

# FACULTAD DE INGENIERÍA



Carrera de Ingeniería de Sistemas Computacionales

“GENERADOR DE CÓDIGO DE PROGRAMACIÓN EN LA ETAPA DE IMPLEMENTACIÓN DE UN SOFTWARE DE LA EMPRESA INKA MALL S.A.C., 2019”

Tesis para optar el título profesional de:

Ingeniero de Sistemas Computacionales

Autores:

Nicky Alejandro Cotrina De La Cruz

Eduard Elias Marquez Zavaleta

Asesor:

Mg. Rolando Javier Berrú Beltrán

Trujillo - Perú

2021

## DEDICATORIA

*A Dios por sus grandes bendiciones y brindarme inteligencia y sabiduría para desarrollarme profesionalmente. A mis padres y hermanos, quienes siempre me apoyaron incondicionalmente a lo largo de mi vida con respecto a mi crecimiento espiritual, personal y profesional. Porque junto a ellos y su apoyo podré llegar mucho más lejos, lo cual considero fundamental para poder seguir creciendo en cada aspecto de mi vida.*

*Eduard Elías Márquez Zavaleta*

*A Dios, ya que sin él ninguna meta la podría cumplir. A mis padres, quienes a pesar de todo no dudaron un solo minuto en mí y me apoyaron en todo. A mi pareja y a mi hija que son mi impulso a diario para cumplir mis objetivos.*

*Nicky Alejandro Cotrina de la Cruz*

## AGRADECIMIENTO

A Dios, por brindarnos la vida. Porque nos permite culminar satisfactoriamente nuestra etapa universitaria, brindarnos sabiduría y los medios necesarios en nuestro desarrollo profesional, como también en el desarrollo de esta investigación.

A nuestras familias, quienes nos apoyaron incansablemente y de manera incondicional, motivándonos durante toda nuestra formación profesional y poder culminar el desarrollo de esta investigación.

Gracias Mg. Rolando Javier Berrú Beltrán, por su constante asesoría y apoyo en la realización de esta investigación, como también por el tiempo brindado para responder a nuestras dudas.

Al Mg. Víctor E. Dávila Rodríguez, Coordinador de Carrera de Ingeniería de Sistemas Computacionales de la Universidad Privada del Norte, por su apoyo en las coordinaciones necesarias para poder sustentar este trabajo de investigación.

Nuestro agradecimiento a la empresa Inka Mall S.A.C., por permitirnos acceder a la información necesaria para el desarrollo de esta investigación.

## TABLA DE CONTENIDOS

<b>DEDICATORIA.....</b>	<b>2</b>
<b>AGRADECIMIENTO.....</b>	<b>3</b>
<b>ÍNDICE DE TABLAS.....</b>	<b>5</b>
<b>ÍNDICE DE FIGURAS.....</b>	<b>7</b>
<b>ÍNDICE DE ECUACIONES.....</b>	<b>9</b>
<b>RESUMEN.....</b>	<b>10</b>
<b>ABSTRACT.....</b>	<b>11</b>
<b>CAPÍTULO I. INTRODUCCIÓN.....</b>	<b>12</b>
<b>CAPÍTULO II. METODOLOGÍA.....</b>	<b>37</b>
<b>CAPÍTULO III. RESULTADOS.....</b>	<b>53</b>
<b>CAPÍTULO IV. DISCUSIÓN Y CONCLUSIONES.....</b>	<b>65</b>
<b>REFERENCIAS.....</b>	<b>68</b>
<b>ANEXOS.....</b>	<b>72</b>

## ÍNDICE DE TABLAS

<b>Tabla 1.</b> Descripción del instrumento para el indicador de porcentaje de cumplimiento de estándares de nomenclatura .....	38
<b>Tabla 2.</b> Descripción del instrumento para el indicador de tiempo de implementación de codificación. ....	38
<b>Tabla 3.</b> Descripción del instrumento para el indicador de tiempo de corrección de errores de codificación.....	39
<b>Tabla 4.</b> Resultados de la conversión a minutos de los tiempos obtenidos al implementar código de programación manual.....	43
<b>Tabla 5.</b> Detalle de la cantidad correcta e incorrecta y porcentaje de cumplimiento del uso de estándares de nomenclatura en el código de programación manual. ....	44
<b>Tabla 6.</b> Resultados de la conversión de los tiempos invertidos al solucionar errores en el código de programación realizado manualmente. ....	45
<b>Tabla 9.</b> Cuadro comparativo con respecto a metodologías de desarrollo de proyectos de software. ....	46
<b>Tabla 8.</b> Resultados de la conversión a minutos de los tiempos de implementación de código en el Post-Test. ....	48
<b>Tabla 9.</b> Detalle de la cantidad correcta e incorrecta y porcentaje de cumplimiento del uso de estándares de nomenclatura en el Post-Test. ....	49
<b>Tabla 10.</b> Resultados de la conversión a minutos de los tiempos de corrección de errores en el Post-Test. ....	49
<b>Tabla 11.</b> Contraste de resultados de pruebas Pre y Post Test de la etapa de implementación de un software.....	53
<b>Tabla 12.</b> Datos estadísticos descriptivos para la prueba t en el cumplimiento de estándares de nomenclatura. ....	54
<b>Tabla 13.</b> Resultados de XLSTAT en la prueba t para las muestras Pre y Post Test de la etapa de implementación de un software. ....	54
<b>Tabla 14.</b> Comparación de resultados del PreTest vs PostTest de la influencia reflejada en la etapa de implementación de un software. ....	56
<b>Tabla 15.</b> Contraste de resultados de pruebas Pre y Post Test de la dimensión de estándares de nomenclatura. ....	57
<b>Tabla 16.</b> Datos estadísticos descriptivos para la prueba t en el cumplimiento de estándares de nomenclatura. ....	58

<b>Tabla 17.</b> <i>Resultados de XLSTAT en la prueba t para las muestras Pre y Post Test del cumplimiento de estándares de nomenclatura.</i> .....	58
<b>Tabla 18.</b> <i>Comparación de resultados del PreTest vs PostTest de estándares de nomenclatura.</i> .....	60
<b>Tabla 19.</b> <i>Contraste de resultados de pruebas Pre y Post Test de la dimensión de codificación.</i> .....	61
<b>Tabla 20.</b> <i>Datos estadísticos descriptivos para la prueba t para los tiempos de implementación de código</i> .....	61
<b>Tabla 21.</b> <i>Resultados de XLSTAT en la prueba t para las muestras Pre y Post Test del tiempo de implementación de código.</i> .....	62
<b>Tabla 22.</b> <i>Comparación de resultados del PreTest vs PostTest del cumplimiento de estándares de nomenclatura.</i> .....	64

## ÍNDICE DE FIGURAS

<b>Figura 1.</b> Arquitectura ODBC (Open Data Base Conectivity). Fuente: Simba, S.F. ....	28
<b>Figura 2.</b> Esquema del proceso de la metodología Scrum. Fuente: Muradas, Y, 2018.....	32
<b>Figura 3.</b> Evaluación de la zona de aceptación o rechazo del cumplimiento de la etapa de implementación de un software. Fuente: Creación propia. ....	55
<b>Figura 4.</b> Gráfico de barras de la comparación de las muestras del Pre-Test vs Post-Test de la etapa de implementación de un software. Fuente: Creación propia. ....	56
<b>Figura 5.</b> Evaluación de la zona de aceptación o rechazo del cumplimiento de estándares de nomenclatura. ....	59
<b>Figura 6.</b> Gráfico de barras de la comparación de las muestras del Pre-Test vs Post-Test de estándares de nomenclatura .....	60
<b>Figura 7.</b> Evaluación de la zona de aceptación o rechazo de la eficacia de la codificación. ....	62
<b>Figura 8.</b> Gráfico de barras de la comparación de las muestras del Pre-Test vs Post-Test de la eficacia en la codificación. ....	63
<b>Figura 9.</b> Matriz de validación de expertos. ....	79
<b>Figura 10.</b> Aspectos del plan de auditoria a la codificación de software de la empresa Inka Mall S.A.C.....	80
<b>Figura 11.</b> Ticket N° 01 del historial de tiempos de codificación. ....	81
<b>Figura 12.</b> Ticket N° 02 del historial de tiempos de codificación. ....	82
<b>Figura 13.</b> Ticket N° 03 del historial de tiempos de codificación. ....	83
<b>Figura 14.</b> Ticket N° 04 del historial de tiempos de codificación. ....	84
<b>Figura 15.</b> Ticket N° 05 del historial de tiempos de codificación. ....	85
<b>Figura 16.</b> Ticket N° 06 del historial de tiempos de codificación. ....	86
<b>Figura 17.</b> Ticket N° 07 del historial de tiempos de codificación. ....	87
<b>Figura 18.</b> Ticket N° 08 del historial de tiempos de codificación. ....	88
<b>Figura 19.</b> Diagrama de Casos de Uso del Negocio .....	93
<b>Figura 20.</b> Diagrama del caso de uso - Obtener base de datos .....	93
<b>Figura 21.</b> Diagrama del caso de uso - Establecer el tipo de datos .....	94
<b>Figura 22.</b> Diagrama de caso de uso - Crear plantillas.....	94
<b>Figura 23.</b> Diagrama de caso de uso - Generar código .....	95

- Figura 24.** Diseño Mockups de la interfaz del primer paso (Conexión a la base de datos) del generador ..... 95
- Figura 25.** Diseño Mockups del segundo paso (Definir tipo de dato) del generador..... 95
- Figura 26.** Diseño Mockups del tercer paso (Seleccionar plantilla) del generador..... 95



## ÍNDICE DE ECUACIONES

<b>Ecuación 1.</b> Conversión a minutos .....	42
<b>Ecuación 2.</b> Sumatoria de palabras .....	44
<b>Ecuación 3.</b> Porcentaje de cumplimiento de estándares de nomenclatura. ....	44
<b>Ecuación 4.</b> Porcentaje de eficacia del uso de estándares de nomenclatura.....	50
<b>Ecuación 5.</b> Porcentaje de eficacia de codificación.....	50
<b>Ecuación 6.</b> Porcentaje de tiempos estimados para Back End .....	51
<b>Ecuación 7.</b> Promedio del historial de tiempos estimados de codificación .....	51
<b>Ecuación 8.</b> Porcentaje de eficacia de la etapa de implementación de un software .....	52

## RESUMEN

El presente trabajo de investigación se realizó con el objetivo de determinar la influencia del uso de un generador de código de programación en la etapa de implementación de un software de la empresa Inka Mall S.A.C.

El tipo de estudio fue pre-experimental; con una muestra constituida por ocho tablas del modelo de base de datos de un proyecto de la empresa. Para la recolección de datos se utilizó la técnica de observación con el uso de tres fichas de observación.

Las dimensiones comprendidas en la etapa de implementación de un software fueron, codificación y estándares de nomenclatura, mientras que las dimensiones comprendidas en el generador de código de programación fueron funcionalidad y usabilidad. Los resultados obtenidos demostraron que la generación de código cumple al 100% los estándares de nomenclatura establecidos por la empresa, como también superó las expectativas al aumentar hasta en 4365.82% la eficacia con relación a la codificación.

Con base en lo mencionado, podemos concluir que el generador es eficaz e influye de manera positiva en la etapa de implementación de un software, reduciendo así los tiempos de implementación y generando código estandarizado y de calidad.

**Palabras clave:** CRUD, código de programación, estándares de nomenclatura, modelo de base de datos, generador, software.

## ABSTRACT

The present research work was carried out with the objective of determining the influence of the use of a programming code generator on the implementation stage of software of the company Inka Mall S.A.C.

The type of study was pre-experimental; with a sample consisting of eight tables of the database model of a company project. For the data recollection, the observation technique and three observation cards were used.

The dimensions included in the implementation stage of a software were the coding and naming standards, while the dimensions included in the programming code generator were functionality and usability. The results obtained showed that the generation of code meets 100% naming standards established by the company, as well as exceeding expectations by increasing up to 4365.82% efficiency in relation to coding.

Based on the above, we can conclude that the generator is effective and positively influences the implementation stage of a software, thus reducing implementation time and generating standardized and quality code.

Keywords: CRUD, programming code, naming standards, database model, generator, software.

## CAPÍTULO I. INTRODUCCIÓN

### 1.1. Realidad problemática

En la actualidad, los sistemas informáticos han influenciado de gran manera al permitir automatizar procesos en diferentes rubros de negocios y junto con ello también mejoró las buenas prácticas y el cumplimiento de estándares en el desarrollo de estos, para así poder brindar un software de calidad a los usuarios y a la vez también ser entendible a la vista de los programadores para obtener mejores resultados en la identificación de mejoras o cambios en los sistemas, de esta manera se puede cumplir con la teoría que menciona Espinel, Acosta y García (2017), en su artículo titulado Estándares Para La Calidad De Software del año 2017, en el cual menciona que las metodologías y estándares pueden definir algunos criterios de desarrollo teniendo como objetivo principal crear software de buena calidad.

Asimismo, llegan a la conclusión que la evaluación de los estándares en las compañías u organizaciones, fija un punto de referencia al estado actual, brindando la posibilidad de verificar las fortalezas y debilidades, y así luego poder diseñar un plan de acción el que permita generar productos de software de calidad (Espinel et al., 2017).

Por otro lado, el trabajar como programador para implementar un software, demanda algunas complicaciones que la ASOCIACIÓN ESPAÑOLA DE PROGRAMADORES INFORMÁTICOS (s.f.) define como las cinco cosas más frustrantes de la profesión del programador, donde resalta las prisas en los plazos de entrega, esto se ve reflejado en los proyectos informáticos que se basan en metodologías ágiles, cuando por cumplir con las fechas de entrega se deja de aplicar las buenas prácticas en la programación y solo se basa en la funcionalidad a pesar que posteriormente genere otros problemas de código.

Por tal motivo, en estos últimos años se han desarrollado distintas aplicaciones que permiten automatizar procesos repetitivos en el desarrollo de software, que anteriormente solían hacerse a mano, y de esta manera mejorar en velocidad, eficiencia, tolerancia a errores, portabilidad e independencia; tal como menciona Huari (2020) en su investigación “Revisión Sistemática Sobre Generadores De Código Fuente Y Patrones De Arquitectura”, donde afirma que para disminuir el riesgo de demoras en la codificación de los software, se puede usar herramientas informáticas las cuales mejoran la productividad, dentro de estas se puede mencionar a los Generadores de Código Fuente (GCF), que son aplicaciones que producen código automáticamente y son utilizados en situaciones donde la lógica de un flujo del sistema es repetitiva (Huari, 2020).

Cabe mencionar, que también se encontraron limitantes en el uso de estos generadores ya que la gran mayoría son de uso específico para un solo lenguaje de programación; sin embargo, hay diversos generadores para cada lenguaje de desarrollo de software, tal como lo menciona Adárraga y Oliveros (2014) en su investigación “Modelado UML Del Generador De Código De Aplicaciones Web TGENP”, que hay muchos generadores de código existentes en el mercado, estos pueden ser gratuitos o pagos, los cuales son usados en diferentes lenguajes de programación y conexiones a bases de datos, contemplando de forma independiente un nivel de dificultad diferente, pero con el mismo alcance y grado de robustez frente al producto final (Adárraga y Oliveros, 2014).

Asimismo, otras de sus limitaciones es que no permite al usuario manipular las plantillas de generación para adecuarlos a sus necesidades y eso conlleva a que no se pueda tener control y acceso a los códigos, ya que se usan códigos prefabricados, tal

como se ve reflejado en el artículo publicado por Cordero (2015), en el portal web EL FINANCIERO.

También al evaluar el mercado de la industria de software en el Perú, se obtuvo la información brindada por el Diario Gestión en el año 2017, donde mencionaba que el sector software está en constante crecimiento dentro de nuestro país y que según el gerente general de La Asociación Peruana de Software (APESOFT) calcula que en los próximos cinco años, el ratio podría subir a 12%, siempre y cuando se implemente un Ministerio de Tecnologías de Información y Comunicación, como también resalta la cantidad de empresas dedicadas al desarrollo de software que son aproximadamente 500 empresas peruanas de este rubro, y 95 pertenecen a la APESOFT, muy aparte de los programadores y desarrolladores de software que implementan software de manera independiente.

Actualmente, en la realidad de Inka Mall se observa que en el proceso de implementación de un software se invierte demasiado tiempo al momento de desarrollar la parte del Back End de procesos comunes o repetitivos como lo puede ser un CRUD de una tabla, dentro de ello se observó que programando de manera manual los programadores suelen olvidarse de aplicar los estándares de nomenclatura establecidos por la empresa y también que por ser procesos comunes, presentan errores generados al momento de concluir con la codificación manual y se invierte tiempo extra para poder resolverlos, lo cual no aporta realmente a cumplir con un objetivo de la empresa que es obtener la norma ISO/IEC 9126 para Calidad de Software.

Se han considerado los siguientes estudios de investigación como antecedentes sobre los generadores de código en la implementación de software:

Los autores Radosevic y Magdalenic (2011), en la investigación “Source Code Generator Base on Dynamic Frames”, tuvieron como objetivo reducir costos de codificación en el desarrollo de aplicaciones web. Para esto, se desarrolló un generador enfocado en el modelo SCT (Specifications, Configuration and Templates) que creaba dinámicamente archivos XML. Los resultados mostraron que el generador permitía realizar actualizaciones de configuración, bajo diversas conexiones válidas, modificando así el código generado, y creando nuevas especificaciones y plantillas según la necesidad. Por ello se llegó a la conclusión de que el producto era totalmente configurable y brindaba un resultado que ayuda a reducir costos en la codificación de software, sin estar vinculado exactamente a un lenguaje de programación.

Los autores Manikandan, Ramanujam y Sadayappan (2010), en la investigación “C-to-CUDA Code Generation for Affine Programs”, tuvieron como objetivo facilitar el desarrollo manual de código CUDA. Para lograr esto, se desarrolló un sistema generador de código CUDA (Compute Unified Device Architecture) el cual transformaba código C a CUDA en una programación paralela. Los resultados mostraron que al usar este código se lograba optimizar la memoria global y el acceso a la memoria compartida de la GPU, por lo que llegaron a la conclusión, que el framework creado no solo facilitaba la programación paralela de código CUDA, sino que también al usarlo, este era eficiente al optimizar el procesamiento de gráficos.

Los autores Picón y Zulay (2016), en la investigación “Generación Automática de Código Basada en Modelos UML”, tuvieron como objetivo proponer algunas buenas prácticas aplicables en proyectos de desarrollo de sistemas. Ellos plantean que los generadores basados en UML, pueden producir código consistente mediante

prácticas estándar de programación, y así obtener un código seguro y más fácil de mantener que el creado manualmente. Según el análisis de resultados, fue que los generadores no están suficientemente desarrollados para suministrar código complejo, llegando así a la conclusión de que los generadores orientados a objetos presentan limitaciones; sin embargo, el producto generado sí aporta un código confiable y seguro bajo estándares de calidad.

Los autores Rincón, Aguilar e Hidrobo (2010), en la investigación “Generación Automática de Código a Partir de Máquinas de Estado Finito”, tuvieron como objetivo agilizar el desarrollo de software e incrementar su confiabilidad. Por ello, propusieron el desarrollo de un generador de código fuente para lenguajes orientados a objetos basados en UML. Según los resultados, la herramienta generaba un código compilable y ajustado a la funcionalidad expresada en el modelo UML, creando clases, atributos y operaciones. En base a lo mencionado llegaron a la conclusión, que la generación de código es de gran utilidad para acelerar los procesos de desarrollo y obtener resultados confiables y de calidad, ya que la codificación manual se torna algo complicada y con posibles errores.

Lazetic et al. (2012), en la investigación “A Generator of MVC-based Web Applications”, tuvieron como objetivo reducir el tiempo de codificación y mejorar el código realizado manualmente. Por tal motivo propusieron realizar un generador de código MVC (Modelo, Vista, Controlador) para aplicaciones web y reducir errores de codificación. Como resultado, lograron obtener un generador de código con cierta simplicidad, y de acuerdo con ello, llegaron a la conclusión que cumplía con los requerimientos establecidos, pero para una problemática específica; sin embargo, brindó solución al enfoque propuesto y redujeron tiempos de codificación al probarlo en un sistema de gestión que podía demandar 800 horas de codificación manual.



Rosales et al. (2015), en su artículo “An analysis of tools for automatic software development and automatic code generation”, tuvieron como objetivo determinar las características de las herramientas de automatización de código de programación y ver su aporte a la productividad de los desarrolladores. Como resultados llegaron a obtener las características de cada herramienta seleccionada y pudieron determinar la calidad de ellas bajo el modelo de la ISO/IEC 9126 estándar. Finalmente llegaron a la conclusión que aun así hay aspectos por mejorar en cada generador de código automático y que sería mejor si estos se ajustan a una realidad específica.

Los autores Mattingley y Boyd (2011), en su investigación “CVXGEN: a code generator for embedded convex optimization”, tuvieron como objetivo reducir tiempo al momento de resolver problemas de optimización convexa. Por ello propusieron e implementaron un generador de código C que pueda incrustarse en sistemas de alto nivel sin realizar codificación manual y que al ser compilado se produzca un software que brinde soluciones a estos problemas. Los resultados que obtuvieron fueron que el código generado resolvió problemas hasta en 14 iteraciones propuestas, por lo que llegaron a la conclusión, que el código generado por CVXGEN es adecuado para incrustar en aplicaciones en tiempo real, es robusto y a su vez el primer generador de optimización convexa, mostrando viabilidad y rapidez.

