PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

ESCUELA DE POSGRADO

REVISIÓN SISTEMÁTICA SOBRE GENERADORES DE CODIGO FUENTE Y PATRONES DE ARQUITECTURA

Tesis para optar el grado de Magíster en Informática con mención en Ciencias de la Computación que presenta

MARÍA ROSARIO HUARI CASAS

Dirigido por

DR. HÉCTOR ANDRÉS MELGAR SASIETA

San Miguel, marzo del 2020

Resumen.

Los proyectos de desarrollo de software, están sujetos a situaciones que, pueden ocasionar demora en la entrega del producto o generar aplicaciones de mala calidad, debido a deficiencias en la organización estructural del código y a la falta de integración de los componentes de software.

Para reducir el riesgo de demoras en la parte de la codificación de los programas, se puede hacer uso de herramientas informáticas que mejoren la productividad, entre estas herramientas se puede encontrar a los Generadores de Código Fuente (GCF), aplicaciones que generan código automáticamente, los cuales se utilizan en situaciones donde la lógica para armar un sistema es repetitiva; es decir las interfaces para el ingreso de datos, la conexión a las bases de datos, los reportes o salida de datos cumplen la misma lógica teniendo como única variante la estructura de datos.

Así mismo, para mejorar la calidad de los productos, antes de empezar a codificar los programas, es muy importante planificar y diseñar el patrón de arquitectura del aplicativo; tener un esquema de organización para agrupar, acoplar y encapsular los programas.

Mediante la revisión sistemática de la literatura se logra identificar patrones de arquitectura utilizados en la generación de código fuente de aplicaciones web; así como los principales *frameworks* y herramientas.

Las etapas de la revisión sistemática realizadas son: planeamiento, ejecución, reporte y divulgación. Así mismo, en las preguntas de investigación, se aplicaron los criterios del método PICOC, se realizaron estrategias de búsqueda y selección de fuentes haciendo uso de las principales bases de datos científicas o académicas. Los resultados de la revisión sistemática de la literatura, estuvieron orientadas a analizar los GCF, para comprender su importancia en el desarrollo de sistemas informáticos, así como analizar su relación con los patrones de arquitectura de software.

Finalmente, en base a la información obtenida, se llega a la conclusión de que los patrones de arquitectura tienen un papel importante en la generación de código fuente, ya que estandariza y organiza la aplicación en bloques o capas, de esta manera los desarrolladores tienen la opción de elegir herramientas estructuradas, reutilizables y eficientes.

Palabras Clave: generador de código fuente, GCF, patrones de arquitectura, *frameworks.*

Índice

Índice 1
Lista de Tablas 3
Lista de Ilustraciones 4
1 Introducción. 5
1.1 Definición del problema. 6
1.2 Objetivos. 10
1.2.1 Objetivos específicos. 10
1.3 Resultados Esperados. 10
1.4 Justificación. 13
1.5 Marco Conceptual. 14
1.5.1 Generador de Código Fuente (GCF). 14
1.5.2 Arquitectura de Software. 14
1.5.3 Patrón de Arquitectura. 15
1.5.4 Patrones de Diseño. 15
1.5.5 Frameworks de Aplicaciones. 16
2 Metodología. 17
2.1 Revisión Sistemática. 17
3 Protocolo de Revisión. 19
3.1 Preguntas de Investigación. 19
3.2 Estrategia de búsqueda y selección de fuentes. 22
3.2.1 Estrategia para derivar términos de búsqueda. 22
3.2.2 Selección de fuentes y recursos. 25
3.2.3 Documentación del proceso de búsqueda. 26
3.2.4 Criterios de inclusión / exclusión para la selección de estudios. 27
3.2.5 Listas de verificación y criterios de evaluación de calidad. 28
3.2.6 Proceso de selección. 30
3.3 Estrategia de extracción de datos. 30
3.3.1 Formularios de extracción de datos. 31
3.3.2 Procedimientos de extracción de datos. 33
3.4 Síntesis y análisis de los datos extraídos. 34

4 Ejecución de la revisión sistemática 37
4.1 Búsqueda y selección de estudios primarios. 37
4.2 Evaluación de la calidad de los estudios. 43
4.3 Extracción y Síntesis 44
4.3.1 Años de Publicación 45
4.3.2 Fuentes de Publicación. 46
4.3.3 Numero de Citas de los Estudios. 49
4.3.4 Centros de Investigación. 49
5 Resultados de la Revisión Sistemática de la Literatura 52
5.1 Generadores de Código Fuente y Patrones de Arquitectura (PI1). 52
5.1.1 Importancia de los Generadores de Código fuente (PI1a). 52
5.1.2 Importancia de los patrones de arquitectura en los GCF (PI1b). 57
5.1.3 Patrones de Arquitectura (PI1c) 60
5.1.4 Generadores de Código Fuente (PI1d) 63
5.2 Herramientas o componentes de los GCF (PI2) 66
5.2.1 Lenguajes de programación para la construcción de GCF (PI2a) 66
5.2.2 Lenguajes de Programación del código fuente generado (PI2b). 68
5.2.3 Gestores de Base de Datos usados en la GCF(PI2c) 71
5.2.4 Herramientas de modelado y librerías (PI2d). 73
5.2.5 Entornos de Desarrollo Integrado IDE (PI2e). 77
5.3 *Frameworks* (PI3) 78
5.4 Aplicaciones de los GCF (PI4) 86
6 Trabajos futuros y conclusiones 89
6.1 Conclusiones. 89
6.2 Trabajos futuros. 92
7 Bibliografía 94
ANEXOS 100
Anexo 01: Procesos de los *Frameworks*. 100
Anexo 02: Descripción de los procesos y métodos utilizados en la GCF. 107
Anexo 03 Otros gráficos de la selección inicial de estudios. 117
Anexo 04: Patrones de Arquitectura Identificados. 119
Anexo 05 Puntaje de Calidad de los Estudios: 127

Lista de Tablas

Tabla 1 Matriz de Marco Lógico. 13
Tabla 2 Etapas de la revisión sistemática. [19] 18
Tabla 3 Método PICOC para formular preguntas de investigación 20
Tabla 4 Definición de preguntas de investigación. 21
Tabla 5 Términos Derivados del PICOC 22
Tabla 6 Términos derivados de palabras clave en documentos relevantes. 23
Tabla 7 Términos derivados de deletreos alternativos y sinónimos 24
Tabla 8 Construcción de términos mediante el uso de Booleano OR 24
Tabla 9 Concatenación de términos mediante el uso de Booleano AND 24
Tabla 10 Procedimiento para la documentación del proceso de búsqueda. 27
Tabla 11 Escala de valores para las listas de control de calidad. 28
Tabla 12 Lista de verificación para estudios cuantitativos. 29
Tabla 13 Lista de verificacion para estudios cualitativos. 29
Tabla 14 Formulario de extracción de datos para estudios cualitativos. 32
Tabla 15 Formulario de extracción de datos para estudios cuantitativos 33
Tabla 16 Resumen de evidencia en generadores de código fuente y patrones de Arquitectura. 34
Tabla 17 Características/componentes/ requerimientos existentes. 35
Tabla 18 Resumen de Métodos/Procesos para la generación de Código Fuente. 35
Tabla 19 Resumen de ventajas y aplicaciones de los GCF. 36
Tabla 20 Selección inicial de estudios. 39
Tabla 21 Resultados de la selección inicial de estudios. 41
Tabla 22 Resultado de selección total de estudios por Base de datos. 42
Tabla 23- Extracción y síntesis de estudios 45
Tabla 24 Fuentes de Publicación y citas. 48
Tabla 25 Estudios por centros de investigación. 51
Tabla 26 Características de los GCF encontrados. 54
Tabla 27 Características de los Patrones de Arquitectura. 58
Tabla 28 Principales Patrones de Arquitectura de los GCF 61
Tabla 29 Principales Nombres de los GCF. 65
Tabla 30 Lenguajes de programación para la Generación de Código fuente 67
Tabla 31 Lenguajes de programación generados. 69
Tabla 32 Principales gestores de bases de datos encontradas. 72
Tabla 33 Principales herramientas de modelado. 74
Tabla 34 Librerías de Desarrollo encontradas. 75
Tabla 35 Entorno de Desarrollo Integrados. 78
Tabla 36 Principales Frameworks para la GCF. 80
Tabla 37 Aplicaciones de los generadores de código. 87
Tabla 38 Frameworks. 106
Tabla 39 Cadenas de búsqueda para la selección de estudios. 118
Tabla 40 Puntaje de calidad de los estudios seleccionados (i). 127
Tabla 40 Puntaje de calidad de los estudios seleccionados (ii). 128

Lista de Ilustraciones

Ilustración 1 Árbol de problemas (causas y efectos). ... 7
Ilustración 2 Fases del Proceso de Selección ... 39
Ilustración 3 Formulario de proceso de selección inicial de estudios de la Herramienta StART ... 40
Ilustración 4 Visión general de la selección inicial por BBDD ... 40
Ilustración 5 Proceso de selección de estudios primarios. ... 42
Ilustración 6 Resultado de la Selección con la herramienta StArt ... 43
Ilustración 7- Nro. de publicaciones acumulado (Fuente Propia) ... 46
Ilustración 8 Estudios por tipo de publicación ... 48
Ilustración 9 Nro. de citas por estudio (Fuente Propia). ... 49
Ilustración 10- Publicaciones por país ... 50
Ilustración 11- Importancia de los GCF ... 55
Ilustración 12- Importancia de los patrones de arquitectura ... 58
Ilustración 13- Importancia de los patrones de arquitectura ... 67
Ilustración 14- Lenguajes de programación generados por el GCF ... 70
Ilustración 15 - Proceso de generación de Código Fuente. ... 108
Ilustración 16 Modelo de generador de código. ... 110
Ilustración 17 Proceso completo para la generación de código fuente. ... 112
Ilustración 18 Proceso de GCF del estudio nro. 28 ... 115
Ilustración 19 Selección inicial con Mendeley. ... 117
Ilustración 20 Selección inicial de estudios con StArt. ... 117
Ilustración 21 El patrón Arquitectónico MVVM [24] ... 119
Ilustración 22 Patrón de Arquitectura MVC. ... 120
Ilustración 23 Arquitectura dirigida por modelos MDA ... 120
Ilustración 24 Arquitectura J2EE de un sistema de aplicación WEB. ... 121
Ilustración 25 Modelo de Vistas de Arquitectura 4+1 ... 122
Ilustración 26 Vista de Arquitectura de 3 niveles. [12] ... 123
Ilustración 27 Arquitectura Genesys [44] ... 123
Ilustración 28 Arquitectura Titán para desarrollo de aplicaciones móviles. [18] 124
Ilustración 29 Arquitectura NHibernateMapper [45] ... 125
Ilustración 30 Arquitectura OPRF [46] ... 125

1 Introducción.

Los requerimientos para desarrollar software moderno van cambiando constantemente, el tiempo para desarrollar es cada vez más corto y la complejidad va aumentando; los usuarios esperan que las aplicaciones interactúen con otras aplicaciones, dispositivos, servicios y funciones de red para realizar autenticación, obtener y publicar información, además requieren que las aplicaciones se instalen en una serie de entornos como la nube y funcionen en diferentes dispositivos como PC, *smartphones*, *tablets*, portátiles, televisores, etc. [1].

Por otro lado, la arquitectura de software, es considerada como la parte central del desarrollo de sistemas de información modernos; tiene como objetivo desarrollar sistemas de forma eficiente, estructurada y con capacidad de reúso [2]. La arquitectura es una parte del diseño de software que a su vez es parte del desarrollo de software (requerimientos, diseño, implementación, pruebas y mantenimiento) [3].

Para agilizar el desarrollo de aplicaciones, simplificar la necesidad de comprender, hacer uso de las técnicas y herramientas más eficientes al momento de hacer la codificación de los programas; surgen los generadores de código, los cuales se utilizan en situaciones donde la lógica para armar un sistema es repetitivo; es decir, las interfaces para el ingreso de datos, la conexión a las bases de datos, los reportes o salida de datos cumplen la misma lógica teniendo como única variante la estructura de datos. No se trata de crear plantillas para hacer formularios, sino, más bien que se genere código fuente para tener una versión completa o casi completa de todo el sistema basado en una arquitectura [4].

Un generador de código fuente (GCF) ayuda a construir programas de manera automática y estandarizada, optimizando el tiempo y el costo de

producción de software; evitando que los desarrolladores de software tengan que escribir código a mano [5].

