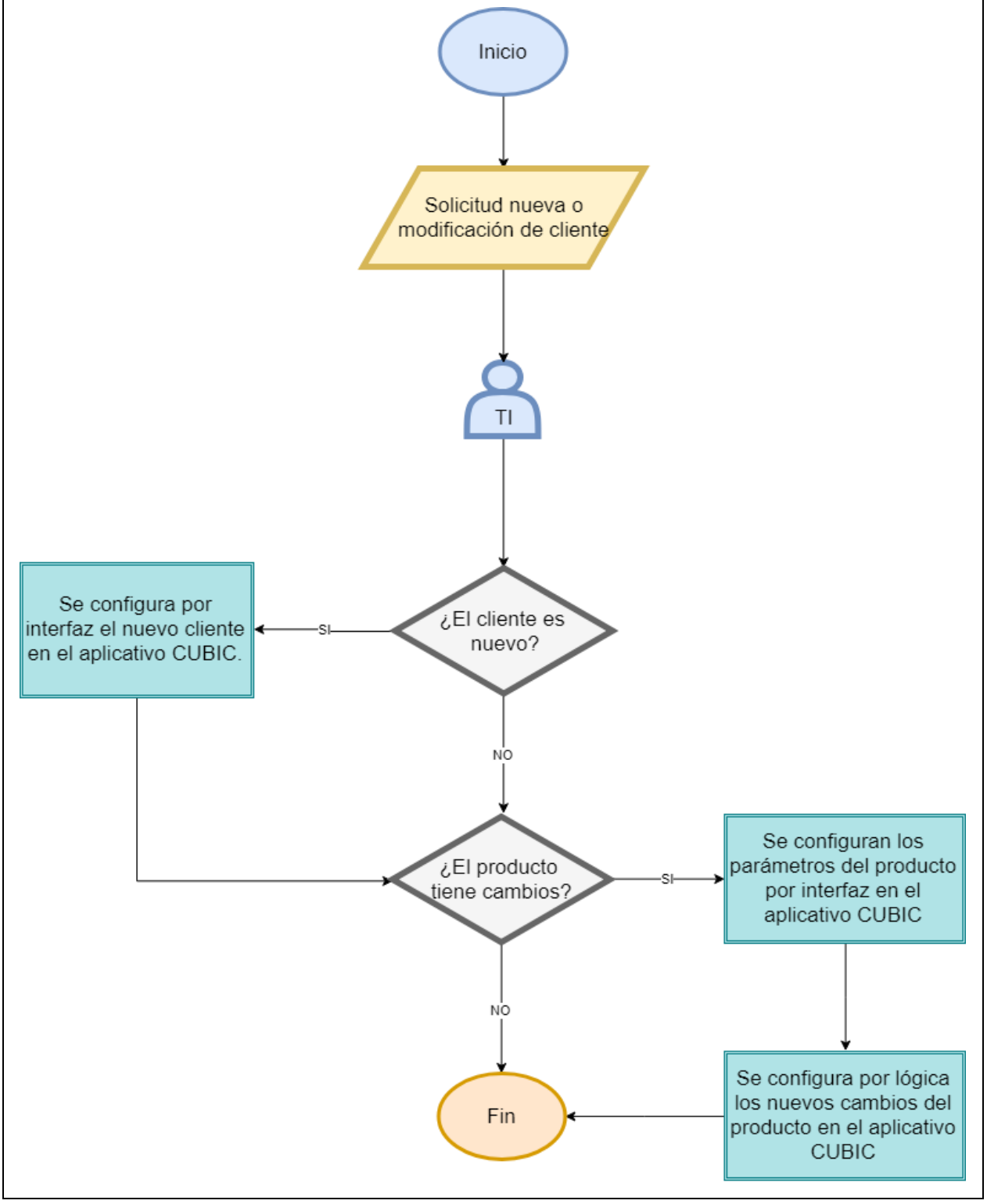
Con el fin de solucionar las fallas y complicaciones que se tenía al momento de crear y/o modificar un cliente o sus funcionalidades, se realizó la aplicación web de tal forma que la mayoría

de configuraciones que tiene el cliente se puedan parametrizar desde la base de datos por medio de la aplicación web.

También, se desarrolló la aplicación con buenas prácticas y con una arquitectura que permite que la lógica que tenga cada cliente en el código fuente no interfiera con la de otros clientes, facilitando las modificaciones que se realicen sobre el código fuente y evitando posibles fallas o inconsistencias en los productos e insumos generados por la aplicación.