El autor Gaitán (2017) en su investigación “Líneas de Productos Software: Generando Código a Partir de Modelos y Patrones”, tuvo como objetivo minimizar costos y mejorar la eficiencia de procesos en una línea de producto software. Por ello propusieron desarrollar un generador de código basado en patrones de diseño como MVC (Modelo, Vista, Controlador) para generar aplicaciones ágiles en ambientes transaccionales. Como resultados pudieron obtener validaciones correctas en los CRUD (Create, Read, Update and Delete) generados de acuerdo a la base de datos

seleccionada. Finalmente, llegaron a la conclusión que el generador es una herramienta que permite reducir tiempo de desarrollo, incrementa la productividad de los desarrolladores y que brinda un código de programación, flexible y correcto. Los autores Herrera, Quiñonez y Casierra (s.f.) en su investigación “Generador automático de aplicaciones web e interfaces de usuarios con funcionalidad responsiva en el lenguaje Python”, tuvieron como objetivo reducir actividades repetitivas que produzcan pérdida de tiempo y retraso en los periodos de entrega de un software. Por ello propusieron crear un generador de código basado en buenas prácticas y código escalable que permita crear aplicaciones web. Como resultados obtuvieron la generación de CRUDs con interfaces agradables e intuitivas que a la vez son responsivas o adaptables al tamaño de visualización, y se llegó a la conclusión que dicho proyecto también ayudaba a reducir costos al momento de desarrollar un software, brindando un código con buenas prácticas y que el producto obtenido no solo se puede conectar a un motor de base de datos, sino también a otras. Los autores Martínez, Camacho y Biancha (2010), en su investigación “Diseño De Framework Web Para El Desarrollo Dinámico De Aplicaciones”, tuvieron como objetivo reducir tiempos de desarrollo de software, y estructurar el código de sistemas web de acuerdo con los estándares de codificación. Por ello propusieron la implementación de un framework orientado al lenguaje de programación PHP, que establezca y brinde un código estandarizado. Al finalizar la investigación llegaron a la conclusión que su propuesta brinda autenticación, administra roles y usuarios, genera código a partir de tablas de base de datos, y gracias a ello pudieron cumplir con los estándares establecidos por los desarrolladores, siendo así un código de calidad y que aporta a la reducción de tiempo del desarrollo de un software.

Los autores Alonso et al. (2012), en su investigación “Generación Automática de Software para Sistemas de Tiempo Real: Un Enfoque basado en Componentes, Modelos y Frameworks”, tuvieron como objetivo describir un enfoque que beneficiara a los desarrolladores y que aporte de manera considerable al desarrollo de RTS (Sistemas de Tiempo Real). Por ello realizaron un análisis de cada uno de los diseños y propusieron una técnica que se enfoque en la combinación de componentes y diseños de software, logrando como resultados, la mejora de gestión de la complejidad de los RTS, minimización de riesgos de desarrollo, reutilización de código y mejora de comunicación en los desarrolladores, llegando a la conclusión que dicha técnica es la suma de un enfoque basada en uso de frameworks de programación orientada a objetos.

Los autores Chávez, Hermosa y Villacís (2016), en su investigación “Generación de Código Fuente para Gestión de Información de MySQL, SQL Server y Access para Java, PHP y ASP”, tuvieron como objetivo facilitar el trabajo de los programadores al momento de desarrollar un sistema y mejorar tiempos de entrega. Por ello realizaron un generador de código fuente para controlar las bases de datos MySQL, SQL Server y Access, usando los lenguajes ASP, JSP y PHP en base a plantillas. Luego de ello, los autores pudieron concluir y demostrar que la correcta funcionalidad y uso de este generador y la autogeneración de código fuente, puede ahorrar varias horas de trabajo en un proyecto informático, como también recurso humano y por ende costos económicos.

Los autores Adárraga y Oliveros (2014), en su investigación “Modelado UML Del Generador De Código De Aplicaciones Web TGENP”, tuvieron como objetivo mejorar el funcionamiento y estructura del generador TGENP para que su código sea entendido más rápido y fácilmente por los desarrolladores. Por ello, plantearon

realizar un modelado UML para la implementación de la segunda versión del software y así mejorar la principal funcionalidad de TGENP que es la de generar código fuente PHP con la arquitectura de desarrollo MVC la cual se logra conectar a una base de datos MySQL. Luego de ello, los autores pudieron concluir y demostrar que TGENP puede cubrir aspectos del desarrollo software que brinda al desarrollador la facilidad de luego de haber realizado la creación de su base de datos, obtener todos los métodos básicos, vistas, modelos y demás código en sólo 3 pasos.

Los autores González, Álvarez y Acosta (2012), en su investigación “jPET 2.0: Un generador automático de casos de prueba sobre programas Java”, tuvieron como objetivo mejorar el generador jPET 2.0 para su mejor funcionamiento y aplicación en la creación de casos de prueba desarrollados en Java. Para ello, propusieron añadir una nueva funcionalidad a este generador de código la cual se encargaría de generar pruebas en código java a partir de los casos de prueba almacenados en los ficheros .xml que este mismo almacena, y así el desarrollador pueda verificar el funcionamiento del código que pretende testear. Al finalizar su investigación, los autores pudieron concluir que la funcionalidad añadida realiza la creación de casos de prueba propuestos y que jPET es una herramienta muy útil para los desarrolladores de software gracias al tiempo que ahorran al obtener los tests de forma automática.

La autora Huari (2020), en su investigación “Revisión Sistemática Sobre Generadores De Código Fuente Y Patrones De Arquitectura”, tuvo como objetivo brindar información detallada y clara a los desarrolladores para que puedan elegir la mejor opción que más se adecúe a sus requerimientos: por ello, planteó realizar una revisión sistemática a las principales herramientas de generación de código fuente (GCF), con el fin de reducir las deficiencias en la estructura, funcionamiento e integración de los componentes de software. Al finalizar su investigación, pudo

llegar a la conclusión que los GCF permiten automatizar la generación de código, facilitando el desarrollo y mantenimiento de aplicaciones, reduciendo el tiempo y minimizando errores de programación.

El autor Urbina (2019), en su investigación “Generador De Código De Funcionalidades Tipo CRUD En La Mantenibilidad De Software Aplicado A Sistemas De Información Empresariales”, tuvo como objetivo demostrar que el desarrollo y aplicación de un software generador de código tipo CRUD favorece la mantenibilidad de código en sistemas de información empresariales; por ello, planteó realizar un generador de código para funcionalidades tipo CRUD con la estructura de programación orientada a objetos y así ayudar en la mantenibilidad de los sistemas de información empresariales. Al finalizar su investigación, pudo llegar a la conclusión el software generador de código tipo CRUD tiene una influencia positiva sobre la mantenibilidad de software el cual respeta los principios de código de mejora del mismo.

El autor Sánchez (2018), en su investigación “DotNetGenerator: Generador de Código para Arquitectura Microsoft .NET a partir de modelos ISML”, tuvo como objetivo a reducir tiempos en la implementación de un software, como también los costos del desarrollo y a la vez garantizar la calidad de estos productos software; por ello, propuso realizar e implementar un generador de código basado en el lenguaje .Net, con la estructura MVC y el cual es guiado por modelos ISML (Information Systems Modeling Language) para ayudar en las tareas repetitivas. Al finalizar su investigación llegó a la conclusión que dicho generador llegó a brindar la ayuda necesaria reduciendo tiempos en implementación de tareas comunes haciendo uso de menor esfuerzo y costos en los equipos de desarrollo.

El desarrollo de este proyecto de investigación se justifica con el propósito de desarrollar un software, en el cual se pueda obtener el beneficio de reducir el tiempo de la etapa de implementación del mismo, lo cual es favorable también para los programadores, ya que al brindarles un software capaz de generar código de programación le será más eficaz que programarlo o escribir todas esas líneas de código de manera manual, en donde se desperdiciaría mucho tiempo y que incluso a veces afectaría al cronograma de entregas de requerimientos según el proceso de implementación del software en cuestión.

Por otro lado, el tiempo que se invierte en programación de sistemas también requiere muchas veces el trabajo de horas extras, lo cual al disminuir este tiempo se podrá ahorrar costos que en estas circunstancias demandarían pagos por el tiempo extra laborado, movilidad, alimentación e incluso mayor uso de energía al usar los equipos, y esto afecta tanto a las empresas como también a los programadores independientes según su realidad.

Consecuentemente, la reducción de horas de trabajo también logra aportar a lo que se viene a conocer como Green Computing, que es el uso ambientalmente responsable y ecológico de las computadoras y recursos eléctricos, como también mejora la capacidad de productividad de los programadores, ya que muchas veces suelen sacrificar horas en familia, amigos, estudios e incluso alimentación, y esto genera un impacto de manera negativa en su salud física, emocional y también su rendimiento laboral.

Además, al analizar los puntos mencionados de sacrificios de los programadores, se puede describir que también son afectados de manera social, al no interactuar con su familia y amigos suele tornarse un ambiente antisocial y eso afectaría a la calidad de vida de los programadores, ya que se aíslan del mundo y suelen estar desinformados

de lo que pasa en su entorno, como también si fuera el caso de tener hijos pequeños, estos se verían afectados al no compartir momentos con sus padres.

En el desarrollo del presente proyecto de investigación, se encontraron las siguientes limitaciones:

Fue difícil realizar la recolección de datos, ya que no se podía interrumpir a los programadores dentro de sus horarios laborales, lo cual afectaba a la producción de la empresa; sin embargo, como solución a esta restricción, aprovechamos que uno de los autores es trabajador de la empresa y este recolectó los datos de tiempos de desarrollo dentro de su producción laboral en la empresa, sin ser afectada ninguna de las dos partes.

El análisis para encontrar la cantidad de errores de los estándares de nomenclatura tuvo que realizarse en la misma empresa por confidencialidad y fuera de los horarios laborales, para ello se propuso realizar la obtención de datos en el momento de descanso o al finalizar el horario laboral diario.

Conjuntamente, en este trabajo de investigación se recogió conceptos como:

### **SOFTWARE**

Según Sánchez (s.f.), en su documento de definiciones llamado “Software 1. Sistema Operativo. Software de Aplicación” (pp. 2-3), menciona que se conoce como software al componente básico de la informática que brinda a los usuarios un entorno gráfico agradable, como también menciona de manera técnica que este se conoce por su soporte lógico de un sistema informático el cual procesa programas y datos.

Posteriormente, también menciona que existe una clasificación de este término, que vendría a ser la siguiente:

- **Software de sistema:** este diseño de software está orientado básicamente al sistema informático que brinda la comunicación de todas las características internas de una computadora para que esta pueda funcionar y brindar interfaces de uso del equipo, también es conocido como Sistema Operativo.
- **Software de Programación:** este básicamente es el conjunto de herramientas que permiten a los programadores desarrollar programas informáticos, son plataformas que ayudan a la creación de otros softwares o aplicaciones.
- **Software de aplicación:** son los que permiten a los usuarios tener facilidad en el desarrollo de tareas específicas y brindan automatización o asistencia en cualquier concepto de negocio, diversión (juegos), entre otros. Estos son el producto creado por medio del uso de cualquier software dedicado para programación.

Según Berzal (2004) en su libro “Desarrollo De Sistemas De Información” (pp. 3-18), menciona las etapas del proceso de desarrollo de software, las cuales son:

- Planificación
- Análisis
- Diseño
- Implementación
- Pruebas
- Instalación o despliegue
- Uso y mantenimiento

Basados en estas estas etapas mencionadas nos enfocaremos específicamente en:



## **Implementación**

Dentro de esta etapa, Berzal menciona que es fundamental haber comprendido el problema que se quiere resolver, las funciones a crear y el diseño que debe tener el sistema, así también, los desarrolladores deben conocer sobre principios básicos de diseño y análisis que permita crear un sistema de información de calidad.

En esta fase se selecciona las herramientas adecuadas y el entorno de desarrollo más favorable para la implementación del sistema, como también un lenguaje de programación apropiado de acuerdo al tipo de sistema a construir, ya que en esta etapa es momento de desarrollar cada una de las funcionalidades establecidas, para lograr brindar el sistema adecuado y requerido.

También menciona como recomendaciones, que a la hora de programar se debe procurar que el código no resulte indescifrable, sino que este debe ser legible y que pueda ser entendido por otros desarrolladores, se debe elegir cuidadosamente los identificadores de las variables, estructura de datos, selección de algoritmos adecuados para evitar este problema, como también se debe mantener la lógica de la aplicación lo más sencilla posible y comentar de forma adecuada el código de nuestros programas, por último, siempre es recomendable facilitar la interpretación visual de nuestro código de programación estructurándolos por bloques de código, aplicando sangrías y saltos de línea.

## **Estándares de Nomenclatura de Código**

Para poder desarrollar de manera legible y ordenada al implementar un software de aplicación, es necesario aplicar algunos estándares en la nomenclatura de la codificación al momento de programar. A continuación, mencionaremos algunos de ellos:

- **Estilo Pascal (PascalCase)**

Según Benjamín (2008), en su artículo web “Estilo de programación y convención de nombres II” del portal Codigolinea, menciona que este estilo es redactado con la primera letra del identificador y la primera letra de las siguientes palabras concatenadas en mayúsculas. También menciona que este estilo de mayúsculas y minúsculas puede aplicarse en identificadores de tres o más caracteres, por ejemplo: **ImageSprite**

Continuando con los demás estilos en la nomenclatura de código de programación, podemos mencionar también que, de acuerdo a Acedo (2017) en su artículo llamado “Estándares de nomenclatura: Snake Case, Kedab Case, Camel Case” del portal Apuntes de Programación, define estilos para estandarizar la nomenclatura de código fuente de un sistema en desarrollo:

- **Estilo Camel Case**

Este estilo debe estar conformado por la primera letra del identificador en minúscula y si tuvieran más palabras concatenadas, estas deben tener la primera letra en mayúscula, por ejemplo: `imageSprite`

- **Upper Case**

Está conformado por todas las letras del identificador escritas en mayúsculas, por ejemplo: `EJEMPLODENOMENCLATURA`.

Este mayormente es usado, para las constantes definidas en PHP.

- **Snake Case**

Se caracteriza por que cada una de las palabras se separa por un guion bajo (`_`). Es común en los nombres de variables y funciones de lenguajes como C, aunque también Ruby y Python lo adaptaron.

Ejemplo: `ejemplo_de_nomenclatura`.

- **Kebab Case**

Por último, este estilo es igual que el Snake Case, pero esta vez, son guiones medios (-) los que separan las palabras. Su uso más común es de las urls.

Ejemplo: ejemplo-de-nomenclatura

Los sistemas implementados mayormente necesitan estar relacionados a un modelo de base de datos estructurado según lo que se requiere en el software, y por tal motivo también deben tener una forma de establecer una comunicación entre ambos. Esta comunicación se realiza a través de componentes conformados de pequeños códigos o también llamados cadenas de conexión.

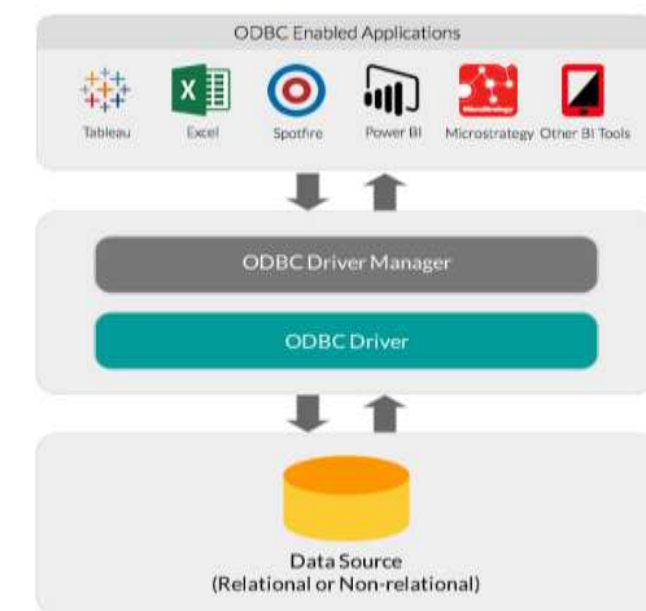
Según el portal web Simba (s.f.), en su artículo titulado “What is ODBC – Open Database Connectivity” describe lo siguiente sobre las cadenas de conexión - ODBC:

**ODBC**

Open Database Connectivity (ODBC) es descrita como una interfaz de programación de aplicaciones (API), la cual es abierta para acceder a una base de datos. Según historia en 1992, Microsoft se asocia con Simba para construir el primer controlador ODBC del mundo; SIMBA.DLL, y a partir de este nació el acceso a datos basado en estándares. Es bueno resaltar que se menciona que al usar las declaraciones ODBC en un programa, puede acceder a los archivos en una cantidad de bases de datos comunes diferentes, y que además del software ODBC, se necesita un módulo o controlador separado para acceder a cada base de datos.

### Componentes:

En la arquitectura ODBC, una aplicación se conecta con la interfaz ODBC, quién a su vez se comunica con el gestor de controlador, y este último se conecta directamente a la base de datos, usando un driver ODBC.



**Figura 1.** Arquitectura ODBC (Open Data Base Connectivity). Fuente: Simba, S.F.

### Cadenas de Conexión

Según el portal Autodesk (2016), en el artículo web titulado “Conexión con una base de datos ODBC”, menciona que ODBC para establecer una conexión de base de datos, utiliza una línea de comando estructurada de acuerdo al motor de base de datos al que se desea conectar, y esta se puede generar de manera automática. Estas líneas son conocidas como “Cadenas de Conexión” y tienen información relevante para conectar una base de datos, como son: Server, nombre de la base de datos, user, password.

A continuación, se presentan algunos ejemplos de cadenas de conexión:

### **ODBC: Cadena de conexión para “Access”**

```
DBQ=D:\persona.mdb;Driver={Microsoft Access Driver
(*.mdb)};DriverId=281;FIL=MS
Access;MaxBufferSize=2048;MaxScanRows=8;PageTimeout=5;SafeTra
nsactions=0;Threads=3;UID=;UserCommitSync=Yes
```

### **ODBC: Cadena de conexión para “SQL Server”**

```
DRIVER=SQL
Server;SERVER=.\SQLEXPRESS;UID=usuario;PWD=;APP=Visual
Basic;DATABASE=persona;Network=DBMSSOCN
```

### **ODBC: Cadena de conexión para “MySQL”**

```
DRIVER=MySQL ODBC 3.51
Driver;DESC=;DATABASE=persona;SERVER=localhost;UID=usuario;P
ASSWORD=;PORT=3306;OPTION=;STMT=;
```

## **Metodologías De Desarrollo Para Proyectos De Software**

### **Metodologías ágiles.**

Según el portal web Open Webinars, en el artículo de Muradas, Y. (2018), titulado “Las 3 metodologías ágiles más usadas”, menciona que en la actualidad el mundo de los negocios y la tecnología deben afrontar cambios constantes; por ello, las gestiones de los proyectos de software deben ser ágiles, pero a la vez los requisitos y las soluciones de estos proyectos, deben ser adaptables a una nueva realidad o requerimiento de negocio.

Según el artículo, se puede definir a las metodologías ágiles como un conjunto de tareas y procedimientos dirigidos a la gestión de proyectos.

Se menciona que, dentro de estas metodologías, son usadas con más frecuencia las siguientes: Scrum, programación extrema (XP) y Kanban, de estas 3 se usó la metodología XP o programación extrema para ser aplicada al proyecto de investigación.

### **Metodología Scrum**

Según el portal web Proyectos Ágiles (s.f.), en su artículo titulado “Qué es SCRUM”, menciona que Scrum es un proceso en el que se aplican un conjunto de buenas prácticas al momento de trabajar de manera colaborativa y en equipo, siendo flexible para la gestión de desarrollo de proyectos de software.

Por otro lado, describe que este método se caracteriza por sus entregas parciales y regulares del producto final, de acuerdo a sus prioridades. También menciona que es aplicable en proyectos de entorno complejos, donde se necesita obtener resultados lo más pronto posible, ya que los requisitos pueden ser cambiantes o a veces no están bien definidos.

Según el portal web Nextu presenta la publicación de Rebeca (s.f.), con el artículo titulado “La metodología ágil más usada: SCRUM”, menciona que los avances de la tecnología y competencia, ha hecho que la exigencia por productos de calidad sea cada vez mayor, sin descuidar la ejecución rápida y la agilidad en los procesos.

También menciona que la más usada de las metodologías ágiles, es Scrum.

A continuación, se muestra una toma de datos estadísticos en relación a algunas razones para la transformación ágil, estos datos están basados en el portal web

Apiumhub (s.f.), exactamente en su artículo llamado “Transformación Ágil: Pasos, Estadísticas Y Un Ejemplo Práctico”:

- Mejora en 54% la colaboración entre equipos que no acostumbran a trabajar juntos.
- Aumento del 52% en la calidad del software.
- Aumento del 49% en la satisfacción del cliente con el producto.
- Disminución de un 43% en el tiempo de comercialización
- Reducción de un 42% en el coste de desarrollo

Según Muradas, Y. (2018), menciona los roles que conforman el equipo Scrum, los cuales son listados a continuación:

**Stakeholder:** Es el cliente, este tiene como responsabilidad establecer los requerimientos en el documento Product Backlog y recibir el producto al final de cada iteración y proporcionar un feedback (Retroalimentación) correspondiente.

**Product Owner:** Es el intermediario de comunicación entre el cliente (stakeholder) y el equipo de desarrollo, como también debe establecer la prioridad de cada uno de los requerimientos según la necesidad de la solicitud del cliente.

**Scrum Master:** La persona que desempeña este rol, actúa como facilitador ante el equipo de desarrollo eliminando aquellos impedimentos que determine durante el proceso, como también se encarga que el equipo cumpla los valores y principios ágiles, las reglas y los procesos de Scrum, orientando al grupo de trabajo.