El presente trabajo trata de la revisión sistemática de generadores de código fuente para aplicaciones web, que utilizan patrones de arquitectura, *frameworks* y otras herramientas como lenguajes de programación, base de datos, etc. con el fin de que los desarrolladores de software, cuenten con la información que necesiten y así poder elegir la opción que más se adecue a sus requerimientos; así, poder mejorar las deficiencias que existen cuando se trata de construir software.

1.1 Definición del problema.

Los desarrolladores, analistas, arquitectos, Ingenieros de software que están involucrados en el desarrollo de software, requieren de herramientas informáticas que les permita optimizar el proceso de construcción de sistemas informáticos; para que la tarea de desarrollo se reduzca a ensamblar estas herramientas debidamente probadas [6].

El avance de la tecnología ofrece una serie de herramientas que ayudan a resolver las dificultades a los que se enfrentan el equipo de desarrollo, permitiendo acelerar el proceso de construcción de software, reutilizar el código y promover buenas prácticas [7]; el reto está en seleccionar la mejor opción entre estas herramientas para poder obtener productos que satisfagan los requerimientos del usuario final.

El problema es que, a pesar de que existen soluciones, el equipo de desarrollo se enfrenta a factores que pueden generar demoras en el desarrollo de aplicaciones, por deficiencias en la estructuración, funcionamiento e integración de los componentes del software.

De acuerdo al árbol de problemas planteado en la

Ilustración 1, se identificó el problema central y los problemas causa que se describen a

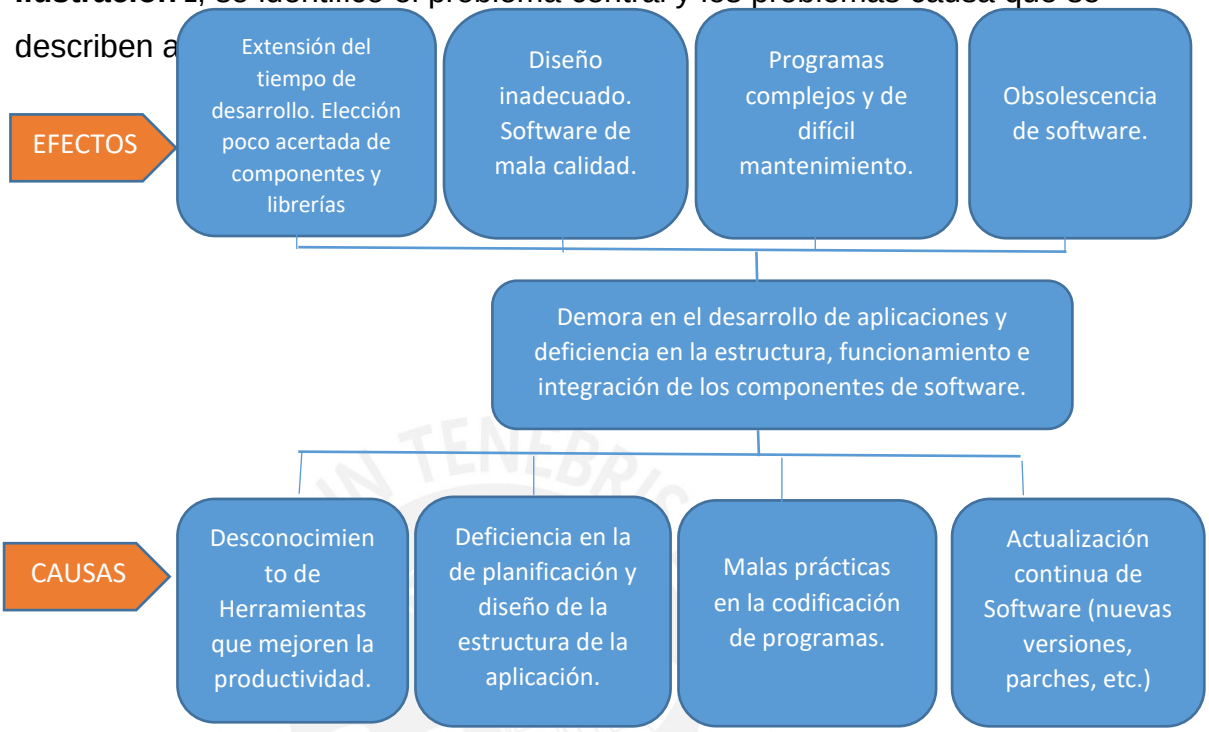

Ilustración 1 Árbol de problemas (causas y efectos).

En el análisis causa efecto se ha identificado los siguientes casos:

1. El equipo de desarrollo no cuenta con suficiente información sobre herramientas que les permita mejorar su productividad, desconocen la existencia de herramientas para crear aplicaciones de forma rápida, como consecuencia el proyecto se puede enfrentar a demoras en el proceso de desarrollo.

   Una de las herramientas que tiene que ver con el desarrollo y generación de código fuente, es el gestor de base de datos. El problema es que los desarrolladores pasan mucho tiempo creando programas manuales para hacer operaciones básicas de base de datos como insertar, actualizar, eliminar, etc. teniendo la opción de realizarlas automáticamente [6]. Por

eso es necesario revisar y evaluar información acerca de los gestores de base de datos en los GCF.

Por otra parte, los Framework generalmente se utilizan para agilizar el proceso de construcción de software, cumplen un papel muy importante en el proceso de desarrollo de un GCF [7] . Un buen *framework* es aquel que resuelve los problemas técnicos más complejos; esto permite que los desarrolladores se concentren más en el diseño y desarrollo del dominio de la aplicación [8], el problema es que a pesar de sus ventajas gran parte de los desarrolladores no lo consideran.

Los lenguajes de programación que se utilizan para la construcción de GCF no necesariamente generan código en el mismo lenguaje [9].

2. Otra problemática es la deficiente planificación y diseño de la estructura de la aplicación, es decir, la falta de definición del patrón de arquitectura del software puede generar un diseño inadecuado y por consiguiente un producto de mala calidad.

   Por otra parte, los patrones arquitectónicos son una pieza clave en la construcción y generación de código fuente ya que forman parte de estructura del sistema [10]; los *frameworks* ayudan a desarrollar cada componente de la estructura que conforma el generador [11]. Por lo tanto, se debería lograr la relación o combinación adecuada sobre el uso de estas herramientas.

   El problema es que no se planifica ni documenta el uso de patrones de arquitectura antes de empezar la codificación de los programas fuentes, por lo general este paso es obviado. La construcción de sistemas de software requiere de un alto nivel de abstracción, de manera que se puede encontrar formas de reutilizar el software existente para aumentar el rendimiento y productividad; los GCF son herramientas que se utilizan en una o más fases del ciclo de vida de desarrollo de software, pueden ser utilizadas para lograr software de buena calidad por eso requieren una

buena implementación basado en una arquitectura tomando en cuenta la importancia de utilizar patrones de arquitectura en el desarrollo de estas herramientas [5].

3. Las malas prácticas en la codificación de programas, como la falta de estandarización en la nomenclatura de variables, uso de componentes, etc., traen como consecuencia aplicaciones de mala calidad y de difícil mantenimiento [12]; es conveniente mantener una estructura que garantice la estandarización y reutilización del código.

4. La actualización continua de software (nuevas versiones, parches, etc.), hace que los aplicativos se vuelvan obsoletos; este fenómeno no se puede evitar, por lo que es necesario proporcionar herramientas que permitan actualizar el código de las aplicaciones de manera automática.

   Hoy en día existen herramientas que permiten generar código fuente; pero su uso está limitado debido a que la habilidad de usar herramientas productivamente es un sello distintivo de un Ingeniero de software especializado [10] [13]; por otra parte, existe aplicaciones que no están estandarizadas. Además, hay dificultad para identificar el patrón de arquitectura que se genera debido a la confusión de conceptos entre patrones de arquitectura, patrones de diseño, estilos arquitectónicos y *frameworks* [10]. Mediante la Revisión sistemática; se pretende identificar los generadores de código fuente, patrones de arquitectura, herramientas, componentes y marcos; que proporcionen información útil a los desarrolladores, para que puedan elegir la opción más se adecue a sus necesidades; así, de esta manera, el equipo de software puede disminuir los problemas que se presentan a la hora de construir aplicaciones web.

1.2 Objetivos.

El objetivo principal, es realizar una revisión sistemática de la literatura para investigar las principales herramientas de generación de código fuente y patrones de arquitectura, que proporcionen información útil a los desarrolladores con el fin de reducir las deficiencias en la estructura, funcionamiento e integración de los componentes de software.

1.2.1 Objetivos específicos.

- Demostrar que los GCF son importantes en la construcción de aplicaciones informáticas.
- Demostrar la importancia del uso de patrones de Arquitectura en la generación de código fuente.
- Identificar las herramientas utilizadas para la GCF.
- Identificar los principales *frameworks* usados para la GCF.
- Averiguar las ventajas y aplicaciones de los GCF.

1.3 Resultados Esperados.

Para describir los resultados esperados se utilizó la matriz de marco lógico que se plantea en la

Objetivo General	Resultados Finales	Medios De Verificación
Realizar una RSL para investigar las principales herramientas de GCF y patrones de arquitectura, que proporcionen información útil a los desarrolladores con el fin de reducir las deficiencias en la estructura, funcionamiento e integración de los componentes de software.	Encontrar las principales herramientas de generación de código fuente y patrones de arquitectura para aplicaciones web.	Resultados de búsqueda en las principales bases de datos de investigación
Objetivos Específicos	**Resultados Intermedios**	**Medios De Verificación**
Demostrar la importancia de los GCF en la construcción de aplicaciones informáticas	Estudios que evidencien la importancia de los GCF y patrones de arquitectura.	Resultados de las búsquedas en las principales bases de datos
Demostrar la importancia de los Patrones de Arquitectura en la generación de código fuente.	Se espera encontrar patrones de arquitectura con información necesaria para demostrar su importancia en la generación de código fuente.	Resultados de las búsquedas en las principales bases de datos
Identificar patrones de arquitectura que se utilicen el la GCF con el fin de mejorar la de planificación y diseño de la estructura de la aplicación.	Estudios que identifiquen patrones de arquitectura.	Resultados de búsqueda en las principales bases de datos de investigación
Identificar GCF y *frameworks* con el fin de disminuir las malas prácticas y mejora la estandarización de código.	Estudios que contengan GCF y *frameworks.*	Resultados de búsqueda en las principales bases de datos de investigación
Identificar Herramientas utilizadas para la GCF con el fin de proporcionar información que ayude a mejorar la productividad.	Encontrar lenguajes de programación, base de datos e IDE.	Resultados de las búsquedas en las principales bases de datos

Tabla 1. Resumiendo, se espera los siguientes resultados:

- Encontrar estudios que evidencien la importancia del uso de los generadores de código en la construcción de aplicaciones web.
- Se espera encontrar patrones de arquitectura con información necesaria para demostrar su importancia en la generación de código fuente.
- Encontrar generadores de código fuente que se utilizan en los estudios.

- Encontrar los patrones de arquitectura que se utilizan en la generación de código fuente.
- Encontrar lenguajes de programación utilizados en la generación de código fuente.
- Encontrar gestores de base de datos que se han utilizado para la GCF.
- Encontrar los entornos de desarrollo integrado utilizados en la GCF.
- Adicionalmente se espera obtener información sobre la aplicación de los GCF.

Objetivo General	**Resultados Finales**	**Medios De Verificación**
Realizar una RSL para investigar las principales herramientas de GCF y patrones de arquitectura, que proporcionen información útil a los desarrolladores con el fin de reducir las deficiencias en la estructura, funcionamiento e integración de los componentes de software.	Encontrar las principales herramientas de generación de código fuente y patrones de arquitectura para aplicaciones web.	Resultados de búsqueda en las principales bases de datos de investigación
Objetivos Específicos	**Resultados Intermedios**	**Medios De Verificación**
Demostrar la importancia de los GCF en la construcción de aplicaciones informáticas	Estudios que evidencien la importancia de los GCF y patrones de arquitectura.	Resultados de las búsquedas en las principales bases de datos
Demostrar la importancia de los Patrones de Arquitectura en la generación de código fuente.	Se espera encontrar patrones de arquitectura con información necesaria para demostrar su importancia en la generación de código fuente.	Resultados de las búsquedas en las principales bases de datos
Identificar patrones de arquitectura que se utilicen el la GCF con el fin de mejorar la de planificación y diseño de la estructura de la aplicación.	Estudios que identifiquen patrones de arquitectura.	Resultados de búsqueda en las principales bases de datos de investigación
Identificar GCF y *frameworks* con el fin de disminuir las malas prácticas y mejora la estandarización de código.	Estudios que contengan GCF y *frameworks.*	Resultados de búsqueda en las principales bases de datos de investigación
Identificar Herramientas utilizadas para la GCF con el fin de proporcionar información que ayude a mejorar la productividad.	Encontrar lenguajes de programación, base de datos e IDE.	Resultados de las búsquedas en las principales bases de datos

Tabla 1 Matriz de Marco Lógico.