Hablando ya de la interacción de los usuarios con la aplicación, se logró implementar un proceso más limpio y sin tantas manualidades y pasos, permitiendo facilidad en su uso y confiabilidad ya que las probabilidades de fallos por errores humanos disminuyeron notoriamente.

Figura 26. Proceso implementado en la creación/modificación de un cliente



9 Conclusiones

Con este proyecto se desarrolló una solución web que nació de la necesidad del área de TI producción Cali de la compañía Cadena (Dvalor S.A.S.) de tener una herramienta configurable y adaptable para el procesamiento de los diferentes productos de la UEN de protección contra el fraude, optimizando tiempos en el procesamiento, delegándole funcionalidades y responsabilidades al sistema que anteriormente se realizaban de forma manual.

Gracias al análisis, no solo a nivel de sistemas sino también a nivel de procesos, se vio la viabilidad de generar una mejora bastante robusta para suplir con las falencias que se estaban presentando con el proceso anterior.

Dicho análisis llevó a desarrollar una aplicación en donde se cumplen las necesidades principales que requiere el producto para llegar a la planta de producción en las mejores condiciones en cuanto a confiabilidad y seguridad de la información.

Generar mecanismos sistematizados para comunicar las áreas involucradas e informar del estado de los productos en producción, fue vital para mantener a las áreas informadas con el fin de que estas tomen de manera oportuna decisiones que ayuden a darle continuidad a los productos y no afectar los tiempos de producción, además de que ayudó a que estas estuvieran informadas ante alguna eventualidad o cambio realizado en los productos.

Además del análisis a nivel de procesos, detectar la mejor manera de realizar las soluciones informáticas ayudan a cumplir con los objetivos, pues facilitar la forma como se interactúa con la aplicación y con la manera en que se crea y se accede a los datos gestionados por la aplicación.

En resumen, se concluye que, con un buen análisis y un estudio de campo, en donde se tenga en cuenta no solo los procesos a nivel de sistemas sino también a nivel de escenarios y procesos, se puede encontrar la mejor forma de realizar las soluciones propuestas, con el fin de satisfacer las necesidades de los clientes y suplir con sus necesidades reales.

10 Recomendaciones

Las recomendaciones que se recolectaron durante la realización del aplicativo son las siguientes:

- Sobre el análisis, diseño y desarrollo
  - Es importante tener una comunicación fluida y constante con las personas que van a interactuar con la aplicación al momento de levantar los requisitos, ya que aportan unas características diferentes del producto que podrían ayudar a comprender mejor el problema.
  - Vale la pena dedicar el mayor tiempo posible a la búsqueda de patrones de diseño y herramientas de desarrollo que faciliten la construcción de la aplicación y si en un futuro se deben realizar ajustes a la base de datos o cambios a nivel de interfaz no sea traumático realizarlas.
  - Se recomienda durante la construcción del proyecto comentar las líneas de código más relevantes y de una manera adecuada, ya que no se sabe si volveremos a tocar dicho código u otra persona lo hará, esto es muy importante para efectos de compresión y mantenimiento.
  - Es muy importante implementar código genérico, recursivo y estandarizado, para efectos de rendimiento de la aplicación.

Referencias

Alaimo, D. M. (2013). *Proyectos Ágiles con Scrum.* Obtenido de Kleer: http://www.elkinforero.com/joomdocs/kleer-proyecto-agiles-con-scrum.pdf

Alarcón, J. (2018). *Entity Framework: Code First, Database First y Model First ¿En qué consiste cada uno?* Obtenido de Campus MVP: https://www.campusmvp.es/recursos/post/entity-framework-code-first-database-first-y-model-first-en-que-consiste-cada-uno.aspx

Álvarez, C. (2014). *Patrones de Diseño (Active Record vs DAO).* Obtenido de genbeta: https://www.genbeta.com/desarrollo/patrones-de-diseno-active-record-vs-dao

Alvarez, M. (2014). *Qué es MVC.* Obtenido de desarrolloweb: https://desarrolloweb.com/articulos/que-es-mvc.html

Apiumhub. (2017). *Beneficions de las pruebas unitarias*. Obtenido de https://apiumhub.com/es/tech-blog-barcelona/beneficios-de-las-pruebas-unitarias/

Beck, K., Beedle, M., Bennekum, A. v., Cockburn, A., Cunningham, W., Fowler, M., . . . Sutherland, J. (2001). *Principios del Manifiesto Ágil*. Obtenido de agilemanifesto.org: http://agilemanifesto.org/iso/es/principles.html