**Scrum Team** (Equipo de desarrollo): Se encarga de desarrollar los requerimientos establecido en el Product Backlog, es un equipo auto gestionado, esto quiere decir que no necesitan de un jefe de equipo, razón por la cual los miembros se deben de

encargar de realizar las estimaciones y en base a la velocidad obtenida en las iteraciones, deben ir construyendo el Sprint Backlog.

## Esquema de Scrum



**Figura 2.** Esquema del proceso de la metodología Scrum. Fuente: Muradas, Y, 2018.

En el artículo de Muradas, también menciona algunos tipos de reuniones que se realizan en Scrum, y que estas son el pilar importante para el desarrollo de la metodología. A continuación, se menciona aquellas reuniones que se enfocan en la revisión constante de los procesos y transparencia de estos.

Las reuniones son:

**Reunión de planificación (Sprint planning meeting):** Se realiza al inicio de cada sprint, con el objetivo de planificar y establecer la cantidad de requerimientos a la que el equipo debe comprometer a desarrollar durante el próximo sprint.

**Reunión diaria (Daily Scrum):** Son reuniones con duración máxima de 15 minutos, en ellas se realiza un informe de lo que se hizo el día de ayer, qué se hará hoy y



cuáles han sido los obstáculos para el desarrollo que han surgido hasta el momento.

El objetivo, es establecer como equipo un plan para las próximas 24 horas.

**Reunión de revisión (Sprint Review):** Se realiza al final de cada sprint, en la cual se exponen los puntos terminados y los que no.

**Reunión de retrospectiva (Sprint Retrospective):** Esta reunión se efectúa al término de cada sprint, en donde se tiene como objetivo que el equipo medite y obtenga como resultados, acciones de mejora. A esta reunión debe participar todo el Equipo Scrum (Dueño de Producto, Equipo de Desarrollo y Scrum Master). Es una de las reuniones más importantes de scrum, ya que es un espacio de reflexión y mejora continua.

A continuación, se define algunos de los términos que se mencionan en el proceso de la metodología Scrum.

**Product Backlog:** Según el portal web Scrum Institute (s.f.), en artículo “The Scrum Product Backlog”, se define, como la lista de todos los requerimientos y artefactos que se deben hacer dentro del proyecto a desarrollar, y este debe ser detallado y creado por el cliente, también llamado Stakeholder.

**Sprint Backlog:** Según el portal web Scrum Institute (s.f.), en el artículo “The Sprint Backlog”, define a este documento como el almacén de un grupo de actividades necesarias para completar el desarrollo del Product Backlog. Estas actividades deben de ser actualizadas diariamente de acuerdo al avance de cada una de ellas y el conjunto de todo, se debe cumplir en un tiempo estimado y coordinado con el cliente.

Continuando con la explicación de algunos conceptos, podemos describir lo siguiente:

## **Código Fuente o de Programación**

Según el portal web Digital Guide (2018), en su artículo “El código fuente: ¿qué es y cómo se escribe?”, menciona que código fuente es llamado a todo texto legible redactado en un lenguaje de programación.

Estos códigos están compuestos por instrucciones de configuración para la ejecución de procesos que un sistema debe de hacer, los cuales posteriormente son pasados al proceso de compilación, en donde se traduce a código máquina, y así las computadoras puedan ejecutar las instrucciones y mostrar un programa o sistema en concreto que realice los procesos definidos según los requerimientos y objetivos que este programa tenga.

Básicamente el código fuente es el contenido que se desarrolla y se almacena en clases según su lenguaje de programación, y dentro de ellas se encuentra especificado cada función y método, como también la lógica de los procesos de un sistema a usar por un usuario.

## **Generadores de código de programación**

Según Durán (2014), en su artículo web “Generadores de código; agilizando el desarrollo”, menciona que los generadores de código nacen como una manera de agilizar el desarrollo, por ejemplo, si se parte siempre de un caso en el que usamos 5 librerías y nuestro diseño es de una manera en particular, entonces este generador sería muy útil, porque se podría disponer de esa plantilla ya creada y luego completar el contenido. Este proceso es muy importante si queremos diseñar un prototipo rápido o común, de lo que va a ser nuestro servicio y con el cual le podemos enseñar a un posible cliente, inversor o al propio equipo, cómo será lo que queremos desarrollar. A continuación, se mostrará algunos tipos de generadores que según Durán pueden ser de 2 tipos:

**Generadores interactivos:** este tipo de generadores son muy comunes actualmente y permiten, que, con un simple sistema de arrastrar y configurar un par de parámetros del elemento, poder generar todo el código necesario para implementar esa funcionalidad. Un ejemplo de ello es el App Studio, el generador de Aplicaciones para Windows Phone que Microsoft pone a disposición de cualquier persona.

**Generadores usando un lenguaje de modelado:** este tipo de generadores son menos comunes, pero son los más “potentes”, ya que usando una descripción del modelo que queremos crear en un lenguaje de modelado como UML, son capaces de crear un porcentaje bastante amplio del código. Un ejemplo de esto es Visual Paradigm, y una de sus ventajas es que permite generar código en diversos lenguajes como C++, Java y PHP.

### **Pruebas “t” de student**

De acuerdo a la página de soporte de Minitab (S.F.), en el tema de apoyo titulado “Pruebas de medias”, nos menciona a las pruebas t como una prueba de hipótesis de la media de una o dos poblaciones distribuidas normalmente.

Asimismo, da a conocer que existen algunos tipos de pruebas t que son aplicables en diferentes situaciones, pero en cada una de ellas se hace uso de un estadístico de prueba que sigue una distribución t bajo una hipótesis nula.

A continuación, se mencionan algunos de estos tipos de prueba:

- **Prueba t de una muestra:** Prueba si la media de una población individual es igual a un valor objetivo.
- **Prueba t de 2 muestras:** Prueba si la diferencia entre las medias de dos poblaciones independientes es igual a un valor objetivo.
- **Prueba t pareada:** Prueba si la media de las diferencias entre las observaciones dependientes o pareadas es igual a un valor objetivo.

- **Prueba t en la salida de regresión:** Prueba si los valores de los coeficientes en la ecuación de regresión difieren significativamente de cero.

## 1.2. Formulación del problema

¿De qué manera influye el uso de un generador de código de programación en la etapa de implementación de un software dentro de la empresa Inka Mall S.A.C. en el año 2019?

## 1.3. Objetivos

### 1.3.1. Objetivo general

Determinar la influencia de un generador de código de programación en la etapa de implementación de un software dentro de la empresa Inka Mall S.A.C. en el año 2019.

### 1.3.2. Objetivos específicos

- Determinar la influencia del generador de código de programación en los estándares de nomenclatura de la empresa establecida para la etapa de implementación de un software.
- Determinar la influencia del generador de código de programación en la dimensión de codificación en la etapa de implementación de un software.

## 1.4. Hipótesis

El generador de código de programación influye positivamente en la etapa de implementación de un software de la empresa Inka Mall S.A.C. en el año 2019.

## CAPÍTULO II. METODOLOGÍA

### 2.1. Tipo de investigación

**De acuerdo con el diseño de investigación:** Pre Experimental.

### 2.2. Población y muestra

- Como población se tomó 27 tablas del diseño de base de datos de 1 proyecto de software de la empresa Inka Mall S.A.C.
- Como muestra se tomó 8 tablas del diseño de base de datos de dicho proyecto, las cuales fueron seleccionadas por la empresa por ser las únicas tablas habilitadas con datos e implementadas para realizar las pruebas.

### 2.3. Técnicas e instrumentos de recolección y análisis de datos

#### **Recolección de datos**

##### **Técnica(s):**

Para la obtención del porcentaje de cumplimiento de estándares de nomenclatura, el tiempo de corrección de errores de codificación y tiempo de implementación de la codificación, se usó la técnica de observación, la cual fue aplicada en los ambientes de la empresa Inka Mall S.A.C, de los cuales, dichos datos sirvieron para el análisis y obtención de resultados, según nuestros indicadores propuestos.

##### **Instrumento(s):**

De acuerdo con las técnicas aplicadas, los instrumentos para los indicadores de porcentaje de cumplimiento de estándares de nomenclatura, tiempo de corrección de errores de codificación y tiempo de implementación de la codificación, fue aplicar fichas de observación enumeradas conforme a sus indicadores.

A continuación, se presenta la descripción de cada uno de los instrumentos que se usó para recopilar los datos requeridos, para su posterior análisis.

Tabla 1

*Descripción del instrumento para el indicador de porcentaje de cumplimiento de estándares de nomenclatura*

Nombre	Ficha de observación n° 1
Objetivo	Recolectar datos necesarios para la medición del uso de los estándares de nomenclatura en el código de programación.  Se siguió los siguientes pasos:
Procedimiento	<ol style="list-style-type: none"> <li>1. Revisar el uso de estándares de nomenclatura establecidos por la empresa en la implementación de un software.</li> <li>2. Mediante la ficha de observación N° 1, detallar las cantidades de cumplimiento o fallo del uso de estándares de nomenclatura establecidos por la empresa.</li> </ol>
Normativa	Estándares establecidos por la empresa
Público Objetivo	Autores de la investigación

Fuente: Creación propia.

Tabla 2

*Descripción del instrumento para el indicador de tiempo de implementación de codificación.*

Nombre	Ficha de observación n° 2
Objetivo	Recolectar los tiempos de duración para el desarrollo de código manual y código generado los cuales son necesarios para una posterior evaluación de reducción de tiempo en la etapa de implementación de un software.
Procedimiento	<p>Se siguió los siguientes pasos:</p> <ol style="list-style-type: none"> <li>1. Controlar el tiempo exacto de desarrollo de código manual para un CRUD por tabla.</li> </ol>

	<ol style="list-style-type: none"> <li>2. Controlar el tiempo exacto de desarrollo de código generado automático para un CRUD por tabla.</li> <li>3. Mediante la ficha de observación N° 2 , detallar los tiempos obtenidos para el desarrollo de cada CRUD.</li> </ol>
Normativa	-
Público Objetivo	Autores de la investigación

Fuente: Creación propia.

Tabla 3

*Descripción del instrumento para el indicador de tiempo de corrección de errores de codificación.*

Nombre	Ficha de observación n° 3
Objetivo	Recolectar los tiempos de duración para corregir errores en el desarrollo de código manual y código generado los cuales son necesarios para la evaluación de reducción de tiempo en la etapa de implementación de un software.
Procedimiento	<p>Se siguió los siguientes pasos:</p> <ol style="list-style-type: none"> <li>1. Controlar el tiempo invertido en la solución de errores en el desarrollo de código manual para un CRUD por tabla.</li> <li>2. Controlar el tiempo invertido en la solución de errores en el desarrollo de código generado para un CRUD por tabla.</li> <li>3. Mediante la ficha de observación N° 3, detallar los tiempos obtenidos para la solución de errores en el desarrollo de cada CRUD.</li> </ol>
Normativa	-
Público Objetivo	Autores de la investigación

Fuente: Creación propia.

## **Análisis de datos**

Para realizar el análisis de los datos obtenidos, se realizó a través del uso de la herramienta estadística llamada XLSTAT, además del tipo de prueba estadística aplicada t de Student.

### **2.4. Procedimiento**

Para poder definir nuestros instrumentos, en primer lugar, se tuvo que realizar la operacionalización de variables (Anexo - 1); en base a ello, se tomó en cuenta los datos que se requerían para poder realizar un análisis diferencial entre los datos que se obtuvieron de la programación manual, con los datos que se obtuvieron luego de aplicar el generador de código de programación.

De acuerdo con los datos que se deseaban obtener y también por la metodología a aplicar haciendo uso de la técnica de observación, se optó por aplicar fichas de observación, en donde se especifique los datos requeridos para cada indicador.

En el procedimiento de la elaboración de los instrumentos de recolección de datos propuestos se consideró lo siguiente:

- Para el indicador del porcentaje de estándares de nomenclatura, se consideró tomar como valores: los datos del evaluador, la fecha y hora en que se realizó la prueba, el nombre de la tabla según la base de datos de prueba, y de acuerdo a los tipos de estándares de nomenclatura que la empresa tenía establecidos, se solicitó completar en una tabla de datos la cantidad de palabras exactas en la cual se cumplía los estándares de nomenclatura, como también la cantidad de palabras que no cumplían los estándares en el código fuente, finalmente también se optó por agregar la columna de observaciones a la tabla, para poder tener mayor alcance de la prueba, y apuntar algunos detalles que el evaluador podía tomar en consideración para el análisis (Anexo - 2).



- Para el indicador de tiempo de codificación se tomó como datos: el nombre del evaluador, fecha y hora de la prueba, el tiempo transcurrido durante la implementación de la codificación manual y el tiempo transcurrido de la codificación generada usando nuestro programa, estos iban de acuerdo al número de prueba que se aplicaba, como también se consideró un campo de observación para que el evaluador pueda anotar a que tabla de base de datos corresponde los datos y poder colocar algún comentario vital para el análisis de estos datos (Anexo - 3).
- Para el indicador de tiempo de corrección de errores en la codificación, se consideró tomar los siguientes datos: nombre del evaluador, fecha y hora de la prueba, el tiempo invertido para solucionar los errores en la codificación manual y también el tiempo invertido en solucionar errores del código generado por nuestro programa, esto se llenaba según el número de prueba realizada, también se consideró colocar la columna de observaciones para que el evaluador coloque el nombre de la tabla de base de datos a la cual correspondía la prueba y colocar algún comentario que considerase vital para el análisis de datos (Anexo - 4).

Los instrumentos a aplicar fueron basados de acuerdo a las dimensiones establecidas en el proyecto y así poder realizar un análisis diferencial de los datos que se obtienen sin nuestro programa, con los datos que se obtienen con él.

Finalmente, se realizó una matriz de validación (Anexo - 6) para poder aplicar estos instrumentos propuestos, los cuales fueron analizados y evaluados previamente por un experto, quien es Magister en Sistemas y conocedor de los temas de programación, etapas de desarrollo del software, y generadores de código de programación.

Continuando con los pasos para la recolección de datos se describe la forma que se siguió de manera estructurada y secuencial, para obtener los datos requeridos del Pre-Test, según nuestros indicadores:

- Para obtener los resultados de los tiempos dedicados de codificación, se consideró el uso de la Ficha de observación N° 2 (Anexo - 4), en la cual se debía realizar el desarrollo de la codificación manual en la etapa de implementación de un software, donde se logró tomar el tiempo invertido de dicha codificación, para luego ser comparado con el tiempo invertido que se tomó al momento de realizar la generación de código con nuestro programa.

Para el procesamiento de la información con respecto al tiempo se tuvo que realizar una conversión de los tiempos tomados en horas y segundos a minutos.

Los cálculos de conversión se realizaron de la siguiente manera:

**Se tiene que:**

$$TM_i = (HRP_i * 60) + \left(\frac{SRP_i}{60}\right) + MRP$$

Ecuación 1. Conversión a minutos

**Dónde:**

**$HRP_i$**  = Cantidad restante de horas de la prueba  $i$  – esima

**$SRP_i$**  = Cantidad de segundos restantes de la prueba  $i$  – esima

**$MRP_i$**  = Cantidad de minutos restantes de la prueba  $i$  – esima

**$TM_i$**  = Valor total en minutos del tiempo de la prueba  $i$  – esima

A continuación, en la Tabla 4 se muestran los resultados de la conversión a minutos para los tiempos de codificación, en donde se considera solo 2 decimales.

Tabla 4

*Resultados de la conversión a minutos de los tiempos obtenidos al implementar código de programación manual.*

N°	Nombre de Tabla de la BD	Tiempo Pre-Test	Tiempo Pre-Test (Minutos)
1	Distributor	5 40' 20''	340.33
2	Product	5 33' 40''	333.66
3	Warehouse	5 15' 15''	315.25
4	Address	5 17' 33''	317.55
5	Coupon	4 58' 37''	298.61
6	SubCategory	4 50' 58''	290.96
7	Category	4 41' 30''	281.5
8	ProductComment	4 45' 16''	285.26
<b>SUMATORIA</b>			<b>2,463.12</b>

Fuente: Creación propia.

- Luego de ello, se realizó la recolección de la información de estándares de nomenclatura, para ello se realizó las pruebas requeridas de análisis y revisión del cumplimiento de estándares, en el cual se pudo aplicar la técnica de observación para encontrar la cantidad exacta de cumplimiento e incumplimiento de los estándares de nomenclatura al momento de realizar código manual y/o generación de código automático, para ello se aplicó el instrumento llamado Ficha de observación N° 1 (Anexo - 3).

Los valores se calcularon en base a las respuestas proporcionadas por las fichas de observación aplicadas al código implementado para la evaluación del cumplimiento de estándares de nomenclatura, luego se calculó el total de palabras analizadas, a partir del total de las palabras que cumplen con los estándares y el total de las palabras que no lo cumplen, como también se calculó el porcentaje de cumplimiento por cada prueba realizada a cada tabla de base de datos, como se detalla a continuación:

**Se tiene que:**

$$TP_i = TPC_i + TPI_i$$

Ecuación 2. Sumatoria de palabras

**Dónde:**

$TPC_i$  = Total de palabras correctas en la prueba  $i$  – esima

$TPI_i$  = Total de palabras incorrectas en la prueba  $i$  – esima

El cálculo del porcentaje de cumplimiento sería:

$$PC_i = \frac{TPC_i}{TP_i} * 100$$

Ecuación 3. Porcentaje de cumplimiento de estándares de nomenclatura.

**Dónde:**

$PC_i$  = Porcentaje de cumplimiento de la prueba  $i$  – esima

En la Tabla 5 se muestra los resultados obtenidos luego de aplicar el instrumento asignado para realizar los cálculos mencionados, en los cuales solo se considera 2 decimales en la cifra de porcentaje de cumplimiento de acuerdo al uso correcto de los estándares de nomenclatura en la codificación realizada manualmente.

Tabla 5

*Detalle de la cantidad correcta e incorrecta y porcentaje de cumplimiento del uso de estándares de nomenclatura en el código de programación manual.*

N°	Nombre de Tabla de la BD	Total	Total	Total	% Cumplimiento Pre-Test
		Cantidad Correcta	Cantidad Incorrecta		
1	Distributor	17	6	23	73.91
2	Product	16	4	20	80
3	Warehouse	17	5	22	77.27
4	Address	15	4	19	78.94
5	Coupon	13	4	17	76.47
6	SubCategory	13	2	15	86.66
7	Category	15	0	15	100
8	ProductComment	14	2	16	87.5
<b>SUMATORIA</b>		120	27	147	

Fuente: Creación propia.

- Posteriormente, también se recolectó la información de tiempos de corrección de errores encontrados en la codificación, el cual fue realizado por uno de los autores que laboraba en la empresa. Estos tiempos fueron tomados luego de finalizar la implementación del código manual, hasta el momento que este código fue ejecutado y probado de manera correcta, se tomó el tiempo invertido para poder hallar, analizar y solucionar los errores encontrados en la codificación manual realizada, si este había sido implementado con algún error de código. Cabe indicar que para ello se aplicó el instrumento llamado Ficha de observación N° 3 (Anexo - 5).

Asimismo, como los datos obtenidos fueron basados en tiempo, se aplicó la conversión propuesta de horas y segundos a minutos.

En la Tabla 6, se muestran los resultados obtenidos de la conversión de los tiempos de corrección de errores de la codificación, en donde se considera solo 2 decimales.

Tabla 6

*Resultados de la conversión de los tiempos invertidos al solucionar errores en el código de programación realizado manualmente.*

N°	Nombre de Tabla de la BD	Tiempo Pre-Test	Tiempo Pre-Test (Minutos)
1	Distributor	37' 10''	37.16
2	Product	20' 37''	20.61
3	Warehouse	25' 23''	25.38
4	Address	17' 16''	17.26
5	Coupon	12' 37''	12.61
6	SubCategory	9' 41''	9.68
7	Category	6' 38''	6.63
8	ProductComment	8' 5''	8.08
<b>SUMATORIA</b>			<b>137.41</b>

Fuente: Creación propia.

Por otro lado, para comenzar con el desarrollo de nuestro generador de código, fue necesario elegir una de las metodologías para proyectos de software para el análisis, control y documentación de los avances del desarrollo de nuestro programa.

A continuación, presentamos un cuadro comparativo de algunas metodologías, el cual nos ayudó a poder tomar la mejor opción con respecto a la metodología a usar.

Tabla 7

*Cuadro comparativo con respecto a metodologías de desarrollo de proyectos de software.*

<b>CRITERIOS</b>	<b>RUP</b>	<b>SCRUM</b>	<b>XP</b>
- Los autores tienen conocimiento en la metodología.	SI	SI	NO
- Se aplica a proyectos de mediano o corto plazo.	NO	SI	SI
- Se puede elegir la secuencia y prioridad de las actividades del cronograma.	NO	SI	NO
- La programación se puede realizar por equipos.	SI	SI	NO
- Es usado por la rapidez de presentación de avances.	NO	SI	SI

Fuente: Creación propia.

En base a los criterios mencionados, optamos por la metodología ágil SCRUM, ya que nos permite realizar avances por tiempos cortos, priorizar a criterio propio las actividades a realizar en el proyecto y cumple con cada uno de los criterios en mención que nos ayudaron en el manejo de entregas del proyecto.

Una vez elegida la metodología, procedimos a realizar un análisis de los requerimientos y actividades que se necesitaban para poder cumplir con nuestro objetivo de implementar el generador de código de programación, a partir de ello, se realizó la documentación de la implementación del generador (Anexo - 8), en primer lugar, se realizó el Product Backlog, donde se listaron todos los requerimientos y actividades necesarias que se tenían que considerar para la implementación de nuestro programa.

Posteriormente, estas actividades se dividieron en Sprints Backlogs, los cuales se realizaron para especificar los avances y entregables del producto, y así poco a poco ir implementando el sistema generador de código de programación que se propuso.