1.4 Justificación.

El uso de una buena arquitectura, reduce el costo y tiempo requeridos para desarrollar aplicaciones, disminuye errores en la etapa de control de calidad, mantiene la aplicación en ejecución y la actualiza para cumplir con los requisitos cambiantes y solucionar problemas [3]. Además, simplifica el soporte técnico y la administración por parte del usuario.

El uso adecuado de la arquitectura de software tendrá como objetivo equilibrar factores como la seguridad, sostenibilidad, robustez y confianza; también debe ejecutarse dentro de los requerimientos para satisfacer al cliente con un tiempo de respuesta óptimo [2] [1]

Las razones principales para usar generadores de código fuente son: simplificación, portabilidad, consistencia, capacidad de reúso; así mismo, junto con otras herramientas mejoran la productividad; ayudando a reducir los problemas en la estructura, funcionamiento e integración de los componentes de software [5].

Para reducir los problemas que se presentan en la construcción de software, se debe elegir herramientas adecuadas a los requerimientos del desarrollador, se realiza una revisión sistemática de la literatura sobre generadores de código fuente, patrones de arquitectura y otras herramientas de desarrollo; para proporcionar información útil a los desarrolladores, para que puedan elegir herramientas entre varias opciones, de esta manera reducir el tiempo de desarrollo de las aplicaciones y ayudar a mejorar las deficiencias en cuanto a estructura, funcionamiento e integración de los componentes de software.

1.5 Marco Conceptual.

1.5.1 Generador de Código Fuente (GCF).

La programación automática o generación automática de programas ha surgido como un enfoque para disminuir la carga de trabajo de los programadores, este tipo de herramientas reducen la cantidad de código fuente tedioso, repetitivo, aburrido y propenso a errores [14].

Un generador de código fuente, es una herramienta informática que genera un tipo particular de código de programación de computadora en un determinado lenguaje de programación, o una herramienta que genera código fuente para generar clases, métodos e interfaces para un desarrollo de software más fácil, rápido y estandarizado y que haga operaciones básicas (insertar, actualizar, eliminar y buscar), a partir de un modelo de datos o un esquema de base de datos [15].

Los Generadores de código fuente, tienen como objetivo principal reducir en gran medida el tiempo de desarrollo y mantenimiento; minimizar los errores, estandarizar y mejorar la coherencia del código, reducir costos debido a la reutilización de las herramientas; ya que estos aplicativos generan el código fuente de manera automática, para diferentes lenguajes de programación, diferentes plataformas y para múltiples motores de bases de datos [5]

1.5.2 Arquitectura de Software.

La Arquitectura de Software es una descripción de subsistemas y componentes de un sistema de software y como se relacionan entre ellos [2]. Los subsistemas y componentes son especificados en diferentes vistas que muestran las propiedades funcionales y no funcionales más importantes del sistema de software. La arquitectura de software de un sistema es un artefacto (resultado de la actividad de diseño del software) a un alto nivel de abstracción [16].

La arquitectura de software implica definir una solución estructurada que satisfaga todos los requisitos técnicos y operacionales y, a la vez, optimizar los atributos comunes de calidad como rendimiento, seguridad y capacidad de administración. Además, implica una serie de decisiones basadas en una amplia gama de factores, y cada una de esas decisiones puede tener un considerable impacto sobre la calidad, rendimiento, mantenimiento y éxito general de ese software [1].

1.5.3Patrón de Arquitectura.

Según [2] expresa un esquema de organización estructural esencial para un sistema de Software, consta un conjunto de sub sistemas predefinidos, especifica sus responsabilidades e incluye reglas y guías para organizar las relaciones entre ellos. Son patrones de alto nivel que se utilizan en la definición de la estructura y organización de un sistema de software.

Un patrón de arquitectura encapsula los elementos y relaciones que existen entre ellos, abstrae su comportamiento para poder obtener una configuración de componentes que satisfaga ciertas necesidades.

Tienen un mayor grado de abstracción comparado con los patrones de diseño. Afectan a la estructura global del sistema. Entre estos, se puede mencionar: *Layers*, *pipes and filters*, *blackboard*, broker, *microkernel*, MVC, etc.

1.5.4Patrones de Diseño.

Brindan una solución probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Los patrones de diseño facilitan la reutilización de arquitecturas y diseños de software exitosos; están orientados como organizar el código fuente, como interactúan las diferentes partes o componentes del software [2] .

Son patrones de bajo nivel que se utilizan para definir de mejor manera los componentes de un sistema de software. Entre estos se puede mencionar: *Interpreter*, *prototype*, *builder*, f*acade*, *iterator*, *adapter*, *bridge*, *state*, etc.

1.5.5Frameworks de Aplicaciones

Según (S4) [17], esta clase de marcos ha sido llamada por muchos nombres, incluyendo marcos de infraestructura de software y marcos arquitectónicos. Estos marcos son colecciones de bibliotecas de software que proporcionan servicios de infraestructura para que otras aplicaciones las utilicen en forma de objetos de software. Algunas clases notables proporcionadas por estos marcos son IGUI, Widgets, servicios de red, servicios web, correo electrónico y otros servicios de mensajería. Estos marcos proporcionan un código ejecutable destinado a fines generales, y se pueden usar para crear aplicaciones de software sofisticadas al usar su funcionalidad de alto nivel. Sin embargo, no resuelven problemas que pertenecen a un determinado dominio o negocio, que es el siguiente nivel en la escala de abstracción del software, es decir, un marco de infraestructura de software puede considerarse como una caja de herramientas de propósito general.

Según (S9) [18] Los marcos de aplicación se consideran aplicaciones semi completas y reutilizables que, cuando son especializadas, producen aplicaciones personalizadas dentro de un dominio específico. Están compuestas por una colección de clases abstractas y concretas.

Un marco de software es una colección de bibliotecas de software reutilizable para un sistema de software (o subsistema). Esto se expresa como un conjunto de clases abstractas y la forma en que sus instancias colaboran para un tipo específico de software. Un marco de software puede incluir programas de soporte, bibliotecas de códigos, un lenguaje de scripting u otro software para ayudar a desarrollar los diferentes componentes de un proyecto de software. Varias partes del marco pueden estar expuestas a través de una interfaz de programación de aplicaciones (API).

2 Metodología.

2.1 Revisión Sistemática.

Se aplica la metodología planteada por B. Kitchenham [19] quien considera que la revisión sistemática es un medio para identificar, evaluar e interpretar toda la investigación disponible relevante a una pregunta de investigación particular, área temática o fenómeno de interés.

Los estudios que contribuyen a una revisión sistemática se llaman estudios primarios, una revisión sistemática es una forma de estudio secundario.

Algunas de las características que diferencian una revisión sistemática de una revisión de literatura especializada convencional son:

• Las revisiones sistemáticas comienzan definiendo un protocolo de revisión que especifica la pregunta de investigación que se aborda y los métodos que se utilizarán para realizar la revisión.

• Se basan en una estrategia de búsqueda definida que tiene como objetivo detectar la mayor cantidad posible de literatura relevante.

• Documentan su estrategia de búsqueda para que los lectores puedan evaluar su rigor, la integridad y repetitividad del proceso (teniendo en cuenta que las búsquedas de bibliotecas digitales son casi imposibles de reproducir).

•Requieren criterios explícitos de inclusión y exclusión para evaluar cada posible estudio primario.

• Especifican la información que se obtendrá de cada estudio primario, incluidos los criterios de calidad para evaluar cada estudio primario.

La **Tabla 2** muestra las etapas de la revisión sistemática:

Etapas de la Revisión sistemática	**Temas**
Planeamiento de la revisión.	Identificación de la necesidad de revisión.
	Encargar una revisión.
	Formulación de las preguntas de investigación.
	Desarrollo de un protocolo de revisión.
	Evaluar el protocolo de revisión.
Ejecución de la revisión.	Identificación de la Investigación.
	Selección de estudios primarios.
	Evaluación de la calidad de los estudios.
	Extracción de datos y seguimiento.
	Síntesis de datos.
Reporte y divulgación.	Especificación de los mecanismos de difusión.
	Formateo del reporte principal de la revisión.
	Evaluación de los reportes

Tabla 2 Etapas de la revisión sistemática. *[19]*

3 Protocolo de Revisión.

3.1 Preguntas de Investigación.

Las preguntas de investigación se formularán según los criterios del método PICOC sugerido por PETTICREW, M.; ROBERTS, H. [20]. Para el criterio *Población* se hace la pregunta: ¿Cuál es población de interés para esta investigación? La búsqueda estaría dada por las palabras: Generadores de código fuente. Para el criterio de *Intervención* se plantea la siguiente pregunta: ¿Qué se quiere saber a cerca de los generadores de código Fuente? La búsqueda se realizará usando los términos: Arquitectura/patrón de arquitectura o patrón de diseño que se genera, las herramientas y tecnologías aplicadas y las metodologías o métodos que usan los generadores de código. Para el criterio de *Comparación* la pregunta vendría a ser: ¿Qué es lo que se quiere comparar? El estudio no incluye la comparación de arquitecturas o patrones, más bien se enfoca en una visión general del estado actual de los GCF; por lo tanto, no se toma en cuenta el criterio de comparación. Para el criterio de *Salidas*(output) no se define resultados esperados porque la investigación no se centra en las salidas para ser evaluados (solo para clasificar las búsquedas).

Con respecto al criterio *Contexto*, los generadores de código se desarrollan en la industria de software, universidades, y los usuarios que lo utilizan como desarrolladores de software, gerentes, estudiantes, etc. Para este estudio se considera como contexto estudios relacionados con aplicaciones web La ***Tabla 3*** muestra la aplicación del método PICOC.

La formulación de las preguntas de investigación tiene una gran importancia en la revisión sistemática. [19] Con ellas se puede identificar y determinar los objetivos y actividades de la investigación.

CRITERIO	PICOC
POPULATION (población)	Generación de código fuente.
INTERVENTION (intervención)	Patrones de Arquitectura, herramientas de desarrollo de software, marcos de trabajo, metodologías, base de datos.
COMPARISON (comparación)	No aplica al estudio.
OUTCOMES (salida)	No se considera para análisis o evaluación.
CONTEXT (contexto)	industria de software, universidades, desarrolladores de software, aplicaciones web.

Tabla 3 Método PICOC para formular preguntas de investigación

A través del presente trabajo se pretende proporcionar información sobre herramientas de generación de código que ayuden a mejorar la productividad de las aplicaciones; así como mejorar la planificación y diseño de la estructura de las aplicaciones a través del uso de Patrones de Arquitectura que estandarizan los programas.

En este sentido, se deberá obtener evidencia sobre estudios a cerca de generadores de código fuente y patrones de arquitectura, tomando en cuenta los objetivos específicos mencionados en el punto 1.2.2 se formulan las siguientes preguntas:

Población	Intervención	Contexto	Preguntas
Generador de Código Fuente.	Patrones de Arquitectura.	**Aplicaciones web**	**PI1a. ¿Cuál es la importancia de los GCF en la construcción de aplicaciones?** **PI1b. ¿Por qué es importante utilizar patrones de arquitectura para generar código fuente?** **PI1c. ¿Qué patrones de arquitectura utilizan los GCF?** **PI1d. ¿Qué GCF se han desarrollado?** **PI4. ¿Qué aplicaciones tienen los GCF en el desarrollo de Sistemas?**

Población	Intervención	Contexto	Preguntas
	Lenguajes		**PI2a**. ¿Qué Lenguajes de programación son usados por los GCF?
	Base de Datos		**PI2b. ¿Para qué gestores de base de datos se genera código fuente?**
	Herramientas		**PI2c**. ¿Qué Herramientas utilizan los GCF?
	IDE		**PI2d.** ¿Qué IDE se utilizan en los GCF?
	Frameworks		**PI3**. ¿Qué *frameworks* se utilizan para la GCF?