Beck, K., Beedle, M., Bennekum, A. v., Cockburn, A., Cunningham, W., Fowler, M., . . . Thomas, D. (2001). *Manifiesto por el Desarrollo Ágil de Software*. Obtenido de agilemanifesto.org: http://agilemanifesto.org/iso/es/manifesto.html

Bioul, G., Escobar, F., Alvarez, M., Nardin, A., & Ricci, E. (2010). *Metodologías Ágiles, análisis de su implementación y nuevas propuestas.* Obtenido de SEDICI: Repositorio Institucional de la UNLP: http://sedici.unlp.edu.ar/bitstream/handle/10915/19292/Documento_completo.pdf?sequence=1

Cadelli, S., & Fernández, J. M. (2015). *Convivencia de metodologías: Scrum y RUP en un proyecto de gran escala.* Obtenido de SEDICI: Repositorio Institucional de la UNLP: http://hdl.handle.net/10915/47082

Cai, S. (2015). *Introduction to the C# Language and the .NET Framework*. Obtenido de Microsoft: https://docs.microsoft.com/es-es/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework

Cai, S. (2017). *ADO.NET Overview*. Obtenido de Microsoft: https://docs.microsoft.com/es-es/dotnet/framework/data/adonet/ado-net-overview

Castillo, A., Barrios, J., Montilva, J., & Rivero, D. (2010). Conceptualización del proceso de implementación de software: perspectivas ágil y disciplinada. *Ciencia e Ingeniería, 31*(3). Obtenido de http://erevistas.saber.ula.ve/index.php/cienciaeingenieria/article/view/1147

Catalá, E., & Puig, E. (2017). *Buenas prácticas de codificación para capas de acceso a datos de aplicaciones (III)*. Obtenido de Microsoft: https://msdn.microsoft.com/es-es/communitydocs/net-dev/dev/buenas-practicas-parte3

Cuéllar, J. (2010). *Estilos y patrones básicos en arquitectura de software*. Obtenido de http://josecuellar.net/estilos-patrones-basicos-arquitectura-software/

Dirección general. (2018). *La metodología Scrum, el último grito en gestión de proyectos*. Obtenido de Escuela Europea de Management: http://www.escuelamanagement.eu/direccion-general-2/la-metodologia-scrum-ultimo-grito-gestion-proyectos

E.V.A. UCI, I. D. (s.f.). *Proceso unificado de desarrollo*. Obtenido de EcuRed: https://www.ecured.cu/Proceso_unificado_de_desarrollo

etsii. (s.f.). *Tema 1: Patrones Arquitectónicos*. Obtenido de Escuela Técnica Superior de Ingeniería Informática: http://www.lsi.us.es/docencia/get.php?id=1891

Expósito, C., Expósito, A., López, I., Melián, M., & Moreno, J. (s.f.). *Introducción a UML*. Obtenido de Universidad de La Laguna: https://campusvirtual.ull.es/ocw/course/view.php?id=132

Florez, H., & Ardila, D. (2014). *Diseño e Implementación de un sistema de información con Ingeniería Web para la administración de proyectos de investigación.* Obtenido de Biblioteca Digital USB: http://bibliotecadigital.usb.edu.co/bitstream/10819/3973/1/Diseno_Implementacion_Sistema_Florez_2014.pdf

Galván, P. (2018). Integración Continua. *SG*(54). Obtenido de https://sg.com.mx/revista/54/integraci-n-continua

Gil, C. (2008). RUP: Metodología en los sistemas y aplicaciones basadas en la web. *Revista Avances*. Obtenido de http://www.unilibre.edu.co/revistaavances/avances-8/r8_art9.pdf

Khan, A. (2012). *Transact-SQL : Introduction and Overview*. Obtenido de The Windows Club: https://www.thewindowsclub.com/transact-sql-introduction

Latham, L., Wenzel, M., Wagner, B., tompratt, & Jones, M. (2017). *Overview of the .NET Framework.* Obtenido de Microsoft: https://docs.microsoft.com/en-us/dotnet/framework/get-started/overview

Lerman, J. (2011). *Data Points - Demystifying Entity Framework Strategies: Loading Related Data*. Obtenido de Microsoft: https://msdn.microsoft.com/magazine/hh205756.aspx

Mersino, A. (2018). *Agile Projects are More Successful than Traditional Projects*. Obtenido de Vitality Chicago: https://vitalitychicago.com/blog/agile-projects-are-more-successful-traditional-projects/