Por otro lado, al haber obtenido nuestro programa, se realizaron pruebas iniciales y el análisis de retroalimentación para ir mejorando el software, y así poder brindar un producto adecuado para el uso de los usuarios y obtener mejores resultados de acuerdo a nuestros indicadores expuestos.

Cabe mencionar que se realizó un análisis de casos de uso para nuestro producto a desarrollar, como también se realizó el diseño de interfaces de nuestro programa haciendo uso de la herramienta software Balsamic Mockups, para luego proceder a implementar cada interface diseñada.

Luego de finalizar el desarrollo de nuestro software generador de código de programación, se procedió a realizar las pruebas y levantamiento del Post-Test en base a los siguientes pasos:

- En primer lugar, se tomó el tiempo invertido en la generación de código de programación automatizado en base a la Ficha de Observación N°1, la cual ya tenía los datos del Pre-Test, para ello se realizó la generación de un CRUD por tabla de base de datos en prueba.

En la Tabla 8 se muestra los resultados de la conversión a minutos de los tiempos obtenidos para la implementación de codificación, luego de aplicar el generador de código. En la conversión solo se consideró 2 decimales.

Tabla 8

*Resultados de la conversión a minutos de los tiempos de implementación de código en el Post-Test.*

N°	Nombre de Tabla de la BD	Tiempo Post-Test	Tiempo Post-Test (Minutos)
1	Distributor	6' 20''	6.33
2	Product	6'	6
3	Warehouse	5' 40''	5.66
4	Address	5' 27''	5.45
5	Coupon	5'	5
6	SubCategory	5' 13''	5.21
7	Category	4' 45''	4.45
8	ProductComment	4' 56''	4.93
<b>SUMATORIA</b>			43.03

Fuente: Creación propia.

- Siguiendo con la recolección de datos Post-Test, se realizó el análisis y evaluación de los estándares de nomenclatura con respecto al código obtenido por nuestro programa.

En la Tabla 9 se muestra los resultados obtenidos de los estándares de nomenclatura luego de aplicar la propuesta del generador de código de programación, para así realizar los cálculos realizados con los datos del Pre-Test, los cuales eran calcular la cantidad total de palabras y calcular el porcentaje de cumplimiento por cada prueba realizada, para ello se aplicó las mismas ecuaciones aplicadas anteriormente.



Tabla 9

*Detalle de la cantidad correcta e incorrecta y porcentaje de cumplimiento del uso de estándares de nomenclatura en el Post-Test.*

N°	Nombre de Tabla de la BD	Total Cantidad Correcta	Total Cantidad Incorrecta	Total	% Cumplimiento Post-Test
1	Distributor	23	0	23	100
2	Product	20	0	20	100
3	Warehouse	22	0	22	100
4	Address	19	0	19	100
5	Coupon	17	0	17	100
6	SubCategory	15	0	15	100
7	Category	15	0	15	100
8	ProductComment	16	0	16	100
<b>SUMATORIA</b>		147	0	147	

Fuente: Creación propia.

- De igual manera se procedió con la recolección de los tiempos de corrección de errores encontrados en la codificación realizada con el generador de código de programación.

En la Tabla 10 se muestra los resultados de la conversión de los tiempos obtenidos luego del análisis y solución de errores en el código generado.

**Tabla 10**

*Resultados de la conversión a minutos de los tiempos de corrección de errores en el Post-Test.*

N°	Nombre de Tabla de la BD	Tiempo Post-Test	Tiempo Post-Test (Minutos)
1	Distributor	0	0
2	Product	0	0
3	Warehouse	0	0
4	Address	0	0
5	Coupon	0	0
6	SubCategory	0	0
7	Category	0	0
8	ProductComment	0	0
<b>SUMATORIA</b>			0

Fuente: Creación propia.

Por otro lado, para poder obtener los resultados diferenciales con respecto al Pre-Test y Post-Test de la dimensión de estándares de nomenclatura y la dimensión de codificación, fue necesario realizar una evaluación en base a eficacia, para ello se usó las siguientes ecuaciones:

- Cálculo de eficacia con respecto a la dimensión de estándares de nomenclatura .

**Se tiene que:**

$$REE_i = \frac{PO_i}{PE} * 100\%$$

Ecuación 4. Porcentaje de eficacia del uso de estándares de nomenclatura

**Dónde:**

**PO** = Porcentaje obtenido de la prueba i – esima

**PE** = Porcentaje esperado

**REE<sub>i</sub>** = Resultado de eficacia de estandares en la prueba i – esima

**Nota:** La variable de porcentaje esperado fue tomada en base a los criterios de auditoria (Anexo - 7) de la empresa Inka Mall S.A.C, en donde se interpretó el valor del porcentaje de cumplimiento que la empresa desea obtener, por ello el valor de **PE = 100%**.

- Cálculo de eficacia con respecto a la dimensión de codificación:

**Se tiene que:**

$$RET_i = \frac{TE}{TIC_i + TSE_i} * 100\%$$

Ecuación 5. Porcentaje de eficacia de codificación

**Dónde:**

**TIC** = Tiempo de implementacion de codigo de le prueba i – esima

**TSE** = Tiempo de Solucion de errores de la prueba i – esima

**TE** = Tiempo esperado

**RET<sub>i</sub>** = Resultado de eficacia de tiempos en la prueba i – esima

**Nota:** La variable de tiempo esperado, tuvo el valor obtenido a partir del cálculo del promedio de los datos que se reportan en los tickets o historial de

tiempos estimados de desarrollo de la empresa (Anexo - 8), de dichos tiempos ingresados en los tickets, se nos indicó por parte de la empresa que solo se usa el 70% para la implementación del Back End.

Por ello, primero se calculó el porcentaje a usar de los tiempos estimados que la empresa nos brindó, para ello se aplicó lo siguiente:

**Se tiene que:**

$$TBE_i = TET_i * 0.7$$

Ecuación 6. Porcentaje de tiempos estimados para Back End

**Dónde:**

$TET_i$  = Tiempo estimado del ticket  $i$  – esima

$TBE_i$  = Tiempo estimado para el Back End del ticket  $i$  – esima

Luego de ello, a los tiempos obtenidos se aplicó la conversión a minutos, para así poder procesar la información obteniendo el promedio de los tiempos del historial, aplicando lo siguiente:

**Se tiene que:**

$$\bar{X} = \frac{\sum_{i=1}^n TBE_i}{n}$$

Ecuación 7. Promedio del historial de tiempos estimados de codificación

**Dónde:**

$TBE_i$  = Tiempos estimados para el Back End del ticket  $i$  – esima

$\bar{X}$  = Promedio de tiempo estimado

Por ello el valor de  $TE = \bar{X}$ .

- Finalmente, también se realizó el cálculo del promedio general de eficacia con los resultados obtenidos en el análisis de datos de nuestras dimensiones. Este se calculó haciendo uso de la siguiente ecuación:

**Se tiene que:**

$$\bar{X}_f = \frac{REE_i + RET_i}{2}$$

Ecuación 8. Porcentaje de eficacia de la etapa de implementación de un software

**Dónde:**

$REE_i$  = Resultado de eficacia de estandares en la prueba i – esima

$RET_i$  = Resultado de eficacia de tiempos en la prueba i – esima

$\bar{X}_f$  = Promedio final en base a eficacia

### CAPÍTULO III. RESULTADOS

A continuación, se muestran e interpretan los resultados de la aplicación del generador de código de programación en la etapa de implementación de un software en la empresa Inka Mall S.A.C. durante el año 2019. Los datos fueron recolectados por medio de nuestras fichas de observación y posteriormente aplicando las ecuaciones y pasos descritos en la sección de procedimiento.

Tabla 11

*Contraste de resultados de pruebas Pre y Post Test de la etapa de implementación de un software.*

	Pre -Test	Post - Test	Diferencia (d)	$d_i - \bar{d}$	$(d_i - \bar{d})^2$
1	68.25	1916.12	-3669.65	696.17	484656.15
2	86.58	2018.75	-3844.35	521.47	271933.57
3	73.32	2137.02	-4104.67	261.15	68200.63
4	74.75	2217.43	-4264.3	101.52	10306.82
5	76.19	2412.50	-4649.09	-283.27	80240.48
6	82.62	2317.28	-4455.97	-90.15	8126.57
7	91.00	2704.50	-5227	-861.18	741626.69
8	84.02	2446.05	-4711.55	-345.73	119527.50
<b>SUMATORIA</b>	612.67	35539.25	-34926.58		1784618.40
<b>MEDIA</b>	76.58	4442.41	-4365.82	<b>Varianza (s)</b>	223077.30

Fuente: Creación propia.

#### Hipótesis Estadísticas:

**Hipótesis H<sub>0</sub>:** Los resultados obtenidos en la etapa de implementación de un software sin el generador es mayor a los resultados obtenidos en la etapa de implementación de un software con el generador propuesto.

$$H_0: RE_f > 0$$

**Hipótesis H<sub>a</sub>:** Los resultados obtenidos en la etapa de implementación de un software sin el generador es menor a los resultados obtenidos en la etapa de implementación de un software con el generador propuesto.

$$H_a: RE_f < 0$$

Los datos fueron procesados y analizados haciendo uso de la prueba t de student en el software estadístico XLSTAT en conjunto con Excel 2016.

A continuación, presentaremos los resultados obtenidos por el software estadístico:

Tabla 12

*Datos estadísticos descriptivos para la prueba t en el cumplimiento de estándares de nomenclatura.*

Variable	Observaciones	Mínimo	Máximo	Media	Desv. típica
Pre -Test	8	68.250	91.000	79.591	7.660
Post - Test	8	1916.120	2704.500	2271.206	253.861

Fuente: Creación propia.

Prueba t para dos muestras independientes / Prueba bilateral:

El software consideró como intervalo de confianza para la diferencia entre las medias al 95%: [-2384.205; -1999.025]

Los resultados obtenidos con el software se pueden apreciar en la Tabla 13.

Tabla 13

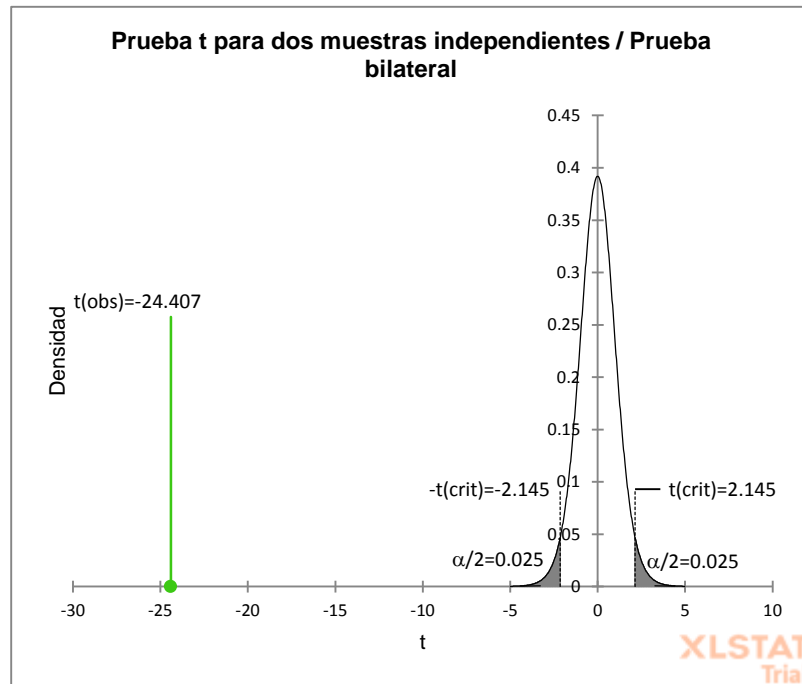
*Resultados de XLSTAT en la prueba t para las muestras Pre y Post Test de la etapa de implementación de un software.*

<b>Diferencia</b>	-2191.615
<b>t (Valor observado)</b>	-24.407
<b> t  (Valor crítico)</b>	2.145
<b>GL</b>	14
<b>valor-p (bilateral)</b>	< 0.0001
<b>Alfa</b>	0.05

Fuente: Creación propia.

Interpretación del Software:

- Puesto que el valor-p computado es menor que el nivel de significación  $\alpha=0.05$ , se debe rechazar la hipótesis nula  $H_0$ , y aceptar la hipótesis alternativa  $H_a$ .

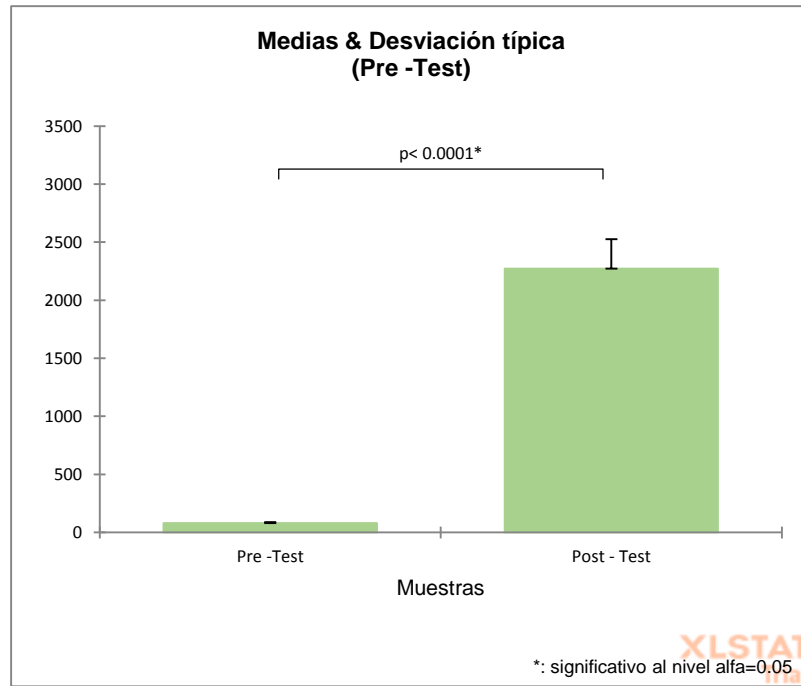


**Figura 3.** Evaluación de la zona de aceptación o rechazo del cumplimiento de la etapa de implementación de un software. Fuente: Creación propia.

Interpretación propia:

- Puesto que  $t = -24.407 < |t| = 2.145$ , estando este valor dentro de la región de rechazo, se concluye que  $RE_f < 0$ , se rechaza  $H_0$  y  $H_a$  es aceptada, por lo tanto, se prueba la validez de la hipótesis con un nivel de error alfa = 0.05, siendo que la implementación del generador propuesto influye de manera eficaz en la etapa de implementación de un software.

De igual manera el software nos brindó el gráfico de barras establecido para dicha prueba realizada, donde muestra la comparación de los porcentajes del Pre y Post Test.



**Figura 4.** Gráfico de barras de la comparación de las muestras del Pre-Test vs Post-Test de la etapa de implementación de un software. Fuente: Creación propia.

Tabla 14

*Comparación de resultados del PreTest vs PostTest de la influencia reflejada en la etapa de implementación de un software.*

Pre-Test	Post-Test	AUMENTO
Porcentaje	Porcentaje	Porcentaje
76.58%	4442.41%	4365.82%

Fuente: Creación propia.

Se puede observar en la Tabla 14 que el porcentaje de eficacia en la etapa de implementación de un software al ser codificado de forma manual (Pre-Test) es de 76.58% y con el generador de código (Post-Test) es del 4442.41%, lo que representa un incremento del 4365.82% de eficacia en los resultados.

Para poder realizar el análisis general y obtener el resultado mostrado, se realizó el siguiente análisis y procesamiento de datos:



### **Análisis de la dimensión de estándares de nomenclatura.**

A continuación, se muestran e interpretan los resultados de la aplicación del generador de código de programación en la dimensión de estándares de nomenclatura de la etapa de implementación de un software en la empresa Inka Mall S.A.C. durante el mes de mayo del 2019.

Podemos observar en la Tabla 15 la contrastación de los resultados de las pruebas realizadas Pre y Post Test para evaluar la eficacia de la dimensión de estándares de nomenclatura.

Tabla 15

*Contraste de resultados de pruebas Pre y Post Test de la dimensión de estándares de nomenclatura.*

	<b>Pre -Test</b>	<b>Post - Test</b>	<b>Diferencia (d)</b>	$d_i - \bar{d}$	$(d_i - \bar{d})^2$
1	73.91	100	-26.09	-8.68	75.41
2	80	100	-20	-2.59	6.73
3	77.27	100	-22.73	-5.32	28.34
4	78.94	100	-21.06	-3.65	13.35
5	76.47	100	-23.53	-6.12	37.50
6	86.66	100	-13.34	4.07	16.53
7	100	100	0	17.41	302.98
8	87.5	100	-12.5	4.91	24.07
<b>MEDIA</b>	82.59	100	-17.41	<b>Varianza (s)</b>	63.11

Fuente: Creación propia.

### **Hipótesis Estadísticas:**

**Hipótesis H<sub>0</sub>:** El cumplimiento de estándares de nomenclatura sin el generador es mayor al cumplimiento de estándares de nomenclatura con el generador propuesto.

$$H_0: RE_1 > 0$$

**Hipótesis H<sub>a</sub>:** El cumplimiento de los estándares de nomenclatura sin el generador es menor al cumplimiento de los estándares de nomenclatura con el generador.

$$H_a: RE_1 < 0$$

Los datos fueron procesados y analizados haciendo uso de la prueba t de student en el software estadístico XLSTAT en conjunto con Excel 2016.

A continuación, presentaremos los resultados obtenidos por el software estadístico:

Tabla 16

*Datos estadísticos descriptivos para la prueba t en el cumplimiento de estándares de nomenclatura.*

Variable	Observaciones	Mínimo	Máximo	Media	Desv. típica
Pre -Test	8	73.910	100.000	82.594	8.493
Post - Test	8	100.000	100.000	100.000	0.000

Fuente: Creación propia.

Prueba t para dos muestras independientes / Prueba bilateral:

El software consideró como intervalo de confianza para la diferencia entre las medias al

95%: [-23.846; -10.966]

Los resultados obtenidos con el software se pueden apreciar en la Tabla 17.

Tabla 17

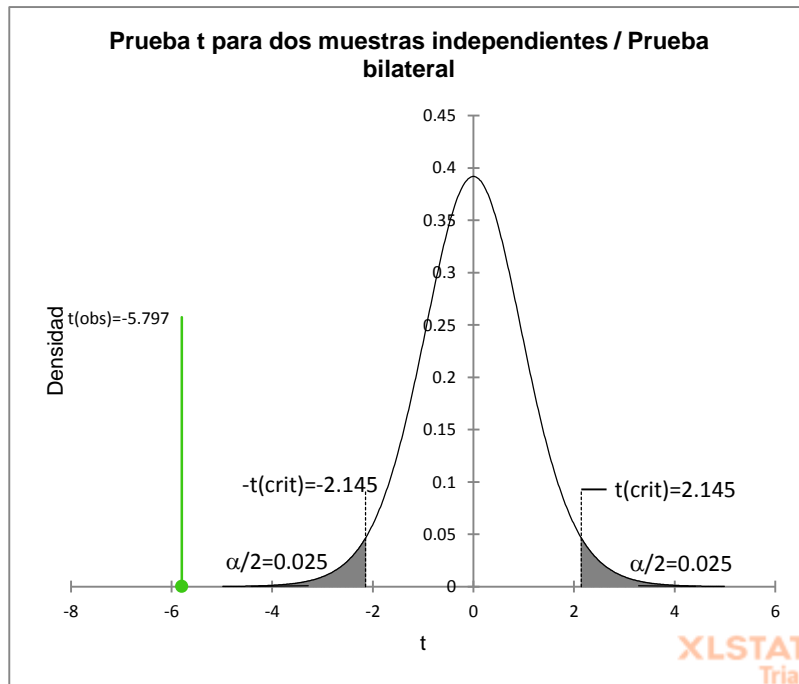
Resultados de XLSTAT en la prueba t para las muestras Pre y Post Test del cumplimiento de estándares de nomenclatura.

<b>Diferencia</b>	-17.406
<b>t (Valor observado)</b>	-5.797
<b> t  (Valor crítico)</b>	2.145
<b>GL</b>	14
<b>valor-p (bilateral)</b>	< 0.0001
<b>Alfa</b>	0.05

Fuente: Creación propia.

Interpretación del Software:

- Puesto que el valor-p computado es menor que el nivel de significación  $\alpha=0.05$ , se debe rechazar la hipótesis nula  $H_0$ , y aceptar la hipótesis alternativa  $H_a$ .