Tabla 4 Definición de preguntas de investigación.

Pregunta 1 (PI1):

PI1a. ¿Cuál es la importancia de los GCF en la construcción de Sistemas de Software?

PI1b. ¿Por qué es importante utilizar patrones de arquitectura para generar código fuente?

PI1c. ¿Qué patrones de arquitectura utilizan los GCF?

PI1d. ¿Qué GCF se han desarrollado?

Pregunta 2 (PI2):

PI2a. ¿Qué lenguajes de programación se utilizan para construir los GCF?

PI2b. ¿Qué lenguajes de programación generan los GCF?

PI2c. ¿Para qué gestores de base de datos se genera código fuente?

PI2d. ¿Qué herramientas y librerías son usados para desarrollar generadores de código fuente?

PI2e. ¿Qué entornos de desarrollo utilizan los gestores de código fuente?

Pregunta 3 (PI3):

PI3. ¿Qué *frameworks* se han empleado para la generación de código fuente?

Pregunta 4 (PI4):

PI4. ¿Cuáles son las principales aplicaciones de los GCF como herramienta para desarrollo de sistemas informáticos?

3.2 Estrategia de búsqueda y selección de fuentes.

3.2.1 Estrategia para derivar términos de búsqueda

Se obtiene los principales términos de búsqueda de las preguntas de investigación, identificando población, intervención, comparación, resultado y contexto; como se muestra en la

Tabla 5 Términos Derivados Del PICOC

Concepto	Descripción
Población	**Source code generator**
Intervención	Architectural pattern, tools1, frameworks, databases, IDE, programming language
Comparación	No se aplica.
Salidas	No se aplica.
Contexto	web application, software developers. Academic and software industry.

Tabla 5 Términos Derivados del PICOC

I) Derivar los principales términos o frases de búsqueda de las preguntas de investigación identificando Población, Intervención, comparación, Resultado y Contexto.

II) Encontrar palabras o frases clave en artículos relevantes, que no necesariamente son parte de los estudios primarios; es decir, documentos relacionados a generadores de código o patrones de arquitectura. La **Tabla 6**. Contiene esta información.

Titulo	Autor	Palabras clave
A Model-Driven Method for the Development of Web Applications User Interaction Layer	Ricardo A.C. de Souza Roberto S.M. de Barros (2008)	Web Applications, Model-Driven Development
Applying input-output tree to the implementation of a rapid prototyping tool for java web applications - Hsiao_CHU.	CHUN-FENG HSIAO, AND CHIH-PING CHU (2014)	automatic programming, CASE, software reuse
Estilos y Patrones en la Estrategia de Arquitectura de Microsoft	Carlos Reynoso – Nicolás Kicillof (2004) *	Framework, styles and patterns, idioms,
Líneas de Productos Software: Generando Código a Partir de Modelos y Patrones	Carlos Alberto Gaitán Peña (2017) *	LSP, Arquitectura Dirigida por Modelos, Patrones de Diseño, Generadores, Usabilidad, Bases de Datos Transaccionales.
Identificación y Clasificación de Patrones en el Diseño de Aplicaciones Móviles.	Darío Yorio*	design patterns, architectural patterns
Arquitectura de software para aplicaciones Web	Juan Tahuiton Mora (2011) *	Arquitectura aplicaciones web, patrón mvc, vistas.
An extensible, maintainable and elegant approach to hardware source code generation in Reconfig - P	Van Nguyen*, David Kearney, Gianpaolo Gioiosa (2010)	Source code generator, design patterns.
A framework for automatic generation of web-based data entry applications based on XML	Volker Turau (2002)	Web-based data entry, frameworks, automated prototyping
GENERACIÓN AUTOMÁTICA DE CÓDIGO BAJO EL PATRÓN MVC A PARTIR DE ESQUEMAS PRECONCEPTUALES	Carlos Mario Zapata Jaramillo, Gloria Lucía Giraldo Gómez, John Jairo Chavera Mojica (2011)	generación automática de código, MVC, herramienta CASE, regla heurística

Tabla 6 Términos derivados de palabras clave en documentos relevantes.

III) Encontrar ortografías y sinónimos alternativos para los términos y/o frases de búsqueda con la ayuda de un diccionario de sinónimos e internet. También consultar con algún experto en el contenido del tema, en la **Tabla 7** se muestra los sinónimos de los términos seleccionados.

source code generator	automatic generation of source code, automatic source code generation, SCG, automatic programming.
architecture	design patterns, architectural patterns, software architectural design patterns
Tools (development)	frameworks, programming language, database, IDE.

Tabla 7 Términos derivados de deletreos alternativos y sinónimos

IV) Utilizar OR booleano para construir cadenas de búsqueda a partir de los términos de búsqueda identificados en (i), (ii) y (iii) (**Tabla 8**).

source code generator OR automatic generation of source code OR automatic source code generation OR SCG OR automatic programming
architecture OR architecture patterns OR software architectural design patterns.
Web OR web application
framework OR database OR IDE OR programming language

Tabla 8 Construcción de términos mediante el uso de Booleano OR

V) Utilizar AND booleano para concatenar los términos de búsqueda y restringir la investigación (**Tabla 9**).

("Source code generat*" or "automatic generation of source code" or "automatic source code generat*" or SCG or "automatic programming") and ((design or architect*) and patterns)
("Source code generat*" or "automatic generation of source code" or "automatic source code generat*" or SCG or "automatic programming") and (architect* or "architect* pattern") and (web or "web application") and (frameworks or databases or languages or IDE or tool)

Tabla 9 Concatenación de términos mediante el uso de Booleano AND

3.2.2Selección de fuentes y recursos.

Fase de búsqueda primaria.

Se debe buscar bases de datos en línea, motores de búsqueda, actas de congresos y literatura gris (informes técnicos, tesis de maestrías y doctorados). Tratar de hacer una búsqueda exhaustiva para lograr la información precisa y así no perder evidencias [21].

Las bases de datos se tomarán teniendo como referencia revisiones anteriores; por lo que las búsquedas electrónicas serán:

Base de Datos en Línea:

- IEEE XPLORE[1]
- ACM Digital Library[2]
- ProQuest Computing[3]
- Web of Science[4]
- SCOPUS[5]
- Springer link[6]
- Mendeley[7]

Motores de búsqueda en línea:

- Cite Seer (citeseer.ist.psu.edu).
- Google Scholar (scholar.google.com).

informes técnicos

- Búsqueda en principales buscadores de internet (Google, Bing).
- Solicitar información a los autores de los estudios primarios por correo electrónico.

[1] http://ieeexplore.ieee.org/
[2] http://dl.acm.org/
[3] http://search.proquest.com/
[4] http://apps.webofknowledge.com/
[5] https://www.scopus.com/
[6] http://link.springer.com/
[7] https://www.mendeley.com/

Fase de búsqueda secundaria

Teniendo en cuenta que la revisión es un proceso iterativo, esta fase se realiza para complementar la fase de búsqueda inicial; se tomarán en cuenta las referencias de los estudios de la fase de búsqueda inicial, si son relevantes, se podrán agregar a la lista de estudios primarios; también se hará una revisión de las citas, de ser necesario algunos investigadores de estudios observados serán contactados. [21]

3.2.3Documentación del proceso de búsqueda.

La documentación del proceso de búsqueda proporciona transparencia, replicación del proceso de búsqueda y evita el efecto de sesgo. Se seguirá los procedimientos de documentación proporcionados por *Kitchenham*. [19](ver **Tabla 10**).

Origen de datos	Documentación
Base de Datos	Nombre de la base de datos Estrategia de búsqueda para la base de datos. Fecha de búsqueda. Años cubiertos por la búsqueda. Numero de resultados. Numero de duplicados. Variables o *keywords* examinados.
Motor de Búsqueda	Nombre del motor de búsqueda Estrategia de búsqueda. Fecha de búsqueda.
Revista	Nombre de la revista Años cubiertos. Algún tema no cubierto.
Conferencias	Título de los procedimientos. Nombre de la conferencia. Nombre de la revista (si se publicó). Fecha de la conferencia.
Tesis	Título de la tesis. Año de publicación.

	Nombre de la Universidad. URL (si está disponible)
Otras búsquedas.	Fecha de búsqueda/Contacto. URL.

Tabla 10 Procedimiento para la documentación del proceso de búsqueda.

3.2.4 Criterios de inclusión / exclusión para la selección de estudios.

Estos criterios han sido refinados durante la revisión sistemática; para que un estudio cumpla con la revisión sistemática deberá cumplir las siguientes condiciones:

1. Se consideran estudios orientados a aplicaciones web.
2. El estudio debe considerar alguna arquitectura o patrón de arquitectura.
3. El estudio debe considerar cualquiera de estos ítems: *frameworks*, base de datos, lenguajes de programación, IDE.
4. Estudios que presentan revisiones sistemáticas similares.

Se consideran algunos criterios de exclusión que no son relevantes para la investigación; entre ellos tenemos:

Aquellos estudios que no tienen el texto completo o que el acceso a estos es limitado.

Se consideran estudios posteriores al año 2000, no es un limitante; pero tomando en cuenta que el estudio está orientado aplicaciones web, el avance de la tecnología e industrial, etc. se excluyen del estudio.

3.2.5 Listas de verificación y criterios de evaluación de calidad.

La verificación se realiza para poder responder a las preguntas de investigación con más precisión de acuerdo al contexto de esta investigación. Algunas preguntas son sugeridas por *Kitchenham* [19], *Crombie* [22] y *Petti crew* [20] las cuales se han tomado en cuenta para este estudio. Las preguntas se podrán revisar y reformular según avance las tareas de revisión.

Durante la revisión, es probable que se encuentren estudios cualitativos y cuantitativos, por lo tanto, se realizará la verificación para estos dos casos por separado. Así podremos evaluar la calidad de los estudios y hacer preguntas más detalladas. Si no se encuentran estudios cuantitativos, no se debe usar la tabla correspondiente.

Cada una de las preguntas formuladas en las listas de verificación cuantitativas y cualitativas serán respondidas (Si, No, Parcialmente); y cada respuesta tendrá una puntuación de acuerdo con una escala que se muestra a continuación, ver **Tabla 11**.

Respuesta	**Puntaje**
SI	1
NO	0
Parcialmente	0.5

Tabla 11 Escala de valores para las listas de control de calidad.

Listas de revisión para estudios cuantitativos

Ítem	Pregunta	Respuesta SI/NO/Parcial
1	¿Las preguntas de investigación están claramente establecidas para los estudios? [22] [23]	
2	¿Las variables / métricas utilizadas en el estudio se miden y validan adecuadamente? [22] [23] [19]	
3	¿Las métricas utilizadas en el estudio son las más relevantes para responder las preguntas de investigación? [23] [19]	
4	¿Los métodos de recopilación de datos están adecuadamente descritos? [23]	
5	¿Están justificados los métodos estadísticos? [23]	
6	¿Los investigadores discuten algún problema con la validez / confiabilidad de sus resultados? [23]	
7	¿Son creíbles los hallazgos? [19]	
8	¿Existe relación entre los datos, la interpretación y las conclusiones?	

***Tabla 12 Lista de verificación para estudios cuantitativos**.*

Lista de revisión para estudios cualitativo

Ítem	Pregunta	Respuesta SI/NO/Parcial
1	¿El diseño de la investigación es adecuado para llevar a cabo el estudio? [20] [23] [22]	
2	¿Son creíbles los hallazgos? [20]	
3	¿Existe relación entre los datos, la interpretación y las conclusiones?	
4	¿Son claras las suposiciones / perspectivas teóricas / valores que han configurado la forma y el resultado de la evaluación? [23] [20]	
5	¿Las preguntas de investigación están claramente establecidas para los estudios?	

***Tabla 13 Lista de verificacion para estudios cualitativos**.*

3.2.6Proceso de selección.

Proceso de selección preliminar o inicial.

Durante el proceso de selección inicial, se realiza la selección de los títulos y los resúmenes para los estudios primarios potenciales en comparación con los criterios de inclusión y exclusión. Los estudios que cumplan con los criterios mínimos de inclusión se agregarán en la biblioteca de referencia y los que no se descartarán.

Proceso de selección final.