Microsoft. (2013). *ASP.NET MVC Overview*. Obtenido de Microsoft: https://docs.microsoft.com/en-us/previous-versions/aspnet/web-frameworks/dd381412(v=vs.108)

Microsoft. (2015). *Conceptos básicos de prueba unitaria*. Obtenido de Microsoft: https://msdn.microsoft.com/es-co/library/hh694602.aspx

Microsoft. (2016). *Entity Framework Code First Migrations with an existing database*. Obtenido de https://msdn.microsoft.com/en-us/library/dn579398(v=vs.113).aspx

Microsoft. (s.f.). *ASP.NET Dynamic Data Scaffolding*. Obtenido de https://msdn.microsoft.com/en-us/us-us/library/ee377606(v=vs.100).aspx

Microsoft. (s.f.). *Herramientas basadas en la nube para los equipos de desarrollo de software*. Obtenido de Microsoft: https://www.microsoft.com/es-xl/trustcenter/cloudservices/vsts

Microsoft. (s.f.). *Team Foundation Server*. Obtenido de Microsoft: https://visualstudio.microsoft.com/es/tfs/

Microsoft. (s.f.). *TÉRMINOS DE LA LICENCIA DE SOFTWARE DE MICROSOFT*. Obtenido de https://visualstudio.microsoft.com/es/license-terms/mlt553321/

Microsoft. (s.f.). *What is .NET?* Obtenido de https://www.microsoft.com/net/learn/what-is-dotnet

Penadés, M. C., & Letelier Torres, P. O. (2006). *Métodologías ágiles para el desarrollo de software*. Obtenido de Universidad de La Rioja: https://dialnet.unirioja.es/servlet/articulo?codigo=1983605

Péraire, C., Edwards, M., Fernandes, A., Mancin, E., & Carroll, K. (07 de 2007). *The IBM Rational Unified Process for System z.* Obtenido de ibm.com/redbooks: http://www.redbooks.ibm.com/redbooks/pdfs/sg247362.pdf

proyectosagiles.org. (s.f.). *Qué es SCRUM*. Obtenido de proyectosagiles.org: https://proyectosagiles.org/que-es-scrum/

Ruiz Pacheco, J. C. (2017). *Patrones de Diseño - Estrategia*. Obtenido de Microsoft: https://msdn.microsoft.com/es-es/communitydocs/net-dev/csharp/patrones-de-diseno

Ruiz, J. (2017). *C# - Inyección de Dependencias*. Obtenido de Microsoft: https://msdn.microsoft.com/es-es/communitydocs/net-dev/csharp/inyeccion-de-dependencias

Schwaber, K., & Sutherland, J. (2013). *The Scrum Guide.* Obtenido de Scrum Guides: http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-US.pdf#zoom=100

Soji, M. (2014). *Basic Introduction to Data Annotation in .NET Framework*. Obtenido de Code Project: https://www.codeproject.com/Articles/826304/Basic-Introduction-to-Data-Annotation-in-NET-Frame

Sutherland, J. (2013). *Agile Principles and Values, by Jeff Sutherland*. Obtenido de Microsoft: https://msdn.microsoft.com/es-es/library/dd997578(v=vs.120).aspx

Tedeschi, N. (s.f.). *¿Qué es un Patrón de Diseño?* Obtenido de Microsoft: https://msdn.microsoft.com/es-es/library/bb972240.aspx

tutorialspoint. (s.f.). *ASP.NET MVC - Data Annotations*. Obtenido de Tutorials Point: https://www.tutorialspoint.com/asp.net_mvc/asp.net_mvc_data_annotations.htm

Universidad de Alicante. (s.f.). *Modelo vista controlador (MVC)*. Obtenido de Universidad de Alicante: https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html

Villanueva, J. S., & Siachoque, M. M. (2014). SCRUM y RUP: Comparativa y propuesta metodológica. *T.I.A, 2*(1). Obtenido de http://revistas.udistrital.edu.co/ojs/index.php/tia/article/view/5697

w3schools. (s.f.). *SQL Stored Procedures for SQL Server*. Obtenido de w3schools: https://www.w3schools.com/sql/sql_stored_procedures.asp

Wasson, M. (2017). *N-tier architecture style*. Obtenido de Microsoft: https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/n-tier

Anexos

Anexo 1. Diagrama de flujo de producción Loterías – Actual

En el anexo **Diagrama de flujo de producción Loterías – Actual.pdf** se explica de qué forma se realiza actualmente el proceso de producción dentro de la compañía Dvalor S.A.S. con respecto a los productos de loterías.