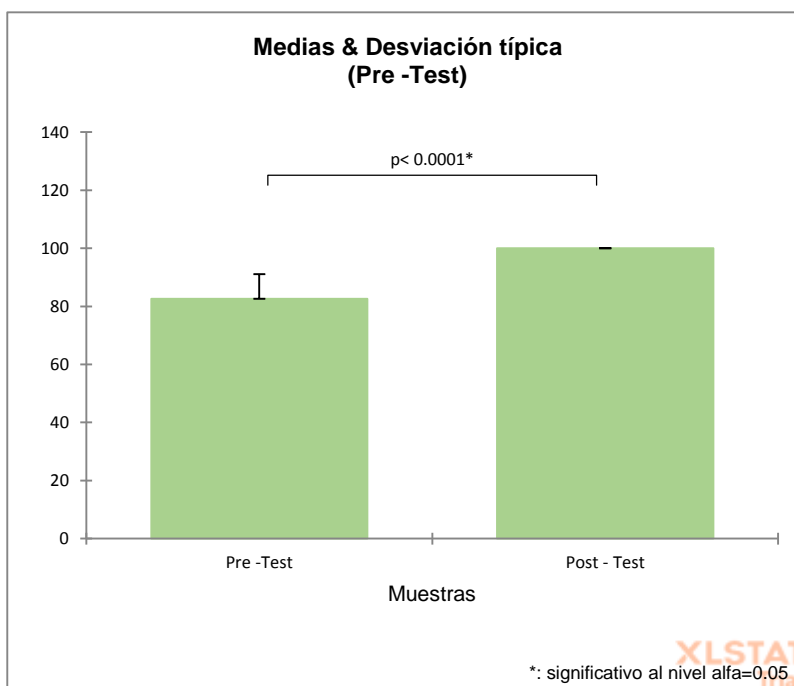


**Figura 5.** Evaluación de la zona de aceptación o rechazo del cumplimiento de estándares de nomenclatura.

Interpretación propia:

- Puesto que  $t = -5.797 < |t| = 2.145$ , estando este valor dentro de la región de rechazo, se concluye que  $RE_1 < 0$ , se rechaza  $H_0$  y  $H_a$  es aceptada, por lo tanto, se prueba la validez de la hipótesis con un nivel de error alfa = 0.05, siendo que la implementación del generador propuesto mejoró la eficacia de los estándares de nomenclatura.

De igual manera el software nos brindó el gráfico de barras establecido para dicha prueba realizada, donde muestra la comparación de los porcentajes del Pre y Post Test.



**Figura 6.** Gráfico de barras de la comparación de las muestras del Pre-Test vs Post-Test de estándares de nomenclatura

Tabla 18

*Comparación de resultados del PreTest vs PostTest de estándares de nomenclatura.*

Pre-Test	Post-Test	AUMENTO
Porcentaje	Porcentaje	Porcentaje
82.59%	100%	17.41%

Fuente: Creación propia.

Se puede observar en la Tabla 18, que el indicador de cumplimiento de estándares de nomenclatura en la implementación de código manual (Pre-Test) es de 82.59% y con el generador de código (Post-Test) es del 100%, lo que representa un incremento del 17.41%.

### **Análisis de la dimensión de codificación.**

A continuación, se muestran e interpretan los resultados de la aplicación del generador de código de programación en la dimensión de codificación en la etapa de implementación de un software en la empresa Inka Mall S.A.C. durante el mes de mayo del 2019.

Podemos observar en la Tabla 19 la contrastación de los resultados de las pruebas realizadas Pre y Post Test para evaluar la eficacia de la dimensión de codificación.

Tabla 19

*Contraste de resultados de pruebas Pre y Post Test de la dimensión de codificación.*

	Pre -Test	Post - Test	Diferencia (d)	$d_i - \bar{d}$	$(d_i - \bar{d})^2$
1	62.58	3732.23	-3669.65	696.17	484656.15
2	93.15	3937.5	-3844.35	521.47	271933.57
3	69.36	4174.03	-4104.67	261.15	68200.63
4	70.56	4334.86	-4264.3	101.52	10306.82
5	75.91	4725	-4649.09	-283.27	80240.48
6	78.58	4534.55	-4455.97	-90.15	8126.57
7	81.99	5308.99	-5227	-861.18	741626.69
8	80.54	4792.09	-4711.55	-345.73	119527.50
<b>SUMATORIA</b>	612.67	35539.25	-34926.58		1784618.40
<b>MEDIA</b>	76.58	4442.41	-4365.82	<b>Varianza (s)</b>	223077.30

Fuente: Creación propia.

### Hipótesis Estadísticas:

**Hipótesis H<sub>0</sub>:** La eficacia de la codificación sin el generador es mayor a la eficacia de la codificación con el generador propuesto.

$$H_0: RE_2 > 0$$

**Hipótesis H<sub>a</sub>:** La eficacia de la codificación sin el generador es menor a la eficacia de la codificación con el generador propuesto.

$$H_a: RE_2 < 0$$

Los datos fueron procesados y analizados haciendo uso de la prueba t de student en el software estadístico XLSTAT, en conjunto con Excel 2016.

A continuación, presentaremos los resultados obtenidos por el software estadístico:

Tabla 20

Datos estadísticos descriptivos para la prueba t para los tiempos de implementación de código

Variable	Observaciones	Mínimo	Máximo	Media	Desv. típica
Pre -Test	8	62.580	93.150	76.584	9.322
Post - Test	8	3732.230	5308.990	4442.406	507.722

Fuente: Creación propia.

Prueba t para dos muestras independientes / Prueba bilateral:

El software consideró como intervalo de confianza para la diferencia entre las medias al

95%: [-4750.891; -3980.754]

Los resultados obtenidos con el software se pueden apreciar en la Tabla 21.

Tabla 21

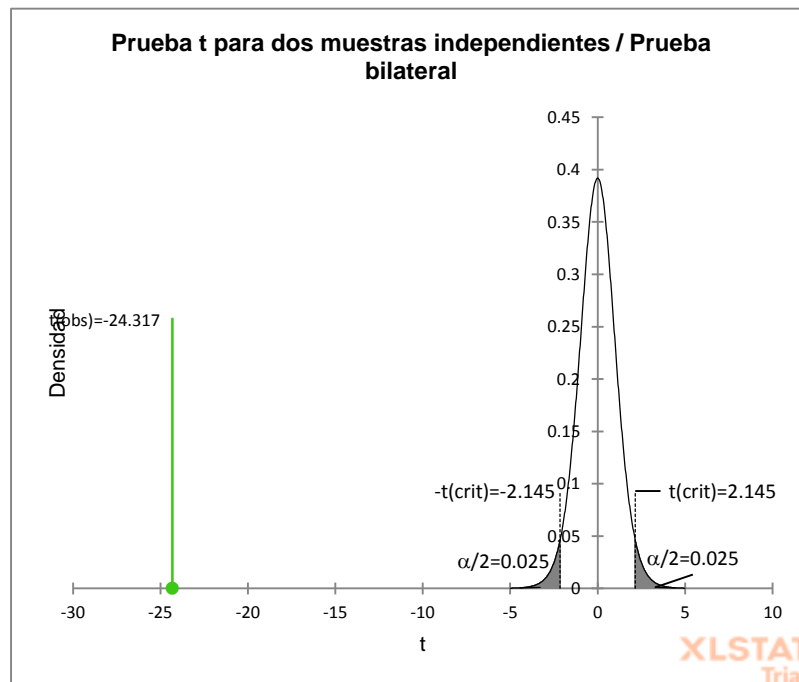
*Resultados de XLSTAT en la prueba t para las muestras Pre y Post Test del tiempo de implementación de código.*

<b>Diferencia</b>	-4365.823
<b>t (Valor observado)</b>	-24.317
<b> t  (Valor crítico)</b>	2.145
<b>GL</b>	14
<b>valor-p (bilateral)</b>	< 0.0001
<b>Alfa</b>	0.05

Fuente: Creación propia.

Interpretación del Software:

- Puesto que el valor-p computado es menor que el nivel de significación  $\alpha=0.05$ , se debe rechazar la hipótesis nula  $H_0$ , y aceptar la hipótesis alternativa  $H_a$ .

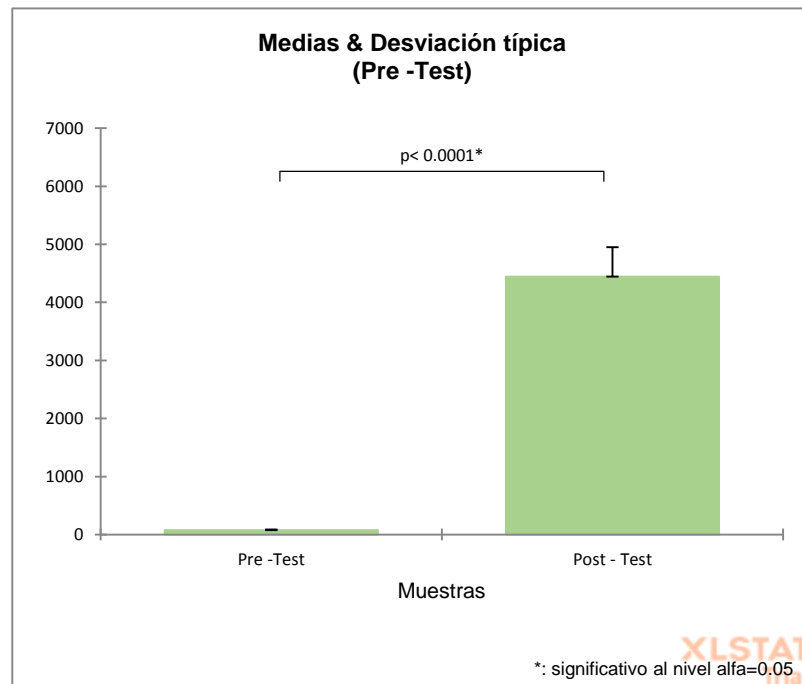


**Figura 7.** Evaluación de la zona de aceptación o rechazo de la eficacia de la codificación.

Interpretación propia:

- Puesto que  $t = -24.317 < |t| = 2.145$ , estando este valor dentro de la región de rechazo, se concluye que  $RE_2 < 0$ , se rechaza  $H_0$  y  $H_a$  es aceptada, por lo tanto, se prueba la validez de la hipótesis con un nivel de error  $\alpha = 0.05$ , siendo que la implementación del generador propuesto mejoró la eficacia de la codificación.

De igual manera el software nos brindó el gráfico de barras establecido para dicha prueba realizada, donde muestra la comparación de los porcentajes del Pre y Post Test.



**Figura 8.** Gráfico de barras de la comparación de las muestras del Pre-Test vs Post-Test de la eficacia en la codificación.

Tabla 22

*Comparación de resultados del PreTest vs PostTest del cumplimiento de estándares de nomenclatura.*

<b>Pre-Test</b>	<b>Post-Test</b>	<b>AUMENTO</b>
Porcentaje	Porcentaje	Porcentaje
76.58	4442.41	4365.83

Se puede observar en la Tabla 22 que la eficacia de la codificación manual (Pre-Test) es de 76.58% y con el generador de código (Post-Test) es 4442.41%, lo que representa un aumento de 4365.83%.



## CAPÍTULO IV. DISCUSIÓN Y CONCLUSIONES

### 4.1 Discusión

En los resultados mostrados en la tabla 18, se observó que el valor del indicador promedio para la etapa de implementación de un software en la dimensión de estándares de nomenclatura fue 82.59% antes de la implementación del generador de código de programación, mientras que luego de ser implementado, el valor fue 100%, esto demuestra una mejora del 17.41%, lo necesario para cumpliendo con el uso de estándares que la empresa demanda al 100%; de acuerdo a ello, la presente investigación cumple con una de las características que menciona Huari (2020), en su investigación “Revisión Sistemática Sobre Generadores De Código Fuente Y Patrones De Arquitectura”, dónde indica que de todos los generadores de código, solo el 27.3% cumple con la generación de código teniendo en consideración la estandarización de este; asimismo, se puede deducir que esto se debe a que muchos de los generadores creados son enfocados más al mejoramiento de re utilidad con 54.5% y reducción de costos y tiempos con 51.5%, como características destacadas según su tabla de resultados que definen la importancia de los generadores de código. De la tabla N° 19, se aprecia el valor del indicador promedio para la etapa de implementación de un software que fue 76.58% antes de la implementación del generador de código de programación. Posteriormente, después de haberse implementado, el valor obtenido fue 4442.41%, con resultados de generación de código completo y correcto al 100%, estos al ser contrastados con los resultados de la investigación de Rincón, Aguilar e Hidrobo (2010), titulada “Generación Automática de Código a Partir de Máquinas de Estado Finito”, se observó que existe una diferencia del 20 %, ya que en sus pruebas obtuvieron el 80% de código generado de forma completa, por ello, se puede deducir que dicha investigación al hacer uso

de elementos UML y clases relacionadas para la generación de código fueron mal definidos y el software no pudo sintetizar de manera completa el código, mientras que en la presente investigación se usaron plantillas estructuradas para que al ejecutarse se obtenga un código completo y correcto.

Finalmente, en los resultados mostrados en la tabla 22, observamos que el valor promedio para la etapa de implementación de un software en la dimensión de codificación fue 76.58% antes de la implementación del generador de código de programación; sin embargo, luego de ser implementado, el valor fue 4442.41%. Esto demuestra una mejora del 4365.82% con respecto a eficacia del generador al reducir un proceso desarrollado en horas a minutos.

## 4.2 Conclusiones

Al finalizar el presente trabajo de investigación, se llegaron a las siguientes conclusiones:

- Con base en nuestro objetivo general, se pudo demostrar que el generador de código de programación tuvo una influencia positiva sobre la etapa de implementación de un software, demostrando como resultados una gran mejoría.
- Se logró demostrar la influencia de nuestro generador en la dimensión de estándares de nomenclatura de la etapa de implementación de un software, mostrando los resultados obtenidos en el análisis de datos, los cuales demuestran una mejora en la implementación de software.
- Se logró demostrar que el uso de nuestro generador es eficaz e influye en la dimensión de codificación de la etapa de implementación de un software, donde se reflejó una gran mejora con respecto a la reducción de tiempo que se suele invertir en la codificación de un software.

### 4.3 Recomendaciones

En un futuro se pueda incluir la generación de código Front-End de un formato estándar para así obtener un producto más completo y más eficaz.

Se podría tomar en cuenta la generación de código de programación en base a la conexión de otros motores de base de datos y no solamente uno.

Se puede realizar un análisis sistemático de control y mejoramiento del generador en un futuro para poder implementar nuevas funcionalidades dentro del software y de acuerdo a las necesidades de la empresa.

## REFERENCIAS

- Acedo, J. (2017). Estándares de nomenclatura: Snake Case, Kebab Case, Camel Case [Mensaje en un blog]. Apuntes de Programación. Recuperado de <http://programacion.jias.es/2017/09/estandares-de-nomenclatura-snake-case-kebab-case-camel-case/>.
- Adárraga Mejía, C. & Oliveros Villanueva, C. (2014). Modelado UML del generador de código de aplicaciones web TGENP (Doctoral dissertation, Universidad del Magdalena). Recuperado de <https://core.ac.uk/download/pdf/270125233.pdf>
- Alonso, D., Pastor, J. Á., Sánchez, P., Álvarez, B., & Chicote, C. V. (2012). Generación automática de software para sistemas de tiempo real: Un enfoque basado en componentes, modelos y frameworks. *Revista Iberoamericana de Automática e Informática industrial*, 9(2), 170-181.
- Apiumhub (s.f.). Transformación Ágil: Pasos, Estadísticas Y Un Ejemplo Práctico [Mensaje en un blog]. Recuperado de <https://apiumhub.com/es/tech-blog-barcelona/transformacion-agil-pasos-estadisticas/>.
- Asociación Española de Programadores Informáticos. (s.f.). Las cinco cosas más frustrantes de la profesión del programador [Mensaje en un blog]. Recuperado de <https://asociacionaepi.es/las-cinco-cosas-mas-frustrantes-de-la-profesion-de-programador/>.
- Autodesk (2018). Conexión con una base de datos ODBC [Mensaje en un blog]. Recuperado de <https://knowledge.autodesk.com/es/support/infrastructure-map-server/learn-explore/caas/CloudHelp/cloudhelp/2017/ESP/MapServer-Help/files/GUID-4BEDC497-7AF3-44A3-9B46-3D9393B6E81F-htm.html>.
- Baskaran, M. M., Ramanujam, J., & Sadayappan, P. (2010, March). Automatic C-to-CUDA code generation for affine programs. In *International Conference on Compiler Construction*(pp. 244-263). Springer, Berlin, Heidelberg.
- Becerra, J. C. (2019). Generador de código de funcionalidades tipo crud en la mantenibilidad de software aplicado a sistemas de información empresariales. Repositorio de la Universidad Privada del Norte. Recuperado de <http://hdl.handle.net/11537/23234>

- Benjamín. (2008). Estilo de programación y convención de nombres II [Mensaje en un blog]. Codigolinea. Recuperado de <http://codigolinea.com/2008/05/25/estilo-de-programacion-y-convencion-de-nombres-ii/#comment-11>.
- Berzal, F. (2006). El ciclo de vida de un sistema de información. pp. 3-18. Diseño de Base de datos. Recuperado en <https://elvex.ugr.es/idbis/db/docs/lifecycle.pdf>
- Chavéz, E., Hermosa, E., & Villacís, C. (2016). Generador de Código Fuente para Gestión de Información de MySQL, SQL Server y Access para JAVA, PHP y ASP. *GEEKS DECC-REPORTS*, 4(1).
- Cordero, C. (2015). Conozca las ventajas y desventajas de los generadores de aplicaciones móviles. El Financiero. Recuperado de <https://www.elfinancierocr.com/tecnologia/conozca-las-ventajas-y-desventajas-de-los-generadores-de-aplicaciones-moviles/A37JVNY7RG2NHDNXXBXS6QERI/story/>.
- Dawcons (2014). DevExpress, el mejor aliado en el desarrollo [Mensaje en un blog]. Recuperado de <http://dawconsblog.blogspot.com/2014/04/devexpress-el-mejor-aliado-en-el.html>.
- Digital Guide (2018). El código fuente: ¿qué es y cómo se escribe? [Mensaje en un blog]. Recuperado de <https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/codigo-fuente-definicion-con-ejemplos/>.
- Espinel González, L. A., Acosta, N. J., & García Tovar, J. L. (2017). Estándares para la calidad de software. *Tecnología Investigación y Academia*, 5(1), 75–84. Recuperado de <https://revistas.udistrital.edu.co/index.php/tia/article/view/8388>
- Gaitán, C. A. P. (2017). Líneas de Productos Software: Generando Código a Partir de Modelos y Patrones. *Scientia et Technica*, 22(2), 175-182.
- Gestión (2017). Apesoft: Con entidad dedicada a las TIC se duplicaría el crecimiento del sector software. Recuperado de <https://gestion.pe/tecnologia/apesoft-entidad-dedicada-tic-duplicaria-crecimiento-sector-software-132697>.
- González Escudero, Á., Álvarez Ramírez, D. & Acosta Berrio, L. (2012). jPET 2.0: un generador automático de casos de prueba sobre programas Java. Repositorio de Universidad Complutense Madrid. Recuperado de [https://eprints.ucm.es/id/eprint/16095/1/memoria\\_JPet\\_2\\_0.pdf](https://eprints.ucm.es/id/eprint/16095/1/memoria_JPet_2_0.pdf)

- Herrera, L., Quiñonez, X. , & Casierra, J. C. (s.f). Generador automático de aplicaciones web e interfaces de usuarios con funcionalidad responsiva en el lenguaje Python. *La simulación en ingeniería, trascendiendo fronteras*, p. 37.
- Huari Casas, M. R. (2020). Revisión Sistemática Sobre Generadores De Código Fuente Y Patrones De Arquitectura. Repositorio de Pontificia Universidad Católica del Perú. Recuperado de <http://hdl.handle.net/20.500.12404/16457>
- Lazetic, S., Savic, D., Vlajic, S., & Lazarevic, S. (2012). A generator of mvc-based web applications. *World of Computer Science and Information Technology Journal (WCSIT)*, 2(4), 147-156.
- Martínez, G. V., & Camacho, G., & Biancha, D. (2010). Diseño de framework web para el desarrollo dinámico de aplicaciones. *Scientia et technica*, xvi (44), 178-183.
- Mattingley, J., & Boyd, S. (2012). CVXGEN: A code generator for embedded convex optimization. *Optimization and Engineering*, 13(1), 1-27.
- Minitab (s.f.). Pruebas de medias – tipos de pruebas t [Mensaje de un blog]. Recuperado de <https://support.minitab.com/es-mx/minitab/19/help-and-how-to/statistics/basic-statistics/supporting-topics/tests-of-means/types-of-t-tests/>
- Muradas, Y. (2016). Las 3 Metodologías Ágiles Más Usadas [Mensaje en un blog]. OpenWebinars. Recuperado de <https://openwebinars.net/blog/conoce-las-3-metodologias-agiles-mas-usadas/>.
- Picón, M., & Maldonado, C. Z. Generación Automática de Código basada en Modelos UML. In *Cuarta Conferencia Nacional de Computación, Informática y Sistemas (CoNCISa 2016)* (pp. 134-138).
- Proyectos Agiles (s.f.). Qué es SCRUM [Mensaje en un blog]. Recuperado de <https://openwebinars.net/blog/conoce-las-3-metodologias-agiles-mas-usadas/>.
- Radošević, D., & Magdalenic, I. (2011). Source code generator based on dynamic frames. *Journal of Information and Organizational Sciences*, 35(1), 73-91.
- Rebeca (s.f.). La Metodología Ágil Más Usada: Scrum [Mensaje en un blog]. Nextu. Recuperado de <https://www.nextu.com/blog/que-es-scrum/>.

- Rincón Nigro, M., Aguilar Castro, J., & Hidrobo Torres, F. (2011). Generación automática de código a partir de máquinas de estado finito. *Computación y Sistemas*, 14(4), 405-421.
- Rosales, V. Y. M., Alor, G. H., García, J. L. A., Zatarain, R. C., & Barrón, M. L. E. (2015). An analysis of tools for automatic software development and automatic code generation. *Revista Facultad de Ingeniería Universidad de Antioquia*, (77), 75-87.
- Sanchez, J. (2016). Software 1. Sistema Operativo. Software de Aplicación. pp. 2-3. Recuperado de: <https://proyectocirculos.files.wordpress.com/2013/11/software.pdf>
- Scrum Institute (s.f.). The Scrum Product Backlog [Mensaje en un blog]. Recuperado de [https://www.scrum-institute.org/The\\_Scrum\\_Product\\_Backlog.php](https://www.scrum-institute.org/The_Scrum_Product_Backlog.php).
- Scrum Institute (s.f.). The Sprint Backlog [Mensaje en un blog]. Recuperado de [https://www.scrum-institute.org/The\\_Sprint\\_Backlog.php](https://www.scrum-institute.org/The_Sprint_Backlog.php).
- Simba (s.f.). What is ODBC – Open Database Connectivity [Mensaje en un blog]. Recuperado de <https://www.simba.com/resources/odbc/>.