En caso de que no esté claro un estudio que cumple con los criterios de inclusión, se obtendrá su copia y se leerá el documento completo. Si el estudio se considera relevante durante esta etapa, se agregará a la biblioteca de referencia final. Si el estudio está incompleto, se intentará contactar al autor para pedir información, si no hubiera respuesta, el estudio se excluye de la investigación.

Cualquier ambigüedad sobre la inclusión de un estudio durante este proceso se discute entre el investigador y el Asesor y se aclarará mediante discusión.

Además, teniendo en cuenta la evaluación de la calidad, para que un estudio pueda ser agregado a la biblioteca de referencia final, se establece un puntaje mínimo de 4 puntos para los estudios cuantitativos y mínimo 3 para los estudios cualitativos (tablas 12 y 13 respectivamente).

3.3 Estrategia de extracción de datos.

Después de que se hayan seleccionado los estudios primarios y se haya evaluado su calidad, se extraen los datos. Los formularios de extracción de datos y la estrategia que se adoptará para registrar los datos se presentan en los siguientes formularios:

3.3.1Formularios de extracción de datos.

Los formularios de extracción de datos están destinados a contener toda la información necesaria para responder a las preguntas de revisión y abordar los criterios de calidad del estudio [19] . Los formularios de extracción de datos diseñados para los estudios cualitativos **Tabla 14** y estudios cuantitativos, **Tabla 15** son:

Datos	**Valor**	**Notas suplementarias**
ID del Estudio.		
Titulo.		
Autor.		
Año de Publicación.		
Tipo de Referencia.		
Editorial.		
Ciudad de Estudio.		
Dominio de Aplicación		
Entorno de estudio.		
Tipo de Estudio.		
Nro. Citas.		
Datos para extraer y usar para responder preguntas de investigación		
¿Qué definición de generador de código fuente es usada?		
¿Por qué nombre es conocido el generador de código fuente?		
¿Qué patrón de arquitectura/patrón de arquitectura es usado?		
¿Qué interfaz de desarrollo integrado (IDE) se ha considerado?		
¿Qué lenguajes de programación son usados?		
¿Qué plataformas son las más usadas?		
¿Qué versión de lenguaje de marcado o modelamiento es usado?		
¿Qué marco de trabajo utilizan?		
¿Qué tipo de notación utiliza?		

Datos para extraer y usar para responder preguntas de investigación		
¿Qué herramientas utiliza?		
¿Qué librerías utilizan los generadores?		
¿Qué requerimientos se necesitan para generar código Fuente? (input)		
¿Existe algún estudio comparativo de patrones de diseño arquitectónico aplicados a Generadores de código?		
¿El estudio considera funciones básicas Para base de datos? (CRUD u otros).		
¿Existe alguna Clasificación de los generadores de código Fuente?		
Evaluación cualitativa de la calidad del estudio		
¿El diseño de la investigación es adecuado para llevar a cabo el estudio?		
¿Son creíbles los hallazgos?		
¿Existe relación entre los datos, la interpretación y las conclusiones?		
¿Son claras las suposiciones / perspectivas teóricas / valores que han configurado la forma y el resultado de la evaluación?		
¿Las preguntas de investigación están claramente establecidas para los estudios?		

Tabla 14 Formulario de extracción de datos para estudios cualitativos.

Elemento de Datos	**Valor**	**Notas suplementarias**
ID del Estudio.		
Titulo.		
Autor.		
Año de Publicación.		
Tipo de Referencia.		
Editorial.		
Ciudad de Estudio.		
Dominio de Aplicación		
Entorno de estudio.		
Tipo de Estudio.		
Citados		

Datos para extraer y usar para responder preguntas de investigación		
¿El estudio incluye una definición de generador de código fuente? ¿Cual?		
¿Qué variables de comparación se usaron?		
¿Qué medidas / métricas se usaron para medir las variables de comparación?		
¿Qué patrones de arquitectura/diseño fueron comparados y por qué?		
¿El estudio contiene algún comentario o información relevante?		
Evaluación cuantitativa de la calidad del estudio		
¿Las preguntas de investigación están claramente establecidas para los estudios?		
¿Las variables / métricas utilizadas en el estudio se miden y validan adecuadamente?		
¿Las métricas utilizadas en el estudio son las más relevantes para responder las preguntas de investigación?		
¿Están justificados los métodos estadísticos?		
¿Está claro el propósito del análisis de datos?		
¿Son creíbles los hallazgos?		
¿Existe relación entre los datos, la interpretación y las conclusiones?		

Tabla 15 Formulario de extracción de datos para estudios cuantitativos

3.3.2Procedimientos de extracción de datos.

Para los estudios primarios seleccionados, la información será extraída y completada en los formularios de extracción de datos, provistos en las **Tabla 14** y **Tabla 15**, correspondientes al tipo de estudio, ya sea cualitativa o cuantitativa. Para verificar que los datos se han extraído de manera efectiva, se puede probar una muestra aleatoria de estudios primarios y los resultados deben ser igual a los datos extraídos por el investigador. Para todos los estudios restantes, la extracción será llevada a cabo por el investigador. En situaciones donde la información es difícil de entender o si la información parece estar incompleta en el estudio, se contactará al autor (es) del estudio para aclaraciones y / o elaboraciones. Los datos extraídos de los estudios se

llevarán a cabo en documentos de Microsoft Word. Para cada estudio se creará un archivo de Word separado y se guardará con el formato “Autor _ estudio _ Año de publicación”. Después de que los datos de todos los estudios se hayan extraído, revisado y verificado, se consolidarán en un único archivo de Word.

3.4 Síntesis y análisis de los datos extraídos.

Una vez extraídos los datos para cada estudio primario, se consolidarán y resumirán en función de cada pregunta de investigación. La información se tabulará, como se indica a continuación, a fin de que sea más útil para un análisis posterior y para encontrar los vacíos de investigación.

Pregunta 1 (PI1)

Formulación de la Pregunta 1: Planteada en la sección 3.1 está compuesta por cuatro preguntas; de la literatura revisada, los datos necesarios para responder a esta pregunta se resumirán y tabularán en la **Tabla 16**.

ID del estudio	**Nombre del GCF.**	**incluye Patrón de Arquitectura. (Si/No)**	**Nombre del Patrón arquitectura.**	**Notas (si las hay)**

Tabla 16 Resumen de evidencia en generadores de código fuente y patrones de Arquitectura.

Preguntas 2 (PI2)

La **Tabla 17** que se muestra a continuación se utilizará para tabular los resultados / información resumida necesaria para responder a las preguntas de investigación 2. Ayudará a organizar las herramientas usadas para la generación de código fuente basado en la literatura revisada.

ID	Lenguajes	Base de Datos	Herramientas	IDE	Notas (si las hay)

Tabla 17 Características/componentes/ requerimientos existentes.

Pregunta 3 (PI3)

La respuesta a la pregunta nos permitirá saber los marcos que se usan en la generación de código fuente. Los resultados de estas preguntas se tabularán con la ayuda de la **Tabla 18**.

ID del estudio	Nombre del Framework usado	Descripción	Notas (si las hay)

Tabla 18 Resumen de Métodos/Procesos para la generación de Código Fuente.

Pregunta 4 (PI4)

Se extrae la información relevante sobre las aplicaciones en el mundo real y las ventajas de utilizar generadores de código fuente. La **Tabla 19** está diseñada para extraer estos datos.

ID del estudio	Ventajas/ desventajas	Principales Aplicaciones	observaciones

Tabla 19 Resumen de ventajas y aplicaciones de los GCF.

4 Ejecución de la revisión sistemática

4.1 Búsqueda y selección de estudios primarios.

Luego de definir el protocolo de revisión, el siguiente paso fue la ejecución de este protocolo, se obtuvieron las primeras publicaciones el día 25 de mayo de 2018; se hicieron otras búsquedas los días 19 y 25 de Julio de 2018.

Springer:("source code generator" OR "automatic generation of source code" OR "automatic source code generator") AND ((design OR architecture) AND patterns)
Scopus: TITLE-ABS-KEY ((“Source code generat*” OR “automatic generation of source code” OR “automatic source code generat*”) AND ((design OR architect*) AND patterns))
IEEE: ("Source code generat*" OR "automatic generation of source code" OR "automatic source code generat*") AND ((design OR architect*) AND patterns)
ProQuest :("source code generat*" or "automatic generation of source code" or "automatic source code generat*") and ("design patterns")
ACM :("Source code generator" OR "automatic generation of source code" OR "automatic source code generator") AND ((design OR architecture) AND patterns)
Web of science:
(("Source code generat*” OR “automatic generation of source code” OR “automatic source code generat*”) AND ((design OR architect*) AND patterns))
Springer:("source code generator" OR "automatic generation of source code" OR "automatic source code generator") AND ((design OR architecture) AND patterns) AND (tool OR database OR framework)
Scopus: TITLE-ABS-KEY **((“Source code generat*” OR “automatic generation of source code” OR “automatic source** code generat*”) AND ((design OR architect*) AND patterns) AND (tool OR database OR framework))

Con la finalidad de conseguir la mayor cantidad de estudios relevantes, se volvió a realizar una búsqueda entre el 15 y 22 de marzo de 2019 aplicando la cadena de búsqueda definida en el protocolo de revisión a las bases de datos electrónicas también definidas en el protocolo.

("source code generat*" OR "automatic generation of source code" OR "automatic source code generat*" OR scg OR "automatic programming") AND (architect* OR

"architect* pattern") AND ("web application" OR web) AND (framework OR database OR IDE OR language OR tool)

En la fase de selección inicial se obtuvieron un total de 367 estudios, para ejecutar el protocolo de revisión se utilizaron herramientas para gestionar referencias bibliográficas como Mendeley[1], Zotero[2] y StArt[3] (State of the Art through Systematic Reviews); este último en particular ha sido probada de manera empírica y utilizada por varios investigadores; obteniendo resultados que han demostrado la efectividad de la herramienta en el desarrollo de revisión sistemática de la literatura.

El resultado de la búsqueda ha sido integrado a la herramienta StArt, en seguida se inició el proceso de selección; en primer lugar, se eliminaron los estudios duplicados y rechazados por no cumplir con criterios como: No tratarse de temas de desarrollo de software o ciencias de la computación, no presentar algún patrón o arquitectura de software o no especificar la utilización de GCF. Luego de la selección inicial, 166 publicaciones son rechazadas, 73 estudios estuvieron duplicados quedando 128 estudios para pasarlos a la etapa de selección final.

[1] Mendeley: https://www.mendeley.com
[2] Zotero: https://www.zotero.org/
[3] Start: http://lapes.dc.ufscar.br/tools/start_tool

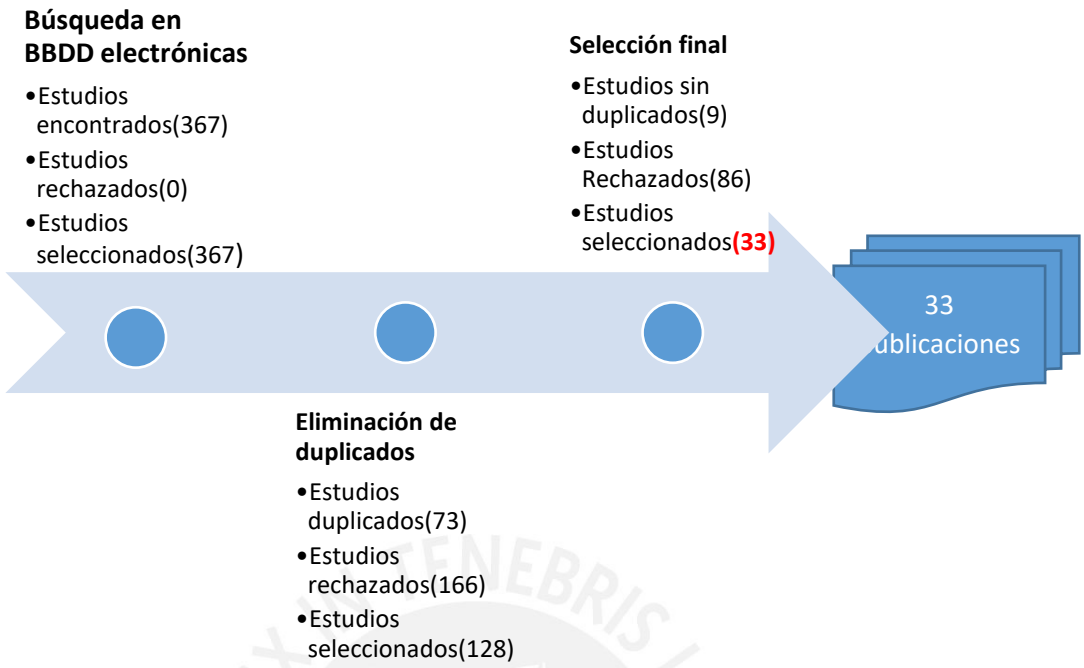

Ilustración 2 Fases del Proceso de Selección

Bases de datos electrónicas	Estudios iniciales encontrados	Estudios duplicados	Estudios Rechazados	Estudios Seleccionados
Scopus	43	0	27	16
IEEE	26	4	7	15
ProQuest	98	25	59	14
Web of science	0	0	0	0
ACM	85	20	34	31
Springer	49	6	30	13
Mendeley	51	18	4	29
Google Scholar	15	0	5	10
	367	**73**	**166**	**128**

Tabla 20 Selección inicial de estudios.