En este flujo se puede apreciar que la mayor carga de procesos se encuentra dentro del área de tecnología, y que las demás áreas, aunque son importantes para el proceso no se involucran formalmente, ocasionando que exista una comunicación deficiente entre las áreas y evitando detectar inconsistencias e incidentes de manera temprana durante el proceso.

Anexo 2. Diagrama de flujo de producción Loterías - A Implementar

En el anexo **Diagrama de flujo de producción Loterías - A Implementar.pdf** se explica de qué forma se va a realizar el proceso de producción dentro de la compañía Dvalor S.A.S. con respecto a los productos de loterías con la implementación del proyecto desarrollado.

En este flujo se puede apreciar una mejora en la distribución de los procesos entre las diferentes áreas involucradas, garantizando una mayor comunicación, control y seguimiento a los productos de loterías.

Anexo 3. Documento de Arquitectura de Software

En el anexo **Plantilla_DAS_Arquitectura-Light_CUBIC.pdf** se encuentra toda la información relacionada a las especificaciones arquitectónicas de la aplicación y la distribución de los componentes físicos y lógicos para la implementación de la aplicación.

Anexo 4. Casos de Uso

En el anexo **Casos de Uso Extendido_CUBIC.xlsx** se encuentran los diagramas de casos de uso y los casos de uso extendidos los cuales retroalimentan las historias de usuario para conocer a detalle los requisitos funcionales del sistema.

Anexo 5. Historias de Usuario

En el anexo **[HU-CUBIC] - Con Estimación.xlsx** se encuentran las historias de usuario identificadas y desarrolladas en la aplicación. Estas historias de usuario ya se encuentran refinadas y estimadas, en donde además se relacionan la cantidad de sprints a desarrollar y el tiempo invertido en cada sprint para entregar el mínimo producto viable.

Anexo 6. Técnicas Ágiles - Elevator Pitch

Dentro de la carpeta "**Técnicas Ágiles\Elevator Pitch"** de los anexos, se encuentran varias fotos en donde se evidencia la realización del Elevator Pitch, el cual consistió en presentar a los altos cargos del área de tecnología de la empresa Dvalor S.A.S. una propuesta como solución y mejora al proceso de loterías que actualmente tiene la empresa, con el objetivo de generar intereses y poder lograr así una reunión en donde se le explicó en detalle en qué consistía la propuesta.

Anexo 7. Técnicas Ágiles - ¿Qué nos permite dormir en las noches y qué no?

Dentro de la carpeta "**Técnicas Ágiles\Que nos permite dormir y que no"** de los anexos, se encuentran varias fotos en donde se evidencia la realización de una técnica que consiste en identificar aquellos temas o fortalezas que nos puede ayudar a llevar a cabo el desarrollo de la aplicación, y también identificar aquellas oportunidades de mejora que se deben solventar para lograr con éxito la realización del proyecto.

Anexo 8. Técnicas Ágiles - Walking Skeleton

Dentro de la carpeta "**Técnicas Ágiles\Walking Skeleton"** de los anexos, se encuentran varias fotos en donde se evidencia la realización del Walking Skeleton, en donde se plasma el

comportamiento que debería tener la aplicación de acuerdo al alcance que va a tener, es una visión general de la aplicación para dar un indicio y recibir retroalimentación de los usuarios.

Anexo 9. Sprint Review

Dentro de la carpeta "**Técnicas Ágiles\Evidencias Sprint Review"** de los anexos, se encuentra las evidencias de las reuniones realizadas con el Product Owner de la empresa Dvalor S.A.S. en donde al finalizar cada Sprint se realizó su respectiva entrega, donde además se evaluaba el alcance realizado, se analizaban posibles cambios en el alcance o se daba por aprobado cada una de las historias de usuario.

Anexo 10. Sprint Retrospectiva

Dentro de la carpeta "**Técnicas Ágiles\Evidencias Sprint Retrospectiva"** de los anexos, se encuentran las evidencias de las retrospectivas realizadas entre el equipo Scrum luego de cada Sprint Review.

Anexo 11. Escenarios de pruebas

Dentro de la carpeta Anexos, se encuentra un archivo llamado **Escenarios de pruebas-CUBIC.xlsx** el cual contiene las evidencias de los escenarios de pruebas realizadas a la aplicación con los procesos principales que realiza la aplicación web con el fin de verificar el cumplimiento de varios de los objetivos planteados.