## ANEXOS

### Anexo 1

- Matriz de Consistencia

<b>GENERADOR DE CÓDIGO DE PROGRAMACIÓN EN LA ETAPA DE IMPLEMENTACION DE UN SOFTWARE DE LA EMPRESA INKA MALL S.A.C.</b>				
<b>PROBLEMA</b>	<b>HIPÓTESIS</b>	<b>OBJETIVO GENERAL</b>	<b>VARIABLE INDEPENDIENTE</b>	<b>METODOLOGÍA</b>
<p>¿De qué manera influye el uso de un generador de código de programación en la etapa de implementación de un software dentro de la empresa Inka Mall S.A.C. en el año 2019?</p>	<p>El generador de código de programación influye positivamente en la etapa de implementación de un software dentro de la empresa Inka Mall S.A.C. en el año 2019</p>	<p>Determinar la influencia de un generador de código de programación en la etapa de implementación de un software dentro de la empresa Inka Mall S.A.C. en el año 2019</p>	<p>Generador de código de programación</p>	<b>Diseño</b>
				<p style="text-align: center;"><math>G \quad O_1 \quad X \quad O_2</math></p> <p>Donde:            G = Muestra            X= Generador de código de programación            O1: Medición pre-test de la Etapa de implementación de un software.            O2: Medición post-test de la Etapa de implementación de un software.</p>



				<b>Población</b>
				Se tomó 27 tablas del diseño de base de datos de 1 proyecto de software de la empresa Inka Mall S.A.C.
		<b>OBJETIVOS ESPECIFICOS</b>	<b>VARIABLE DEPENDIENTE</b>	<b>Muestra</b>
		<p>Determinar la influencia del generador de código de programación en la dimensión de codificación en la etapa de implementación de un software.</p> <p>Determinar la influencia del generador de código de programación en los estándares de nomenclatura de la empresa establecida para la etapa de implementación de un software.</p>	<p>Etapa de implementación de un software.</p>	<p>8 tablas del diseño de base de datos del proyecto de software seleccionado por la empresa Inka Mall S.A.C.</p> <p>T = 8.</p>

## Anexo 2

- Operacionalización de la variable independiente

VARIABLE	DEFINICIÓN CONCEPTUAL	DEFINICIÓN OPERACIONAL	DIMENSIONES	INDICADORES
GENERADOR DE CÓDIGO DE PROGRAMACIÓN	El Generador de código es un sistema que permite implementar código de procesos comunes en un software, el cual brinda a los desarrolladores la opción de generar sus propias plantillas personalizables de acuerdo al lenguaje de programación que desean usar. Basado en Murga (2013)	Para medir el generador de código, se debe tener en cuenta el cumplimiento de la usabilidad y funcionalidad, según el modelo de calidad de software según la ISO/IEC 25010	Funcionalidad	Porcentaje de código generado correctamente
				Porcentaje de procesos generados completos
				Porcentaje de conexiones a base de datos validas
			Usabilidad	Relación entre el generador y el mundo real
				Consistencia y estándares
				Diseño estético y minimalista

- Operacionalización de la variable dependiente.

VARIABLE	DEFINICIÓN CONCEPTUAL	DEFINICIÓN OPERACIONAL	DIMENSIONES	INDICADORES	MEDICIÓN DE INDICADORES
ETAPA DE IMPLEMENTACIÓN DE UN SOFTWARE	El ciclo de vida de un software cuenta con diferentes etapas, una de ellas es la implementación, donde se realiza la programación de procesos a través de códigos específicos de acuerdo con un lenguaje de programación a usar, este proceso demanda muchas horas a invertir, como también realizar el análisis de la base de datos y de cada proceso que se requiere implementar para los procesos por automatizar.  Basado en Berzal, F. (2004). El ciclo de vida de un sistema de información.	Para obtener un código de programación de calidad, se requiere realizar muchas revisiones y analizar cada requerimiento a desarrollar, para de esta manera aplicar la reutilización de métodos y no duplicar códigos que ya están implementado, es necesario para ello reducir las dependencias de los métodos y realizar las referencias necesarias de acuerdo con la base de datos que se diseñe para el sistema.	Codificación	Tiempo en el análisis y solución de errores de codificación.  Tiempo de implementación de la codificación.	$EFICACIA = \frac{TE}{T_1 + T_2} * 100\%$ <p><b>Donde:</b>  <math>T_1</math>: Tiempo de corrección de errores.  <math>T_2</math>: Tiempo de implementación de la codificación.  <math>TE</math>: Tiempo esperado</p>
			Estándares de nomenclatura	Porcentaje de cumplimiento de estándares de nomenclatura	$EFICACIA = \frac{PO}{PE} * 100\%$ <p><b>Donde:</b>  <math>PO</math>: Porcentaje Obtenido.  <math>PE</math>: Porcentaje Esperado.</p>

### Anexo 3

#### FICHA DE OBSERVACIÓN N° 1

**Evaluador a cargo de la prueba:** \_\_\_\_\_

**Fecha y hora de la prueba:** \_\_\_\_\_

**Nombre de la Tabla de base de datos:** \_\_\_\_\_

**Instrucciones:**

1. Lea atentamente los cuatro criterios consignados en la siguiente tabla, cada uno de ellos refleja un punto referente a la adecuación de estándares de nomenclatura del código de programación.
2. Acceda al código y haga revise cada línea, verificando los criterios proporcionados.
3. Llene el casillero que corresponda en cada uno de los criterios a observar. Escriba en el cuadro de observaciones de ser necesario.

Nº	PREGUNTAS / CRITERIOS	Cantidad correcta	Cantidad incorrecta	OBSERVACIONES
1	Los nombres de los métodos usan el tipo de escritura Snake_Case			
2	Los nombres de los parámetros usan el tipo de escritura Camel Case			
3	Los nombres de las funciones usan el tipo de escritura Snake_Case			
4	Los nombres de las variables usan el tipo de escritura Camel_Case			
Total				

Anexo 4

**FICHA DE OBSERVACIÓN Nº 2**

**Evaluador a cargo de la prueba:** \_\_\_\_\_

**Fecha y hora de la prueba:** \_\_\_\_\_

**Instrucciones:**

1. Lea atentamente los cuatro criterios consignados en la siguiente tabla.
2. Controle el tiempo de inicio a fin de la implementación de código de programación realizado a mano y también de manera automática.
3. Llene el casillero que corresponda en cada uno de los criterios a observar. Escriba en el cuadro de observaciones de ser necesario.

Recordar que:

- TCG (Tiempo Con Generador) : Tiempo transcurrido para la codificación de un CRUD generado automáticamente.
- TSG (Tiempo Sin Generador) : Tiempo transcurrido para la codificación de un CRUD desarrollado manualmente.

Nº DE PRUEBA	RESULTADOS		OBSERVACIONES
	TCG	TSG	
<b>Sistema en etapa de implementación</b>			
1			
2			
3			
4			
5			
6			
7			
8			

Anexo 5

**FICHA DE OBSERVACIÓN N° 3**

**Evaluador a cargo de la prueba:** \_\_\_\_\_

**Fecha y hora de la prueba:** \_\_\_\_\_

**Instrucciones:**

1. Lea atentamente los cuatro criterios consignados en la siguiente tabla.
2. Controle el tiempo de inicio a fin de la corrección de errores presentados en el código de programación realizado a mano y también de manera automática.
3. Llene el casillero que corresponda en cada uno de los criterios a observar. Escriba en el cuadro de observaciones de ser necesario.

Recordar que:

- SECG (Solución De Errores Con Generador) : Tiempo transcurrido para la solución de errores de la codificación de un CRUD generado automáticamente.
- SESG (Solución De Errores Sin Generador) : Tiempo transcurrido para la solución de errores de la codificación de un CRUD desarrollado manualmente.

N° DE PRUEBA	RESULTADOS		OBSERVACIONES
	SECG	SESG	
<b>Sistema en etapa de implementación</b>			
1			
2			
3			
4			
5			
6			
7			
8			

Anexo 6

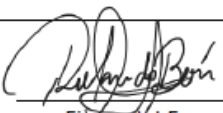
MATRIZ DE VALIDACIÓN DE EXPERTOS				
<b>Título de la investigación:</b>		"Generador de código de programación en la etapa de implementación de un software de la empresa Inka Mall S.A.C., 2019"		
<b>Línea de investigación:</b>		Tecnologías Emergentes		
<b>El o los instrumentos de medición pertenece(n) a la variable:</b>		Etapa de implementación de un software		
<p>Mediante la matriz de validación de expertos, Ud. tiene la facultad de evaluar cada una de las preguntas marcando con una "x" en las columnas de SÍ o NO. Asimismo, le exhortamos en la corrección de los ítems, indicando sus observaciones y/o sugerencias, con la finalidad de mejorar la coherencia de las preguntas sobre la variable en estudio.</p>				
Ítems	Preguntas	Aprecia		Observaciones
		SÍ	NO	
1	¿El instrumento de medición presenta el diseño adecuado?	X		
2	¿El instrumento de recolección de datos tiene relación con el título de la investigación?	X		
3	¿En el instrumento de recolección de datos se mencionan las variables, dimensiones o indicadores de la investigación?	X		
4	¿El instrumento de recolección de datos facilitará el logro de los objetivos de la investigación?	X		
5	¿El instrumento de recolección de datos se relaciona con las variables de estudio?	X		
6	¿La redacción de las preguntas tienen un sentido coherente y no están sesgadas? - (En caso de cuestionarios)			
7	¿Cada una de las preguntas del instrumento de medición se relaciona con cada uno de los elementos de los indicadores? - (En caso de cuestionarios)			
8	¿El diseño del instrumento de medición facilitará el análisis y procesamiento de datos?	X		
9	¿Son entendibles las alternativas de respuesta del instrumento de medición? - (En caso de cuestionarios)			
10	¿El instrumento de medición será accesible a la población/muestra de estudio?	X		
11	¿El instrumento de medición es claro, preciso y sencillo de responder para, de esta manera, obtener los datos requeridos?	X		
<b>Sugerencias:</b>				
Nombre completo: <u>Rolando Javier Borri Beltrán</u> DNI: <u>46689929</u> Grado: <u>Magister</u>		 Firma del Experto		

Figura 9. Matriz de validación de expertos.

## Anexo 7

### Plan de auditoría de Inka Mall S.A.C.

EJECUCIÓN DE PLAN DE AUDITORIA	
1. Objetivo	Es auditar a los proyectos para asegurar que el equipo de desarrollo trabaje bajo los nuevos estándares de Inka Mall.
2. Frecuencia	Por sprint
3. Materiales	Se utilizará los "Items para Auditoría" (Hoja 2)
4. Métricas de evaluación	Se definió una escala 0 - 1 siendo 0: NO CUMPLE, 1: CUMPLE, 0.5 CUMPLE PARCIALMENTE. En caso el desarrollador no haya trabajado en todos los items durante su sprint, su evaluación será en base a lo que haya trabajado. Por ejemplo: si solo trabajó a nivel de Base de datos, entonces se le aplicaran las métricas de la base de datos. (en este caso, si solo son 7 items, entonces la nota máxima será 7, siendo este su 100%.
5. Acciones a ejecutar	Si la nota del proyecto es mayor e igual a 90%, se realizará un documento con las observaciones a corregir Si la nota del proyecto está entre 89% - 50 %, se realizará una amonestación verbal, además del documento con las observaciones a corregir. Si la nota del proyecto es menor 50%, se realizará una amonestación verbal. Si ésta nota es reincidente consecutivamente, se procederá a la amonestación escrita.
6. Población	Es todo el código del proyecto
7. Muestra	Es el código que impacta en el sprint auditado.
8. Duración de auditoría	Tendrá una duración máxima de 2 días Nota: La duración de la auditoría reduzca a un día, una meta a mediano plazo.
9. Equipo Auditor	Deivi Gomez, Graly Troncozo Raul Mera, Alan Rios, Alexis Garcia, Jose Ruiz
10. Revisión de Observación	Después de presentado el documento con observaciones, el equipo de desarrollo tendrá máximo un día para corregir las observaciones El equipo auditor realizará nuevamente el plan de auditoría, verificando que las observación haya sido corregido. (Tiempo máximo 1 día)  el auditado proveera una lista de tickets trabajados



**Figura 10.** Aspectos del plan de auditoría a la codificación de software de la empresa Inka Mall S.A.C



## Anexo 8

### - Historial de tiempos estimados de implementación de CRUD's.

**Feature #41048** Edit Log time Watch

Feature #27613: Instalacion de Modulos en nuevo servidor creado  
 Feature #40376: Inkamall - Modulo xCorporate  
 Feature #40409: xCorporate | Productos  
**Productos | Subcategorias del producto**  
 Added by Anonymous 4 months ago. Updated 14 days ago.

<b>Status:</b>	Closed	<b>Start date:</b>	02/04/2019
<b>Priority:</b>	Normal	<b>Due date:</b>	02/05/2019
<b>Assignee:</b>	Carolina Davies Falen	<b>% Done:</b>	<div style="width: 100%;"><div style="width: 100%;"></div></div> 100%
<b>Category:</b>	-	<b>Estimated time:</b>	(Total: 7.00 h)
<b>Target version:</b>	Sprint IV	<b>Spent time:</b>	6.50 h (Total: 9.50 h)
<b>Quote Approved:</b>		<b>Created by QA:</b>	SI

**Root Cause Analysis (RCA):**

**Description** Quote

Se solicita desarrollar el menú de subcategorias. Este menú debe permitir agregar, editar, mostrar y eliminar los registros de subcategorias.

- La pagina principal del menú debe considerar una tabla con los siguientes registros: Checkbox, Acciones, N°, Categoría, Subcategoría, Key de traducción, Estado.
- El formulario para agregar una subcategoría debe considerar:
  - Categoría: Combobox del listado de las categorías.
  - Nombre subcategoría: Input para registrar el nombre de la subcategoría. (Max. 50 caracteres).
  - Key de traducción: Input. (Max. 50 caracteres).
  - Nombre corto: Input para registrar el nombre corto. (Max. 50 caracteres).
  - Descripción: Input de texto para registrar la descripción. (Max. 50 caracteres).
  - Stock de seguridad: Input numérico. (Max. 4 caracteres).
  - Especificaciones: Registro de las especificaciones en la categoría. Considerar estructura actual.
  - Tiene talla: Checkbox.
  - Habilitar: Checkbox, considerar estados habilitado e inhabilitado.
  - Regresar: Botón para regresar a la lista de registros.
  - Guardar: Botón para guardar el registro actual. Después de guardar debe redirigir a la pantalla de lista de registros de subcategorias.
- Considerar todos los campos obligatorios.

**El campo para las especificaciones no muestra resultados**

- xCMenusubcategoriasAgregar2.png 21.9 KB IMG1 Anonymous, 01/25/2019 02:29 PM
- tablasubcategoría.png 68.8 KB IMG1 Anonymous, 02/04/2019 05:16 PM
- camposubcategorias.png 46.9 KB IMG2 Anonymous, 02/04/2019 05:27 PM
- especificacionesubcat.png 16.5 KB IMG3 Anonymous, 02/04/2019 05:28 PM





Figura 11. Ticket N° 01 del historial de tiempos de codificación.

**Feature #40410** Edit Log time Watch

Feature #27613: Instalación de Modulys en nuevo servidor creado  
 Feature #40376: Inkamall - Modulo xCorporate  
 Feature #40405: xCorporate | Productos

**Productos | Grupos de categoría**  
 Added by Carolina Davies Falen 5 months ago, Updated 20 days ago.

<b>Status:</b> Closed-Client Accepted	<b>Start date:</b> 02/19/2019
<b>Priority:</b> Normal	<b>Due date:</b> 02/19/2019
<b>Assignee:</b> Carolina Davies Falen	<b>% Done:</b> <div style="width: 100%;"><div style="width: 100%;"></div></div> 100%
<b>Category:</b> -	<b>Estimated time:</b> 8.00 h
<b>Target version:</b> Sprint VI	<b>Spent time:</b> 8.00 h
<b>Quote Approved:</b>	<b>Created by QA:</b>
<b>Root Cause Analysis (RCA):</b>	



**Additional Details**  
 Additional Assignee

**Subtasks** Add

**Related issues** Add

**History**

Updated by Anonymous 4 months ago #1

- Attachments tablagrupodecategorias.png added
- Attachments grupodecatxR.png added
- Attachments grupodecatxchangeXC.png added
- Attachments grupodecatxchangeXC.png added
- Attachments grupodecatxchangeXC.png added

Se realizó las pruebas respectivas a las funciones del menú de grupo de categorías.  
 1. En la tabla de registros del menú los datos se muestran correctamente después de agregar un registro. (IMG1)  
 2. La función para **agregar un grupo de categoría**, funciona correctamente después de completar todos los datos del formulario este registro se muestra en el listado de xBackoffice (IMG2). *Aun no se puede validar que se muestre en xR por dependencia con las categorías.*  
 3. La función para editar y eliminar funcionan correctamente y se muestran los cambios en los módulos de xB y xC. (IMG3, IMG4 Y IMG5)  
 4. La función Habilitado, al estar desactivada funciona correctamente ya que se oculta en el módulo de xR.

Se espera que se considere **ocultar el campo de ORDEN** ya que al parecer no se utiliza en el momento de ordenar la lista de grupo de categorías.

Updated by Anonymous 4 months ago #2

Figura 12. Ticket N° 02 del historial de tiempos de codificación.

**Feature #41048** Edit Log time Watch

Feature #27613: Instalación de Módulos en nuevo servidor creado  
 Feature #40376: Inkamall - Módulo xCorporate  
 Feature #40409: xCorporate | Productos

**Productos | Subcategorías del producto**

Added by Anonymous 4 months ago. Updated 14 days ago.

<b>Status:</b>	Closed	<b>Start date:</b>	02/04/2019
<b>Priority:</b>	Normal	<b>Due date:</b>	02/05/2019
<b>Assignee:</b>	Carolina Davies Falen	<b>% Done:</b>	100%
<b>Category:</b>	-	<b>Estimated time:</b>	(Total: 6.00 h)
<b>Target version:</b>	Sprint IV	<b>Spent time:</b>	6.50 h (Total: 9.00 h)
<b>Quote Approved:</b>		<b>Created by QA:</b>	Si

**Root Cause Analysis (RCA):**


**Description** Quote

Se solicita desarrollar el menú de subcategorías. Este menú debe permitir agregar, editar, mostrar y eliminar los registros de subcategorías.

- La página principal del menú debe considerar una tabla con los siguientes registros: Checkbox, Acciones, N°, Categoría, Subcategoría, Key de traducción, Estado.
- El formulario para agregar una subcategoría debe considerar:
  - Categoría: Combobox del listado de las categorías.
  - Nombre subcategoría: Input para registrar el nombre de la subcategoría. (Max. 50 caracteres).
  - Key de traducción: Input. (Max. 50 caracteres).
  - Nombre corto: Input para registrar el nombre corto. (Max. 50 caracteres).
  - Descripción: Input de texto para registrar la descripción. (Max. 50 caracteres).
  - Stock de seguridad: Input numérico. (Max. 4 caracteres).
  - Especificaciones: Registro de las especificaciones en la categoría. Considerar estructura actual.
  - Tiene talla: Checkbox.
  - Habilitar: Checkbox, considerar estados habilitado e inhabilitado.
  - Regresar: Botón para regresar a la lista de registros.
  - Guardar: Botón para guardar el registro actual. Después de guardar debe redirigir a la pantalla de lista de registros de subcategorías.
- Considerar todos los campos obligatorios.

**El campo para las especificaciones no muestra resultados**

- xCMenuSubcategoriasAgregar2.png 21.9 KB Anonymous, 01/25/2019 02:29 PM
- tablasubcategoria.png 68.8 KB IMG1 Anonymous, 02/04/2019 05:16 PM
- compossubcategorias.png 46.9 KB IMG2 Anonymous, 02/04/2019 05:27 PM
- especificacionessubcat.png 18.3 KB IMG3 Anonymous, 02/04/2019 05:28 PM




**Additional Details**

Figura 13. Ticket N° 03 del historial de tiempos de codificación.

**Feature #40410** Edit Log time Watch

Feature #27613: Instalacion de Módulos en nuevo servidor creado  
 Feature #40376: Inkamall - Módulo xCorporate  
 Feature #40409: xCorporate | Productos  
**Productos | Grupos de categoria**  
 Added by Carolina Davies Falen 5 months ago. Updated 20 days ago.

<b>Status:</b>	Closed-Client Accepted	<b>Start date:</b>	02/19/2019
<b>Priority:</b>	Normal	<b>Due date:</b>	02/19/2019
<b>Assignee:</b>	Carolina Davies Falen	<b>% Done:</b>	<div style="width: 100%;"><div style="width: 100%;"></div></div> 100%
<b>Category:</b>	-	<b>Estimated time:</b>	5.50 h
<b>Target version:</b>	Sprint VI	<b>Spent time:</b>	6.00 h
<b>Quote Approved:</b>		<b>Created by QA:</b>	
<b>Root Cause Analysis (RCA):</b>			



**Additional Details**  
Additional Assignee

**Subtasks** Add

**Related issues** Add

**History** #1

Updated by Anonymous 4 months ago

- Attachments tablagrupodecategorias.png added
- Attachments grupodecatxh.png added
- Attachments grupodecatxh.png added
- Attachments grupodecatxh.png added
- Attachments grupodecatxh.png added
- Attachments grupodecatxh.png added

Se realizó las pruebas respectivas a las funciones del menu de grupo de categorias.