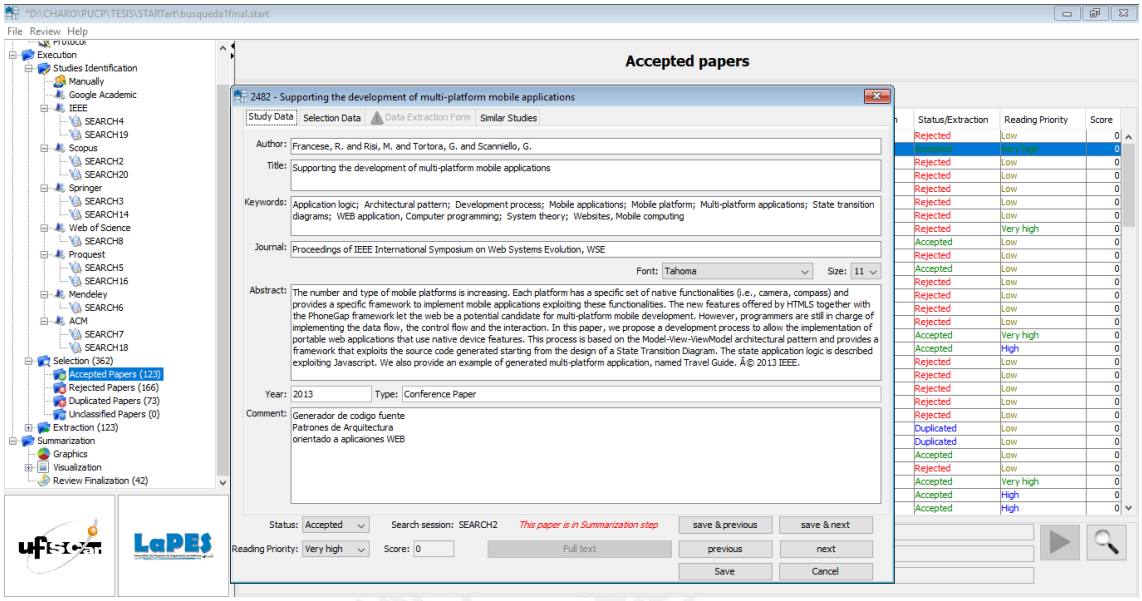

Ilustración 3 Formulario de proceso de selección inicial de estudios de la Herramienta StART

La Ilustración 3 muestra una visión general de la selección inicial de estudios por cada base de datos.

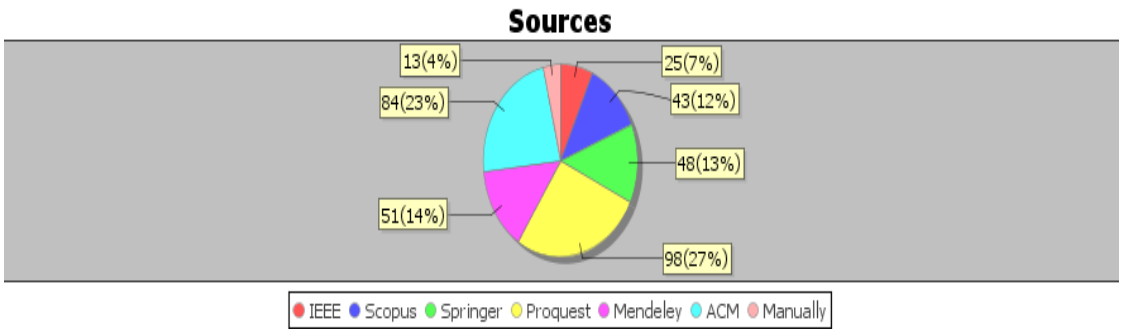

Ilustración 4 Visión general de la selección inicial por BBDD

Luego se procede a verificar el texto de cada estudio para seguir encontrando duplicados y temas similares, ya que algunos estudios tienen el mismo contenido sin embargo el título es diferente; además se rechazó aquellos estudios que no tienen el texto completo o que el acceso a estos es limitado.

Luego de revisar los estudios más relevantes se aplicó los criterios de selección (estudios que estén orientados a Web, Generador de código fuente, tiene algún patrón de arquitectura, a partir del año 2000, documento

incompleto, puntaje de calidad menor a 3 puntos, contenido similar a otro estudio).

La selección final se realizó el 22 de marzo del 2019 el cual incluye la revisión de las referencias de algunos estudios que han sido incluidos a la investigación por tener relación con el tema, como es el caso de los estudios (S27) y (S5); también se obtuvo el texto completo de los artículos, descartándose aquellos que no lo tenían; como es el caso del estudio [5] a pesar de intentar contactar con los autores sin conseguir respuesta. En total se tiene 33 estudios seleccionados para la extracción y síntesis de la información. El resumen de la selección final se muestra en la siguiente tabla: se aceptan 33 estudios, 82 estudios duplicados y 252 estudios rechazados en total.

Bases de datos	**Estudios después de la selección Inicial.**	**Estudios duplicados**	**Estudios Rechazados**	**Estudios Seleccionados**
Scopus	16	0	14	2
IEEE	15	2	6	7
Proquest	14	0	8	6
Web of science	0	0	0	0
ACM	31	1	27	3
Springer	13	0	10	3
Mendeley	29	5	18	6
Google scholar	10	1	3	6
	128	**9**	**86**	**33**

Tabla 21 Resultados de la selección inicial de estudios.

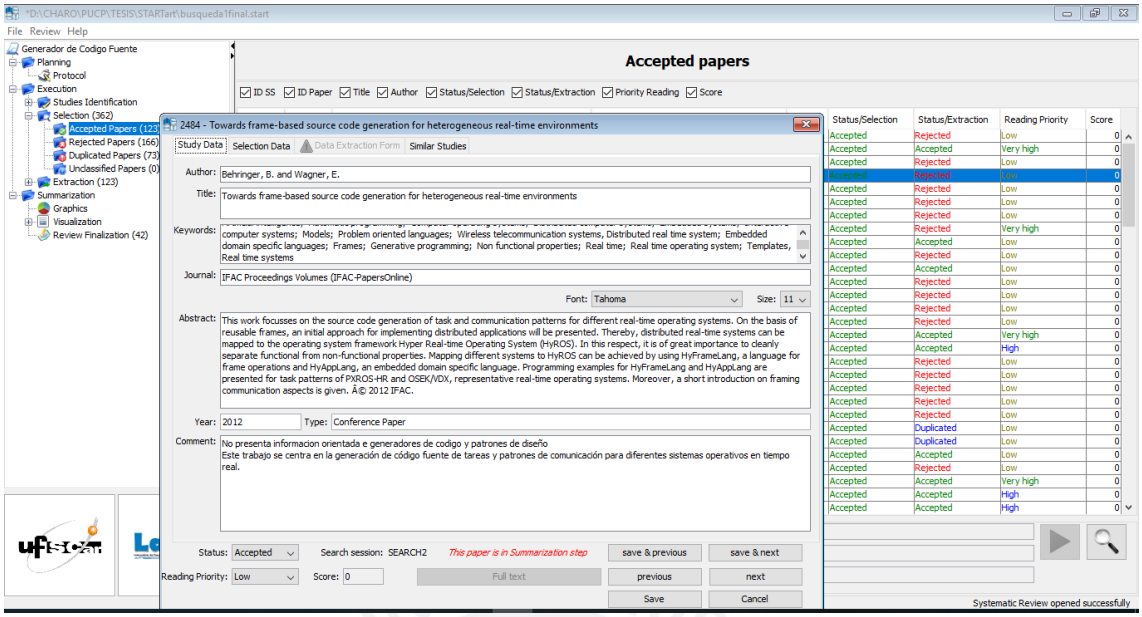

Ilustración 5 Proceso de selección de estudios primarios.

Bases de datos	Total, de estudios seleccionados.	Total, de estudios duplicados	Total, de Estudios Rechazados	Estudios Seleccionados
Scopus	43	0	41	2
IEEE	26	6	13	7
Proquest	98	25	67	6
Web of science	0	0	0	0
ACM	85	22	61	3
Springer	49	6	40	3
Mendeley	51	23	22	6
Google Scholar	15	1	8	6
Total	**367**	**82**	**252**	**33**

Tabla 22 Resultado de selección total de estudios por Base de datos.

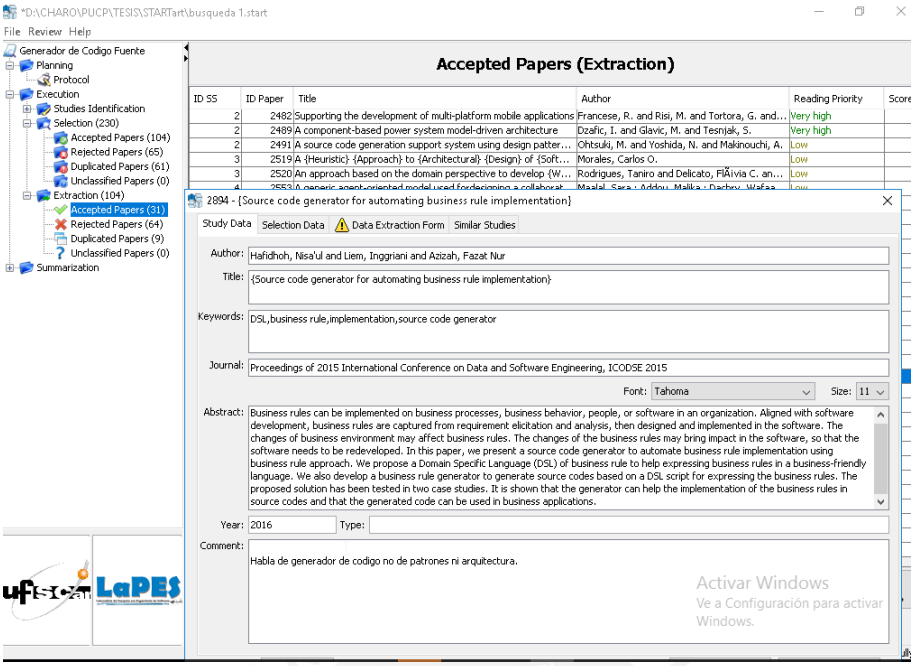

Ilustración 6 Resultado de la Selección con la herramienta StArt

4.2 Evaluación de la calidad de los estudios.

El puntaje se aplicó tomando en cuenta la escala de valores de la **Tabla 11** y los criterios de evaluación de la **Tabla 13** definidos en el protocolo de revisión. El puntaje disgregado se muestra en el **anexo 05.**

4.3 Extracción y Síntesis.

Se hizo la lectura y análisis de los 33 estudios seleccionados para extraer la información más importante relacionada con las preguntas de investigación. A continuación, se muestra el detalle de los estudios relevantes mostrando puntaje de cada estudio, así como el país de estudio según la tabla 13 definida en el protocolo de revisión.

La **Tabla 23** presenta el año de publicación, el autor, el puntaje de calidad y el país de estudio.