1. En la tabla de registros del menú los datos se muestran correctamente después de agregar un registro. (IMG1)
2. La función para **agregar un grupo de categoria**, funciona correctamente después de completar todos los datos del formulario este registro se muestra en el listado de xBackoffice (IMG2). Aun no se puede validar que se muestre en xR por dependencia con las categorias.
3. La función para editar y eliminar funcionan correctamente y se muestran los cambios en los modulos ad xB y xC. (IMG3, IMG4 Y IMG5)
4. La función Habilitado, al estar desactivada funciona correctamente ya que se oculta en el modulo de xR.

Se espera que se considere ocultar el campo de ORDEN ya que al parecer no se utiliza en al momento de ordenar la lista de grupo de categorias.

**Figura 14.** Ticket N° 04 del historial de tiempos de codificación.

**Feature #39661** Edit Log time Watch

Feature #27613: Instalacion de Modulos en nuevo servidor creado  
 Feature #40210: Inkamall - Modulo xReplicate  
 Feature #40239: Módulo xReplicate | Home principal  
 Feature #40246: Home principal | Cuerpo  
 Feature #40245: Cuerpo | Sección banner principal

**Inkamall -> xR backend -> Principal -> Banner de productos**

Added by Anonymous 6 months ago. Updated 3 months ago.

<b>Status:</b>	Test - Failed	<b>Start date:</b>	12/13/2018
<b>Priority:</b>	Normal	<b>Due date:</b>	12/13/2018
<b>Assignee:</b>	Piero Hidalgo	<b>% Done:</b>	<div style="width: 100%;"><div style="width: 100%;"></div></div> 100%
<b>Category:</b>	-	<b>Estimated time:</b>	4.00 h
<b>Target version:</b>	-	<b>Spent time:</b>	4.00 h
<b>Quote Approved:</b>		<b>Created by QA:</b>	No

**Root Cause Analysis (RCA):**

**Description** Quote

1. Realizar la parte de Backend para el insert, update, select y delete de la parte de Banner de productos .

URL: <https://xcorporateinkamall.xssclients.com/Private/global/BannersActive.aspx>  
 URL: <https://xcorporateinkamall.xssclients.com/Private/global/BannersActive.aspx>

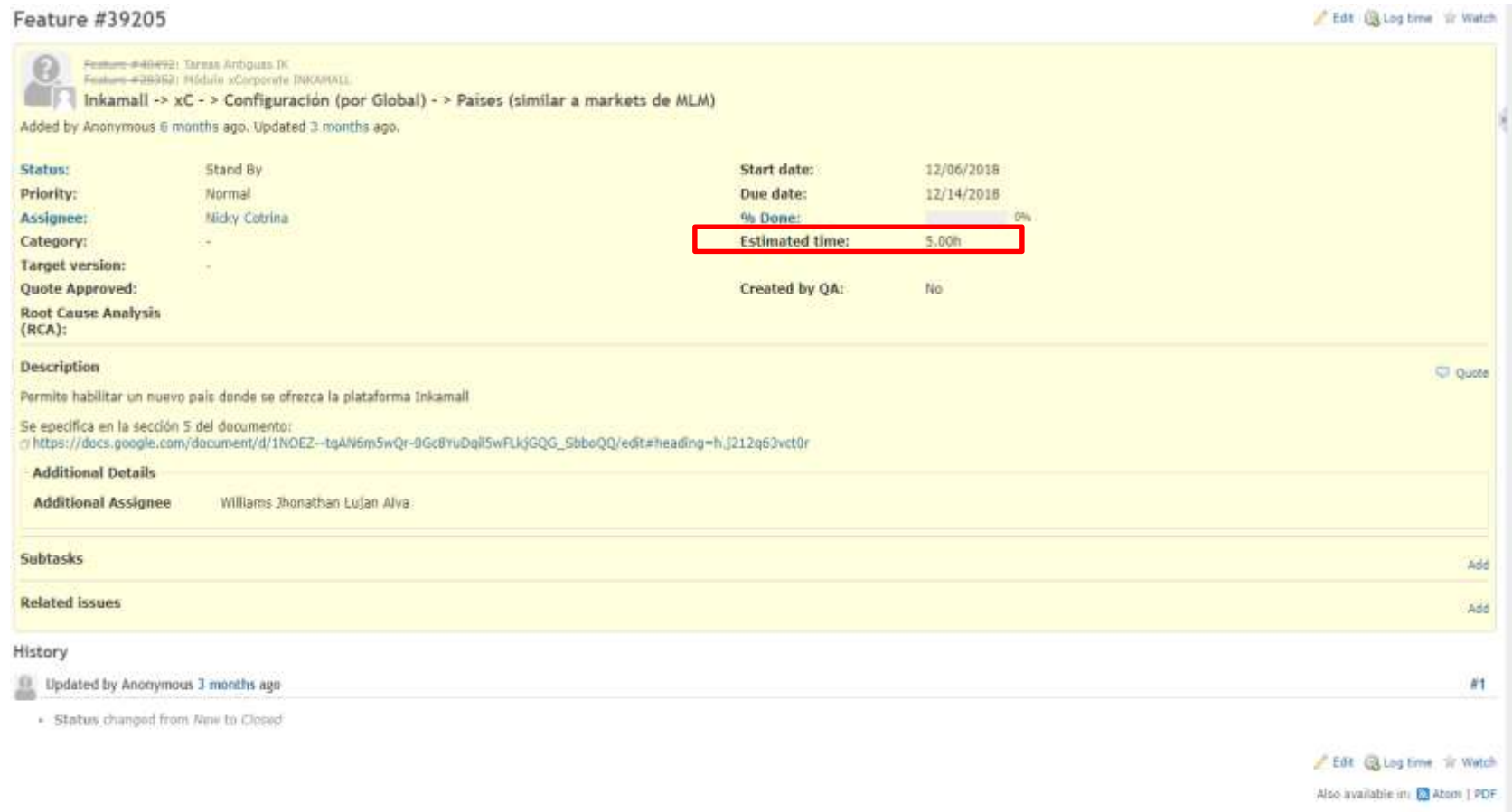
**Additional Details**

**Additional Assignee:** Williams Jhonathan Lujan Alva

**Subtasks** Add

**Related issues** Add

**Figura 15.** Ticket N° 05 del historial de tiempos de codificación.



**Feature #39205** Edit Log time Watch

Feature #48452: Tareas Antiguas TK  
Feature #28952: Módulo xCorporate INKAMALL

**Inkamall -> xC -> Configuración (por Global) -> Países (similar a markets de MLM)**

Added by Anonymous 6 months ago, Updated 3 months ago.

<b>Status:</b>	Stand By	<b>Start date:</b>	12/06/2018
<b>Priority:</b>	Normal	<b>Due date:</b>	12/14/2018
<b>Assignee:</b>	Nicky Cotrina	<b>% Done:</b>	0%
<b>Category:</b>	-	<b>Estimated time:</b>	5.00h
<b>Target version:</b>	-	<b>Created by QA:</b>	No
<b>Quote Approved:</b>	-		
<b>Root Cause Analysis (RCA):</b>	-		

**Description** Quote

Permite habilitar un nuevo país donde se ofrezca la plataforma Inkamall

Se especifica en la sección 5 del documento:  
[https://docs.google.com/document/d/1NOEZ--tqAN6m5wQr-0Gc8YuDqil5wFLkjGQG\\_StiboQQ/edit#heading=h.j212q63vct0r](https://docs.google.com/document/d/1NOEZ--tqAN6m5wQr-0Gc8YuDqil5wFLkjGQG_StiboQQ/edit#heading=h.j212q63vct0r)

**Additional Details**

**Additional Assignee** Williams Jhonathan Lujan Alva

**Subtasks** Add

**Related issues** Add

**History** #1

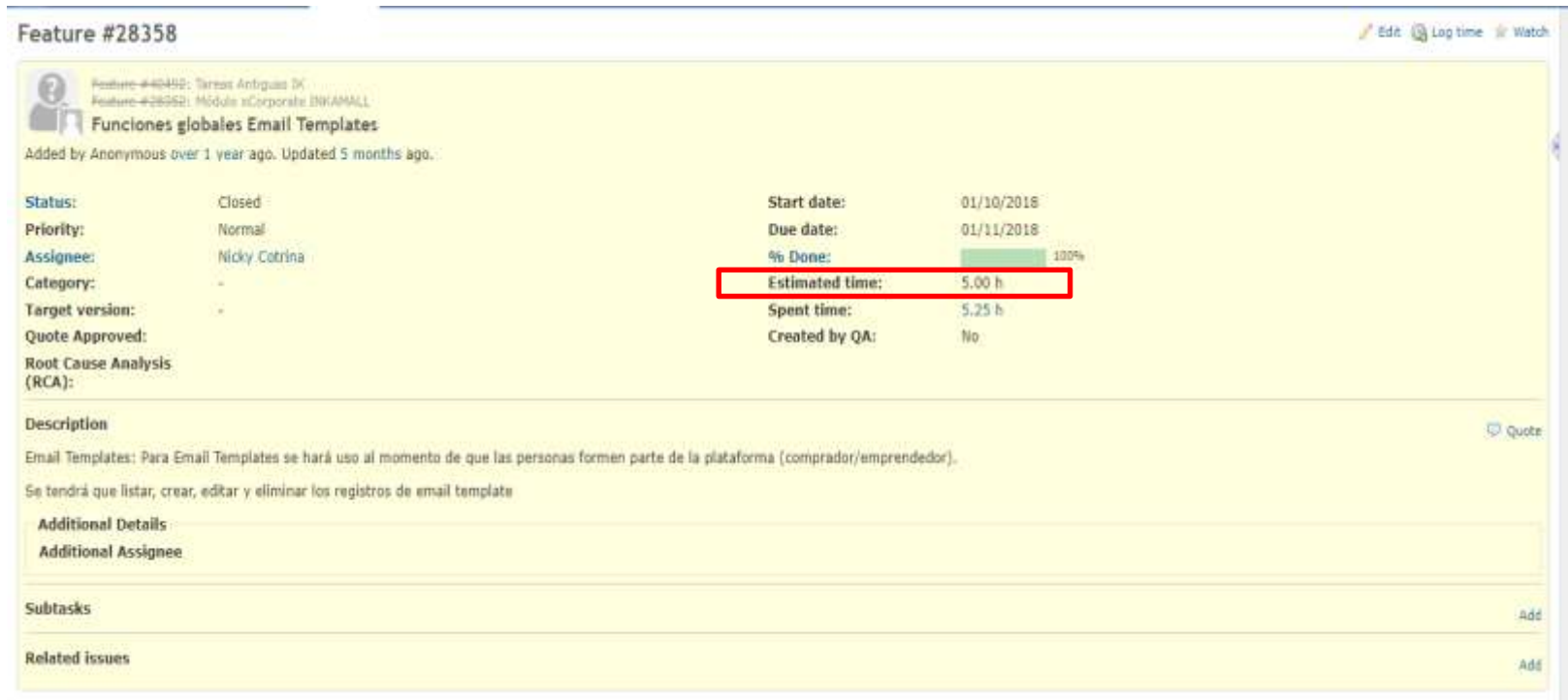
Updated by Anonymous 3 months ago

- Status changed from New to Closed

Edit Log time Watch

Also available in: [Atom](#) | [PDF](#)

Figura 16. Ticket N° 06 del historial de tiempos de codificación.



**Feature #28358** Edit Log time Watch

**Funciones globales Email Templates**  
Added by Anonymous over 1 year ago. Updated 5 months ago.

<b>Status:</b>	Closed	<b>Start date:</b>	01/10/2018
<b>Priority:</b>	Normal	<b>Due date:</b>	01/11/2018
<b>Assignee:</b>	Nicky Cotrina	<b>% Done:</b>	<div style="width: 100%;"><div style="width: 100%;"></div></div> 100%
<b>Category:</b>	-	<b>Estimated time:</b>	5.00 h
<b>Target version:</b>	-	<b>Spent time:</b>	5.25 h
<b>Quote Approved:</b>		<b>Created by QA:</b>	No

**Root Cause Analysis (RCA):**

**Description** Quote

Email Templates: Para Email Templates se hará uso al momento de que las personas formen parte de la plataforma (comprador/emprendedor).  
Se tendrá que listar, crear, editar y eliminar los registros de email template

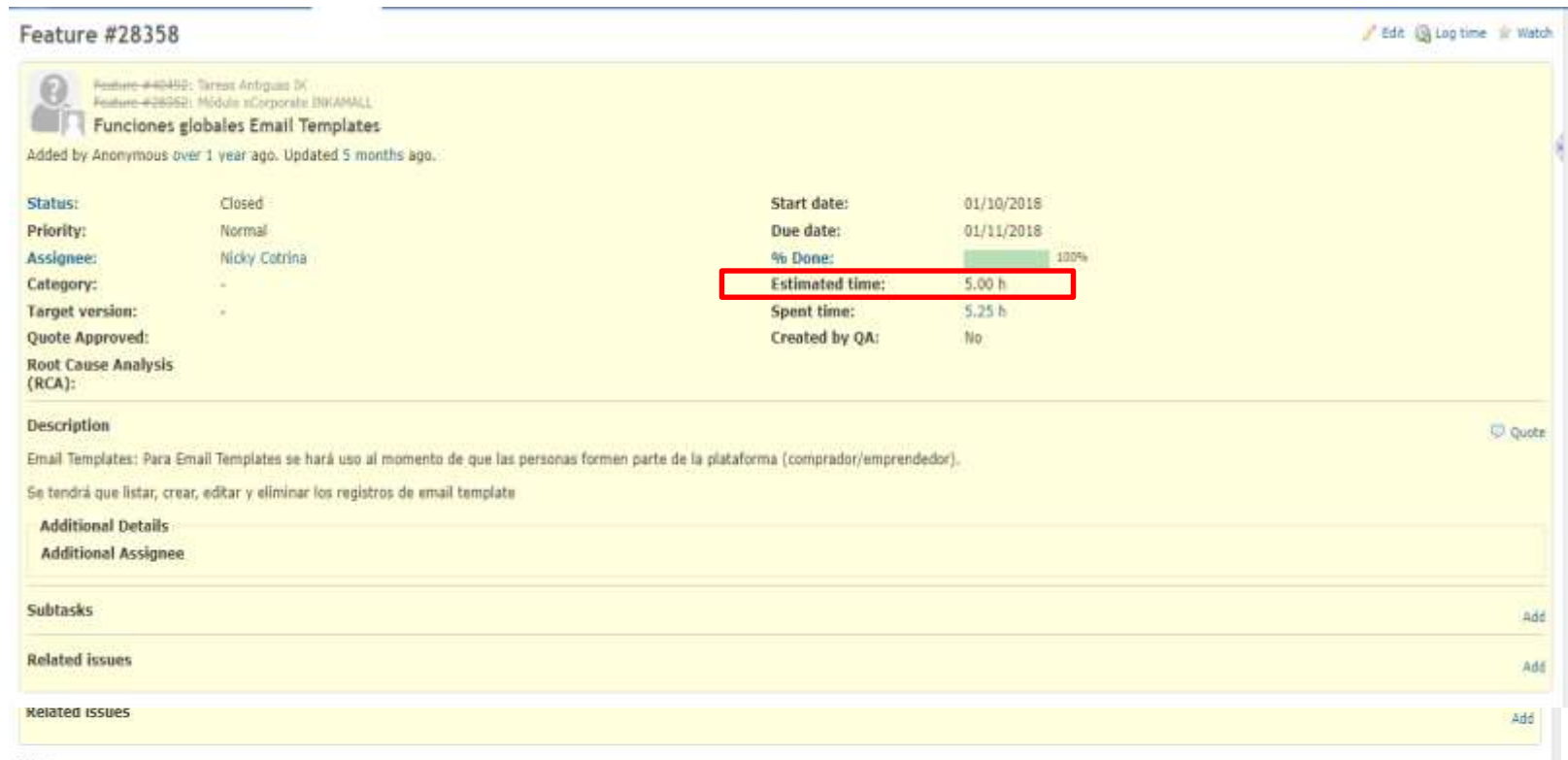
**Additional Details**

**Additional Assignee**

**Subtasks** Add

**Related issues** Add

**Figura 17.** Ticket N° 07 del historial de tiempos de codificación.



**Feature #28358** Edit Log time Watch

Feature #40492: Tareas Antiguas IK  
Feature #28352: Módulo xCorporate INKAMALL  
**Funciones globales Email Templates**  
Added by Anonymous over 1 year ago. Updated 5 months ago.

<b>Status:</b>	Closed	<b>Start date:</b>	01/10/2018
<b>Priority:</b>	Normal	<b>Due date:</b>	01/11/2018
<b>Assignee:</b>	Nicky Cotrina	<b>% Done:</b>	<div style="width: 100%;"><div style="width: 100%;"></div></div> 100%
<b>Category:</b>	-	<b>Estimated time:</b>	5.00 h
<b>Target version:</b>	-	<b>Spent time:</b>	5.25 h
<b>Quote Approved:</b>		<b>Created by QA:</b>	No
<b>Root Cause Analysis (RCA):</b>			

**Description** Quote

Email Templates: Para Email Templates se hará uso al momento de que las personas formen parte de la plataforma (comprador/emprendedor).  
Se tendrá que listar, crear, editar y eliminar los registros de email template

**Additional Details**

**Additional Assignee**

**Subtasks** Add

**Related issues** Add

**Related issues** Add

**Figura 18.** Ticket N° 08 del historial de tiempos de codificación.



## Anexo 9

### Documentación de la implementación del generador.

#### 1. Equipo Scrum

Persona	Contacto	Rol
Nicky Alejandro Cotrina de la Cruz	Nicky_17_11@hotmail.com	Product Owner / Software Developer
Eduard Marquez Zavaleta	eduard_394@hotmail.com	Scrum Master / Tester

#### 2. Valores de Trabajo de Equipo Scrum

Los valores que deben de ser practicados por todos los miembros involucrados en el desarrollo y que hacen posible que la metodología Scrum tenga éxito son:

- ✓ Autonomía del equipo
- ✓ Respeto en el equipo
- ✓ Responsabilidad y auto-disciplina
- ✓ Foco en la tarea
- ✓ Información, transparencia y visibilidad.

#### 3. Responsabilidades del Equipo Scrum

##### 3.1. Scrum Master

- ❖ Proteger al Equipo Scrum de solicitudes externas e interrupciones
- ❖ Actuar como agente de cambio
- ❖ Entrenar al Equipo Scrum
- ❖ Eliminar impedimentos para el Equipo Scrum
- ❖ Garantizar una comunicación eficiente entre Equipo Scrum y Product Owner
- ❖ Facilitar los Eventos Scrum

##### 3.2. Product Owner

- ❖ Gestionar el Product Backlog
- ❖ Gestionar de definir los Sprints
- ❖ Trabajar directamente con el Equipo Scrum

## 4. ARTEFACTOS

### 4.1. Product Backlog

ID	PRIORIDAD	Estimación	REQUERIMIENTOS
Ingresar identificador con la siguiente nomenclatura PB001	Ingresar identificador numérico, en orden ascendente. Ejemplo: 1,2,3	Basada en número de días (no máximo de 8h por día)	Ingresar los requerimientos definidos.
PB001	1	1 día	Estructura de Aplicación Escritorio
PB002	1	1 día	Creación de casos de uso
PB003	1	1 día	Creación de especificaciones de caso de uso
PB004	1	2 días	Diseño de Mockups
PB005	1	3 días	Implementar formulario Lectura Base de Datos
PB006	1	3 días	Implementar formulario Definición de tipo de Datos
PB007	1	4 días	Implementar formulario de Generación Plantillas
PB008	1	6 días	Implementar formulario Generación de Código
PB009	3	2 días	Testing
PB010	3	12 días	Elaborar Documento del Proyecto Tesis
PB011	3	11 días	Elaborar Reuniones Sprint

- ✓ Para la elaboración del Product Backlog no se ha tomado en consideración realizar actividad alguna los días domingos.
- ✓ Se puede apreciar que el total de días que demanda realizar el proyecto es de 46 días.

## 4.2.Sprint Backlog

<b>Id Product Backlog</b>	<b>Id Tarea</b>	<b>Tarea</b>	<b>Días</b>	<b>Dependencia</b>	<b>Responsable</b>	<b>Avance (%)</b>
Ingresar identificador del requerimiento ingresado en el Producto Backlog	Ingresar identificador con la siguiente nomenclatura T001	Ingresar las tareas asignadas en el requerimiento.	Ingresar de forma numérica el tiempo (h) que toma realizar la tarea	Ingresar el identificador de la tarea en caso alguna tarea a desarrollar depende de la culminación de la otra	Ingresar el nombre de la persona responsable a realizar la tarea	Ingresar el porcentaje en el que se va avanzando la solución de la tarea
PB001	T001	Crear estructura de la aplicación escritorio	3 h		Nicky Cotrina de la Cruz	100%
PB002	T002	Creación de casos de uso	3 h	T001	Nicky Cotrina de la Cruz	100%
PB003	T003	Creación de especificaciones de caso de uso	3 h	T001	Nicky Cotrina de la Cruz	100%
PB004	T004	Diseño de Mockups	6 h		Nicky Cotrina de la Cruz	100%
PB005	T005	Diseño de formulario Lectura Base de Datos	5 h	T004	Nicky Cotrina de la Cruz	100%
PB005	T006	Implementar formulario Lectura Base de Datos	8 h	T005	Nicky Cotrina de la Cruz	100%
PB006	T007	Diseño de formulario Definición de tipo de Datos	4 h	T004	Nicky Cotrina de la Cruz	100%
PB006	T008	Implementar formulario Definición de tipo de Datos	5 h	T007	Nicky Cotrina de la Cruz	100%
PB007	T009	Diseño de formulario de Generación Plantillas	8 h	T004	Nicky Cotrina de la Cruz	100%
PB007	T010	Implementar formulario de Generación Plantillas	10 h	T009	Nicky Cotrina de la Cruz	100%

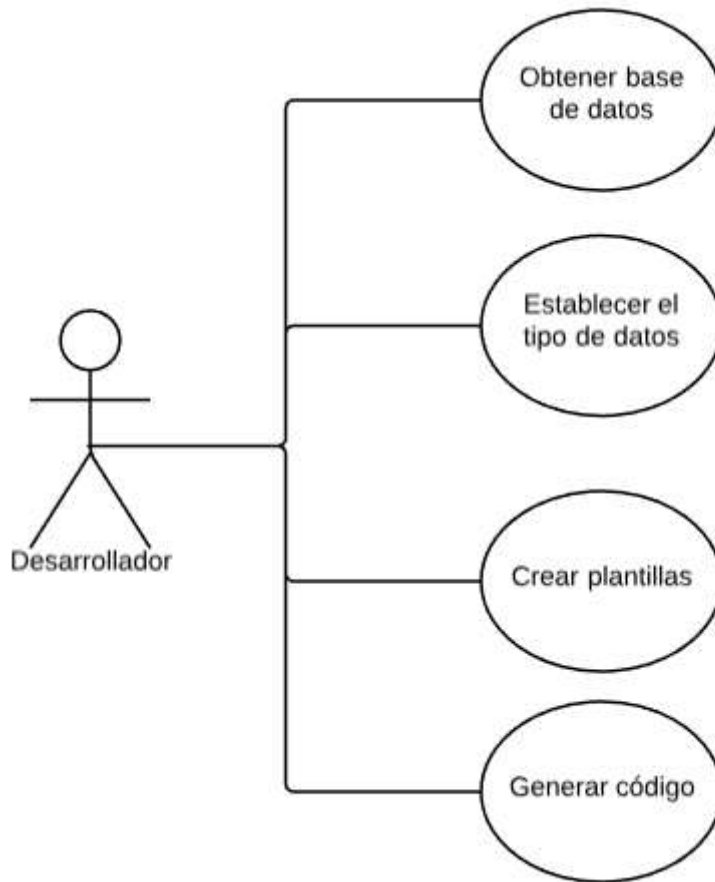
PB008	T011	Diseño de formulario Generación de Código	5 h	T004	Nicky Cotrina de la Cruz	100%
PB008	T012	Implementar formulario Generación de Código	24 h	T011	Nicky Cotrina de la Cruz	100%
PB009	T013	Pruebas funcionales de la aplicación	8 h		Eduard Marquez Zavaleta	100%

### 4.3.Sprint

Las iteraciones del ciclo de vida con el que cuenta el proyecto es de 8 Sprint. De acuerdo a las funcionalidades, los sprint varían entre 1 o 2 semanas, lo cual se aprecia en el siguiente cuadro a detalle:

Nombre Sprint	Requerimientos a terminar
Sprint 1	<ul style="list-style-type: none"> <li>- Estructura de la aplicación escritorio.</li> <li>- Creación de casos de uso del negocio.</li> <li>- Creación de casos de uso del Sistema.</li> </ul>
Sprint 2	<ul style="list-style-type: none"> <li>- Creación de las especificaciones de casos de uso de lectura de base de datos.</li> <li>- Creación de la especificación de casos de uso de generación de código.</li> </ul>
Sprint 3	<ul style="list-style-type: none"> <li>- Diseño de Mockups de los distintos formularios del producto.</li> <li>- Identificación de posibles tablas a usar.</li> </ul>
Sprint 4	<ul style="list-style-type: none"> <li>- Diseño de formulario Lectura Base de Datos.</li> <li>- Implementar funcionalidad de formulario Lectura Base de Datos</li> </ul>
Sprint 5	<ul style="list-style-type: none"> <li>- Diseño de formulario Definición de tipo de Datos</li> <li>- Implementar funcionalidad de formulario Definición de tipo de Datos</li> </ul>
Sprint 6	<ul style="list-style-type: none"> <li>- Diseño de formulario de Generación Plantillas</li> <li>- Implementar funcionalidad de formulario de Generación Plantillas</li> </ul>
Sprint 7	<ul style="list-style-type: none"> <li>- Diseño de formulario Generación de Código</li> <li>- Implementar funcionalidad de formulario Generación de Código</li> </ul>
Sprint 8	<ul style="list-style-type: none"> <li>- Pruebas funcionales de la aplicación</li> </ul>

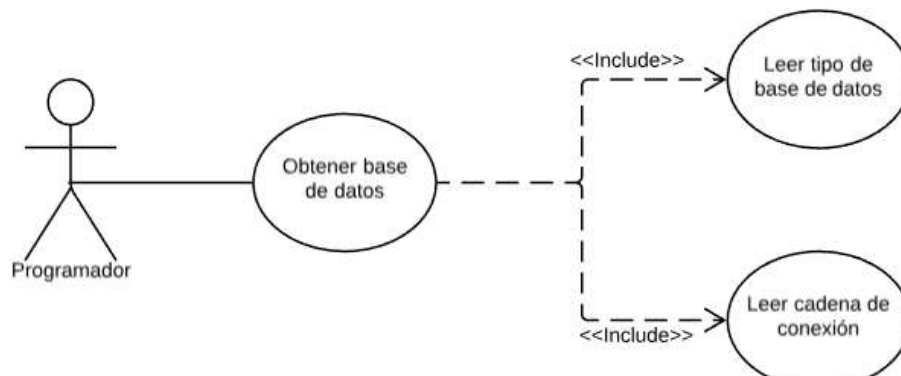
### 5. Caso de uso del negocio



**Figura 19.** Diagrama de Casos de Uso del Negocio

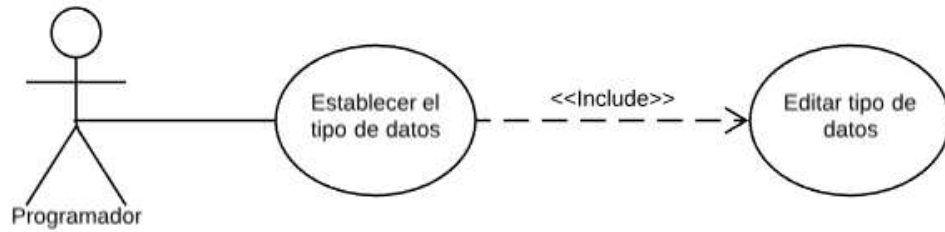
### 6. Caso de uso del sistema

#### 6.1. Obtener base de datos



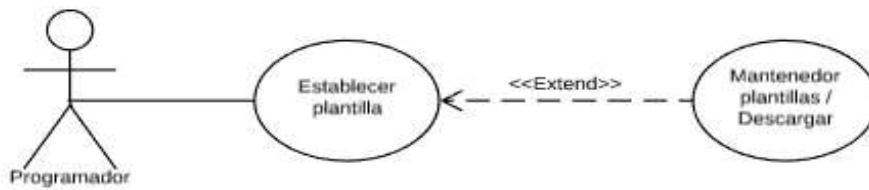
**Figura 20.** Diagrama del caso de uso - Obtener base de datos

## 6.2. Establecer tipo de datos



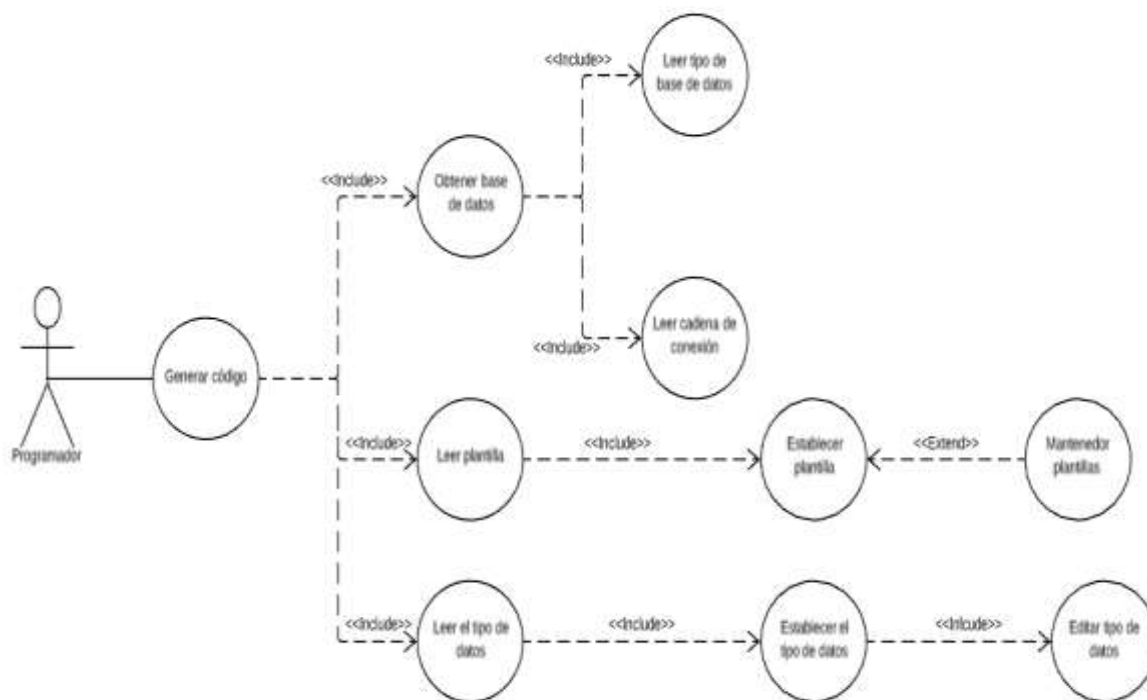
**Figura 21.** Diagrama del caso de uso - Establecer el tipo de datos

## 6.3. Crear plantillas



**Figura 22.** Diagrama de caso de uso - Crear plantillas

### 6.4. Generar código



**Figura 23.** Diagrama de caso de uso - Generar código

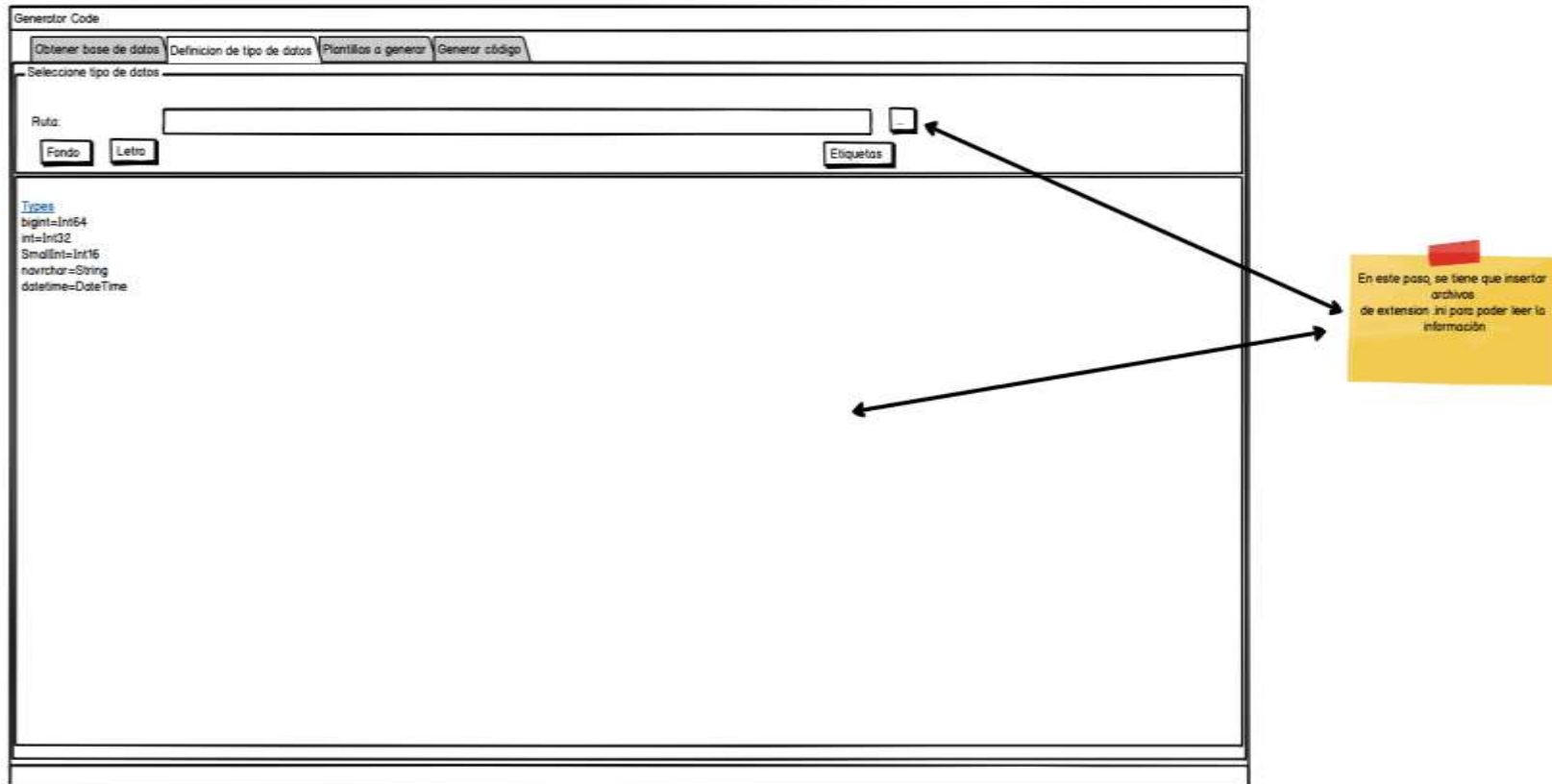
## 7. Especificación de los casos de uso del sistema.

<b>IDENTIFICADOR:</b> CU01	<b>NOMBRE:</b> Obtener base de datos		
<b>CATEGORÍA:</b> Core	<b>COMPLEJIDAD:</b> Alta	<b>PRIORIDAD:</b> Alta	
<b>ACTORES:</b> Desarrollador			
<b>PROPÓSITO:</b> El objetivo es obtener la base de datos a través de la cadena de conexión para seleccionar las tablas a generar.			
<b>PRECONDICIÓN:</b>			
<b>FLUJO BÁSICO:</b>  <p><b>B1.</b> El programador ejecuta el programa se encuentra el primer “tab” (Obtener base de datos).</p> <p><b>B2.</b> El sistema muestra un formulario con los datos que deberá ingresar el programador para poder obtener la base de datos, los datos son: Cadena de conexión y finalmente el botón conectar.</p> <p><b>B3.</b> El sistema muestra las tablas pertenecientes a la base de datos ingresada.</p> <p><b>B4.</b> El programador tiene que seleccionar las tablas con las cuales desea que se genere el código, con la opción de cambiar el nombre de la tabla en la columna “New Name” de la tabla generada. Luego debe dar clic en el botón “Mostrar Columnas &gt;&gt;”</p> <p><b>B5.</b> El sistema mostrará las columnas pertenecientes a las tablas seleccionadas, los datos a mostrar son: Name Table, Column Name, Type Name Complete y Primary Key.</p>			
<b>POSCONDICION:</b>			
<b>FLUJOS ALTERNATIVOS:</b>			
<b>REQUERIMIENTOS ESPECIALES O SUPLEMENTARIOS:</b>			

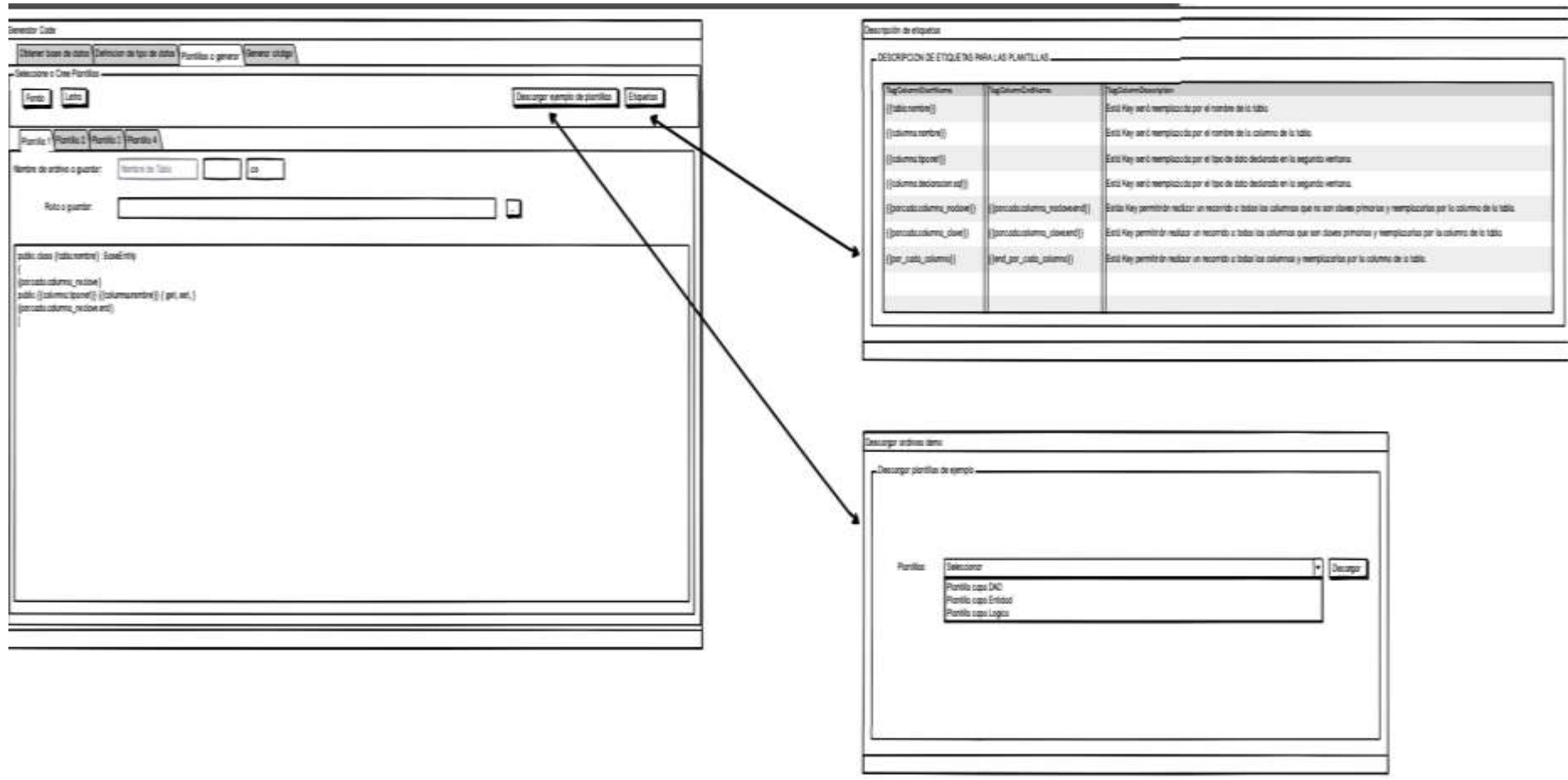


<b>IDENTIFICADOR:</b> CU02	<b>NOMBRE:</b> Generar código		
<b>CATEGORÍA:</b> Core	<b>COMPLEJIDAD:</b> Alta	<b>PRIORIDAD:</b> Alta	
<b>ACTORES:</b> Desarrollador			
<b>PROPÓSITO:</b> El objetivo es generar código según las tablas seleccionadas y las plantillas ingresadas.			
<b>PRECONDICIÓN:</b> Caso de uso obtener base de datos			
<b>FLUJO BÁSICO:</b>  <p><b>B1.</b> El programador elige el segundo “tab” (Definición de tipo de datos).</p> <p><b>B2.</b> El sistema mostrará la opción para descargar un archivo de ejemplo (en formato .txt pero se tendrá que convertir a .ini)</p> <p><b>B3.</b> El programador elegirá su archivo con extensión .ini el cual tiene que contener los tipos de datos del gestor de base de datos con su equivalente según el lenguaje a generar el código.</p> <p><b>B4.</b> El sistema mostrará el contenido del archivo el cual puede editarlo a su gusto y guardar los cambios para luego ser leído el contenido del archivo.</p> <p><b>B5.</b> El programador elige el tercer “tab” (Plantillas a generar).</p> <p><b>B6.</b> El sistema muestra las opciones para descargar modelos de plantillas de las distintas capas (DAO, Entity, Logic), así como la opción de ver las distintas keys que debería contener las plantillas y su descripción.</p> <p><b>B7.</b> El programador elegirá su archivo el cual tiene que contener la plantilla con el código a generar. Esa plantilla tiene que contener las keys las cuales serán reemplazadas por el nombre de tablas, nombre de columnas y el tipo de dato.</p> <p><b>B8.</b> El programador elige el cuarto “tab” (Generar código).</p> <p><b>B9.</b> El programador ingresa el nombre de los archivos a generar y elegirá la ruta de donde se van a guardar los archivos y finalmente dar clic en el botón “GENERAR CÓDIGO”.</p>			
<b>POSCONDICION:</b>			
<b>FLUJOS ALTERNATIVOS:</b>			
<b>REQUERIMIENTOS ESPECIALES O SUPLEMENTARIOS:</b>			





**Figura 25.** Diseño Mockups del segundo paso (Definir tipo de dato) del generador.



**Figura 26.** Diseño Mockups del tercer paso (Seleccionar plantilla) del generador.