ID	Año	Autor	Puntaje de calidad	País estudio
S1	2013	Frances et al. [24]	5	Italia
S2	2004	Zhang et al. [25]	4	USA
S3	2000	Heescha et al. [26]	3.5	Holanda
S4	2014	Morales, Carlos O. [17]	3	USA
S5	2017	Rodrigues et al. [27]	4.5	Alemania
S6	2012	Romano et al. [28]	4	USA
S7	2009	Shih et al. [29]	3.5	USA
S8	2007	Rodrigues da Silva et al. [30]	4	Portugal
S9	2016	Carromeu et al. [18]	4.5	Brasil
S10	2015	Rosales-Morales et al. [10]	5	México
S11	2015	Mizuta et al. [31]	3.5	Taiwán
S12	2014	Alonso et al. [32]	3.5	España
S13	2012	Song et al. [15]	5	China
S14	2015	Freitas et al. [33]	4	Brasil
S15	2016	Deepika et al. [34]	4	India
S16	2002	Codagen [35]	5	Canadá
S17	2000	Grenier, Robert Herbert. [36]	4	USA
S18	2011	Radosevic et al. [37]	3.5	Croacia
S19	2016	Hafidhoh et al. [38]	5	Indonesia
S20	2001	Dong Hyuk Park et al. [39]	4	Corea
S21	2011	Magdalenić et al. [9]	4.5	Croacia
S22	2018	Deeba et al. [40]	5	China

ID	Año	Autor	Puntaje de calidad	País estudio
S23	2010	Dejanović et al. [6]	4	Serbia
S24	2016	El-khoury, Jad [41]	5	Suecia
S25	2004	Fowler et al. [42]	5	Inglaterra (UK)
S26	2011	Zapata et al. [43]	4.5	Colombia
S27	2002	Volker Tarau [11]	4	Alemania
S28	2013	Havva et al. [12]	5	Turquía
S29	2011	Stevica et al. [44]	4	Serbia
S30	2014	Deivis de Jesús Martínez Acosta [7]	3.5	Colombia
S31	2016	Eduardo Chávez et al. [4]	4	Ecuador
S32	2015	Atkinson et al. [45]	4	Alemania, Austria
S33	2002	Suganuma et al. [46]	4	Japón

Tabla 23- Extracción y síntesis de estudios

4.3.1 Años de Publicación

En la **Ilustración 7** se muestra el número de publicaciones por año, considerado a partir del año 2000 según la definición del protocolo; también se puede observar que en los años 2011 y 2016 se tiene el mayor número de estudios relevantes, mientras que para el año en que se ejecutó la búsqueda (2019) no se encontró estudios relacionados al tema de investigación o fueron descartados por no tener el contenido completo.

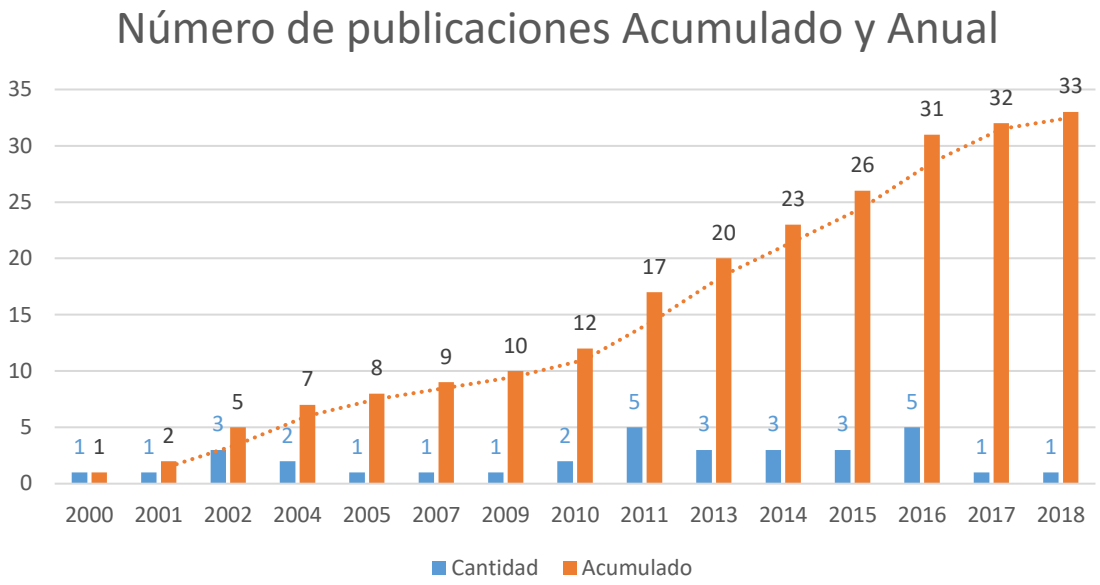

Ilustración 7- Nro. de publicaciones acumulado (Fuente Propia)

4.3.2 Fuentes de Publicación.

La **Tabla 24** muestra las fuentes de publicación y las citas.

Estudio	Citas (Google Scholar)	Tipo	Fuente
S1	17	Simposio	International Symposium on Web Systems Evolution (WSE)
S2	21	Simposio	ACM symposium on Applied computing
S3	10	Revista	Science of Computer Programming
S4	00	Book -chapter	Advances in Product Family and Product Platform Design
S5	08	Revista	Software & Systems Modeling
S6	06	Conferencia	International Conference on Information Technology: New Generations
S7	21	Conferencia	International Computer Software and Applications Conference

Estudio	Citas (Google Scholar)	Tipo	Fuente
S8	41	Conferencia	Workshop on Model-Based Methodologies for Pervasive and Embedded Software (MOMPES'07)
S9	02	Conferencia	International Conference on Computational Science and Its Applications
S10	05	Revista	Revista Facultad de Ingeniería Universidad de Antioquia
S11	10	Conferencia	International Conference on Advanced Information Networking and Applications (AINA'05)
S12	05	Revista	The Scientific World Journal
S13	02	Revista	Applied Mechanics and Materials
S14	02	Revista	Symposium on Components, Architectures and Reuse Software
S15	04	Revista	Conference on Advances in Computing, Communications and Informatics (ICACCI)
S16	02	Revista	*Codagen Technologies Corp*
S17	00	Tesis	"School of Computer and Information Sciences Nova Southeastern University"
S18	10	Conferencia	International Convention MIPRO
S19	03	Conferencia	International Conference on Data and Software Engineering (ICoDSE)
S20	23	Conferencia	Asia-Pacific Software Engineering Conference
S21	01	Revista	Journal Communications of Software and Systems
S22	00	Conferencia	International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)
S23	41	Revista	Computer Science and Information Systems (ComSIS)
S24	02	Revista	SoftwareX Science Direct elsevier
S25	03	Conferencia	OOPSLA (Object-Oriented Programming, Systems, Languages & Applications) Conference

Estudio	Citas (Google Scholar)	Tipo	Fuente
S26	00	Conferencia	Latin American and Caribbean Conference for Engineering and Technology (LACCEI'2011)
S27	55	Simposio	Symposium on Applied computing
S28	02	Conferencia	The International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE)
S29	02	Revista	Electronics
S30	00	Tesis	Universidad Nacional de Colombia Facultad de Ingeniería, Departamento de Ingeniería de Sistemas e Industrial
S31	00	Revista	" GEEKS"-DECC-Reports
S32	06	Revista	In Journal of Object Technology
S33	00	Conferencia	Annual International Computer Software and Applications

Tabla 24 Fuentes de Publicación y citas.

Ilustración 8 Estudios por tipo de publicación

4.3.3Numero de Citas de los Estudios.

La cita se tomó de las publicadas en Google Scholar, como los datos varían de acuerdo a la fecha es necesario señalar que los datos se obtuvieron el día 22/Marzo/2019, con el fin de analizar la importancia de los estudios en el ambiente científico. El rango de citas varia de 0 a 55. Teniendo 10 estudios con mayores citas.

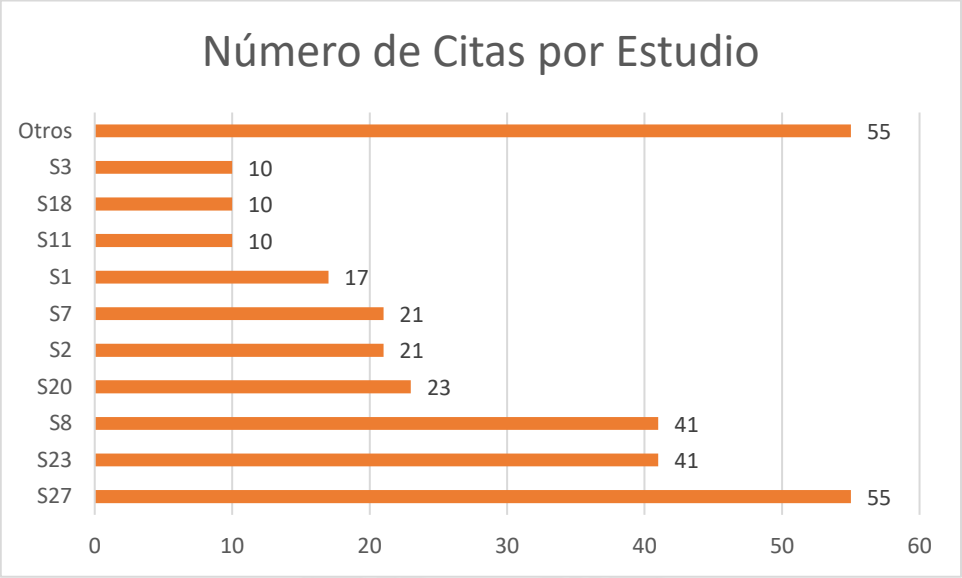

Ilustración 9 Nro. de citas por estudio (Fuente Propia).

4.3.4Centros de Investigación.

Se revisó a los autores de cada estudio para obtener los centros de investigación a los cuales están afiliados. De los 33 estudios se obtuvo 3 estudios de afiliación comercial (S9, S16, S25), los demás son instituciones académicas.

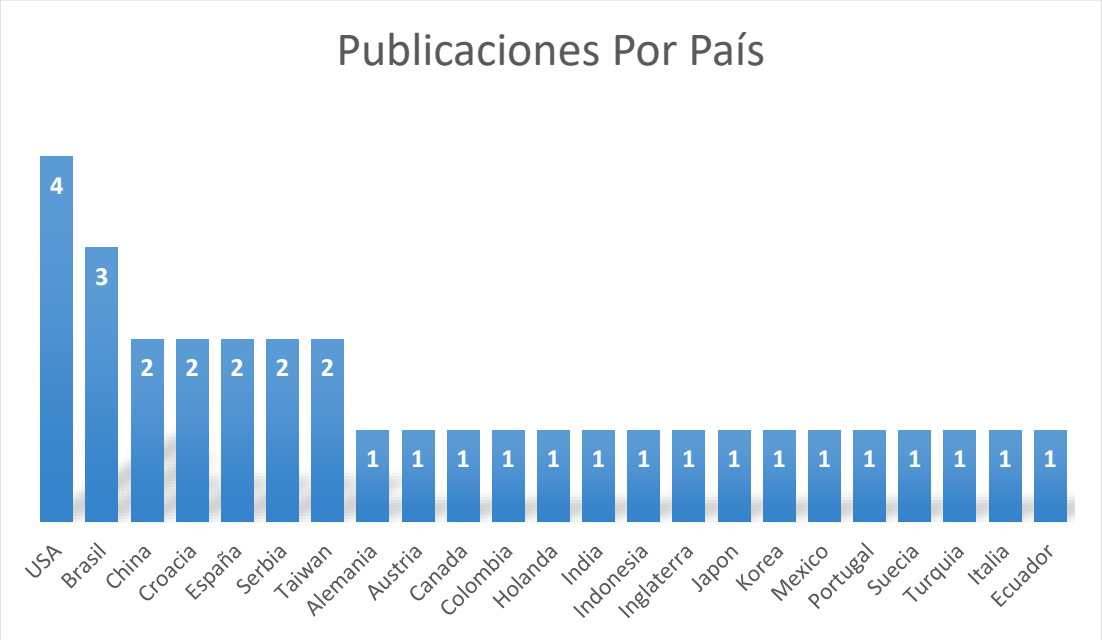

Ilustración 10- Publicaciones por país

En la **Ilustración 10** se muestra la distribución de estudios por el país de afiliación. De un total de 23 países, donde USA y Brasil son los países con más publicaciones (4 y 3 estudios respectivamente), seguido de China, Croacia, España, Serbia y Taiwán (2) los demás tienen 1 estudio por país.

Estudio	Institución de investigación
S1	**Dipartimento di Management & Information Technology University of Salerno, Italy**
S2	**Department of Computer Science Iowa State University. USA**
S3	**University of Groningen, Holanda**
S4	**Northeastern University, USA**
S**5**	**Universidade Federal do Rio de Janeiro Rio de Janeiro Brasil**
S6	**Federal Institute of Education, Science and Technology of São Paulo Universidade Federal de Viçosa (UFV)**
S7	**Tunghai University, Taiwan**
S8	**INESC-ID, Instituto Superior Técnico Brasil.**
S9	**Federal University of Mato Grosso, Brazil**
S10	**Universidad de Antioquia, México**
S11	Hosei University, Tokyo
S12	Universidad Politécnica de Cartagena, España
S13	Beijing University, China

Estudio	Institución de investigación
S14	**Universidade Estadual do Ceará, Brasil**
S15	Centre for Development of Advanced Computing (C-DAC), India
S17	Nova Southeastern University
S18, S21	University of Zagreb, Croacia
S19	**School of Electrical Engineering and Informatics, Indonesia**
S20	**Soongsil University, Seoul, South Korea**
S22	Quaid-e-Awam University of Electronic Science and Technology, China
S23	**University of Novi Sad, Serbia**
S24	KTH Royal Institute of Technology, Stockholm, Suecia
S26	Universidad Nacional de Colombia, Medellín, Colombia
S27	University of Applied Sciences, Alemania
S28	Mevlana University, Turquia
S29	University of Niš Serbia
S30	Universidad Nacional de Colombia
S31	Universidad de las Fuerzas Armadas “ESPE”, Ecuador
S31	University of Mannheim, Alemania University of Innsbruck, Austria
S33	**Hitachi Software Eng. Co., Ltd., Yokohama, Japan**

Tabla 25 Estudios por centros de investigación.

5 Resultados de la Revisión Sistemática de la Literatura

5.1 Generadores de Código Fuente y Patrones de Arquitectura (PI1).

Esta primera parte comprende 4 preguntas que están orientadas a analizar los GCF para comprender su importancia en el desarrollo de sistemas informáticos, analizar el uso de patrones de arquitecturas; también se menciona el nombre de algunas arquitecturas aplicadas y algunos GCF conocidos según los estudios.

5.1.1 Importancia de los Generadores de Código fuente (PI1a).

¿Cuál es la importancia de los GCF en la construcción de Sistemas de Software?

Resultados:

El objetivo de esta pregunta es resaltar el papel que cumplen los GCF en el desarrollo de software, justificando la importancia de su uso.

Según [5], para construir sistemas de software se requiere de un alto nivel de abstracción, así como encontrar formas de reutilizar el software existente para aumentar el rendimiento y productividad; los GCF son herramientas que se utilizan en una o más fases del ciclo de desarrollo de software pueden ser utilizadas para lograr software de buena calidad.

El estudio (S28) [36] describe al GCF como una herramienta esencial que proporciona una codificación más rápida y estandarizada para sistemas de software a gran escala que es necesario para las organizaciones institucionales; por lo general, en estas aplicaciones informáticas a gran escala, la generación de fragmentos de código debe producirse una y otra vez, por lo tanto, lleva mucho tiempo. Por esta razón, y debido a la gran cantidad de módulos de estos

sistemas, la herramienta de generación de código es una necesidad fundamental.

En (S13)[15] se considera a la generación automática de código fuente como una parte esencial de los enfoques basados en modelos para el desarrollo de software. Se cierra la brecha que surge cuando los modelos se utilizan para abstraer un sistema de software concreto. El desarrollo simple y rápido de generadores de código es un requisito clave de los enfoques de hoy en día para el desarrollo impulsado por modelos, que se caracterizan cada vez más por un fuerte enfoque en la agilidad y la especificidad de dominio.

La herramienta de generación de código acorta enormemente el ciclo de desarrollo de los sistemas reduciendo el costo del desarrollo de software, garantizando la buena funcionalidad y escalabilidad [15](S13).

Otra ventaja tiene que ver con la eliminación de errores de codificación manual, y evitar trabajos repetitivos, así como proporcionar una vía rápida a sistemas o aplicaciones implementables y comprobables (S29)

Para responder esta pregunta se ha identificado algunas características y beneficios que pueden brindarnos los GCF:

Características	**Estudios**	**Conteo**	**Porcentaje**
Velocidad	S2, S13, S14, S21, S28, S1, S27	07	21.2%
Rendimiento	S7, S20, S2	03	9.1%
Escalabilidad	S1, S6, S13, S11, S9, S2	06	18.08%
Re utilidad	S1, S2, S6, S4, S11, S12, S5, S21, S25, S12, S3, S8, S9, S20, S29, S30, S31, S33	18	54.5%

Características	Estudios	Conteo	Porcentaje
Portabilidad	S1, S9, S12, S18, S29, S27	06	18.18%
Estandarización	S24, S12, S22, S28, S2, S32	06	27,3%
Reducción de Costo y/o tiempo	S4, S8, S9, S13, S14, S15, S16, S17, S18, S20, S22, S24, S26, S31, S25, S31, S33	17	51.5%
Aplicaciones grandes y complejas	S6, S2, S13, S14, S3, S19, S28, S29, S23, S27, S3	11	33.3%
Productividad	S2, S4, S5, S7, S11, S15, S25, S33, S30, S33	10	30.30%
Mantenibilidad	S2, S4, S5, S15, S16, S33, S30, S9	08	24.24%

Tabla 26 Características de los GCF encontrados.

De la investigación realizada se ha encontrado que los GCF pueden aplicarse para el desarrollo de aplicaciones grandes y complejas, con un 33.3%; pueden reutilizarse en varios proyectos de desarrollo de aplicaciones, con un total de 54.5%; permiten la reducción de costo y/o tiempo, con un total de 51.5%; otras características interesantes que se ha encontrado es de que aumenta la velocidad, portabilidad, productividad, rendimiento, y escalabilidad (orientado principalmente al desarrollo de diferentes patrones de arquitectura) en el desarrollo de aplicaciones empresariales.

También existen otras características de los GCF como la funcionabilidad, fiabilidad y usabilidad.

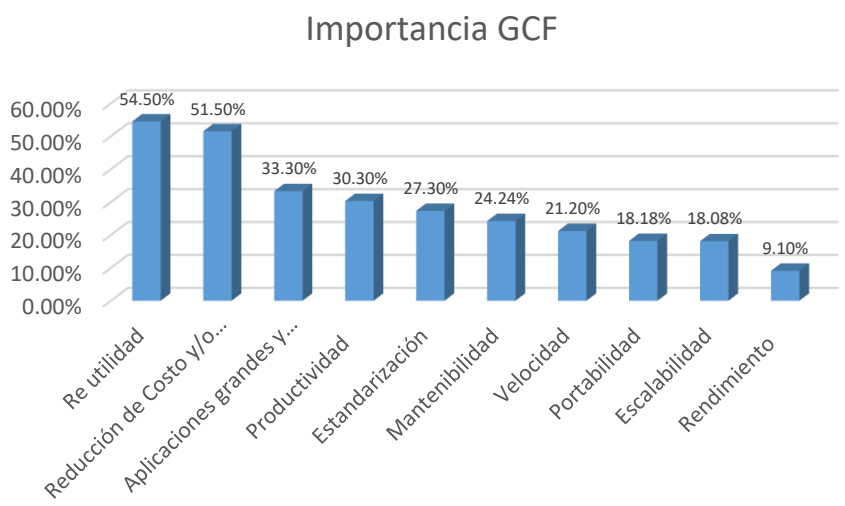

Ilustración 11- Importancia de los GCF

Analisis y discusión.

Según los estudios la importacia del uso de la herramienta de generacion de codigo fuente se basa principalmente en la reutilizacion de codigo, con lo que se reduce el ciclo de desarrollo de software; por lo tanto, el costo y el tiempo de desarrollo de nuevas aplicaciones; ademas, se mejora la calidad del producto debido a las pruebas exhaustivas que se hace al generador de código (S28) [12].

La calidad del software se aprecia en el resultado de la generación del código fuente, obteniendose codigo estandarizado, coherente y homogeneo; sin la intervención humana; lo que contribuye a reducir el tiempo de desarrollo logrando aumentar en la productividad; se puede generar hasta el 90% de las líneas de código(S16)(S30), esto implica ahorro de esfuerzo para el programador (S30) [7], en general, los GCF permiten reducir errores de programación, estandariza y equilibra la habilidad del equipo de desarrolladores .

Por otra parte, los generadores de código, reflejan cualquier cambio realizado en el modelo a todos los archivos generados, lo que hace que la implementación o mantenimiento sea consistente con el modelo, eliminando errores debido a factores humanos, que, inevitablemente ocurren si estos cambios se realizan manualmente (S23) [6]; ademas, los GCF permiten automatizar operaciones a

nivel de transacciones de bases de datos CRUD (Create, read, update, delete)(S6),(S9),(S13),(S14).

La portabilidad es otra caracteristica de la generacion de código, especificamente relacionado a las aplicaciones web; que permite a los usuarios utilizar funciones de otros dispositivos o servicios (S1) [24].

Así como existen ventajas para hacer uso de los GCF en el desarrollo de sistemas informáticos, también existen algunas desventajas que deben tomarse en cuenta:, como la generacion de dependencia, multilenguaje, actualización continua de software, etc.

Debido a la optimización, depuración o generación de nuevas versiones de librerias; tambien es necesario que los GCF se actualicen constantemente, de lo contrario podria quedar obsoleta o podría generar aplicaciones lentas o que no se adapten a las nuevas versiones de los sistemas operativos, browsers, librerias o dispositivos electrónicos [38].

Si se depende de un GCF para el desarrollo de aplicaciones, es posible que ante una modificacion o requerimiento del usuario, se tenga limitaciones para atenderlo, debido a que no se tiene control del GCF, para reducir esta dependencia es conveniente que la empresa desarrolladora genere su propio GCF o tenga las fuentes o la posibilidad de modificar las plantillas para hacer modificaciones a su GCF [7].

Hacer uso de un GCF multilenguaje, podria generar código fuente para diferentes lenguages, pero al estar pensado para que funcione para muchos proyectos y de diferentes areas de conocimiento, hará uso de librerias genéricas que no necesariamente sea compatible con otras herramientas, por lo tanto el resultado no sería óptimo [4]

Los GCF que usan datos estructurados requieren previamente que se construyan los modelos o los diseños de clase en otra herramienta, por lo que

es necesario que el desarrollador previamente conozca acerca de los lenguajes formales como UML [42].

5.1.2 Importancia de los patrones de arquitectura en los GCF (PI1b).

¿Por qué es importante utilizar patrones de Arquitectura para generar código fuente?

El objetivo de esta pregunta es encontrar estudios que demuestren la importancia de aplicar patrones de arquitectura en el desarrollo de software.

(S4) [17] El rol central de la arquitectura de software es resaltar su relación con el análisis de nuevos dominios, de esta manera proporcionar una abstracción de todo el sistema.

Según [12](S28), un GCF proporciona una codificación más rápida y estandarizada para sistemas a gran escala como: sistemas bancarios, sistemas para seguro social, sistema de hospitales, etc. que se caracterizan por la gran cantidad de módulos que manejan; estos sistemas, por lo general siguen un patrón de arquitectura multicapa.

Durante la fase de arquitectura de un proyecto de software, se toman muchas decisiones que influyen en la estructura fundamental y el comportamiento del sistema de software a desarrollar. El uso de patrones de arquitectura, aumentan significativamente la calidad de las aplicaciones de software S3.

Resultados

Para responder esta pregunta se ha identificado algunas características y beneficios que pueden brindarnos los patrones de arquitectura en el desarrollo de aplicaciones informáticas:

Característica	Estudios	Conteo	Porcentaje
Alto nivel de abstracción.	S5, S7, S14, S12, S3, S22, S8, S22, S23, S27, S26	09	27.3%
Reducción de Costo y/o tiempo	S4, S8, S9, S14, S15, S16, S17, S18, S22, S24, S26, S31, S25, S31, S33	15	45.5%
Aplicaciones a gran escala.	S2, S6, S14, S3, S19, S28, S29, S23, S27, S3	09	27.3%
Calidad	S3, S7, S15, S16, S30	04	12.1%
Funcionalidad, fiabilidad y usabilidad	S10, S17, S16, S9, S24, S20, S2, S32, s27, S31	10	3.3%
Mantenibilidad	S2, S5, S15, S16, S33, S30, S9	06	18.2%
Configuración	S21, S24, S18, S19, S20, S2	06	18.2%
Estandarización	S28	01	3%

Tabla 27 Características de los Patrones de Arquitectura.

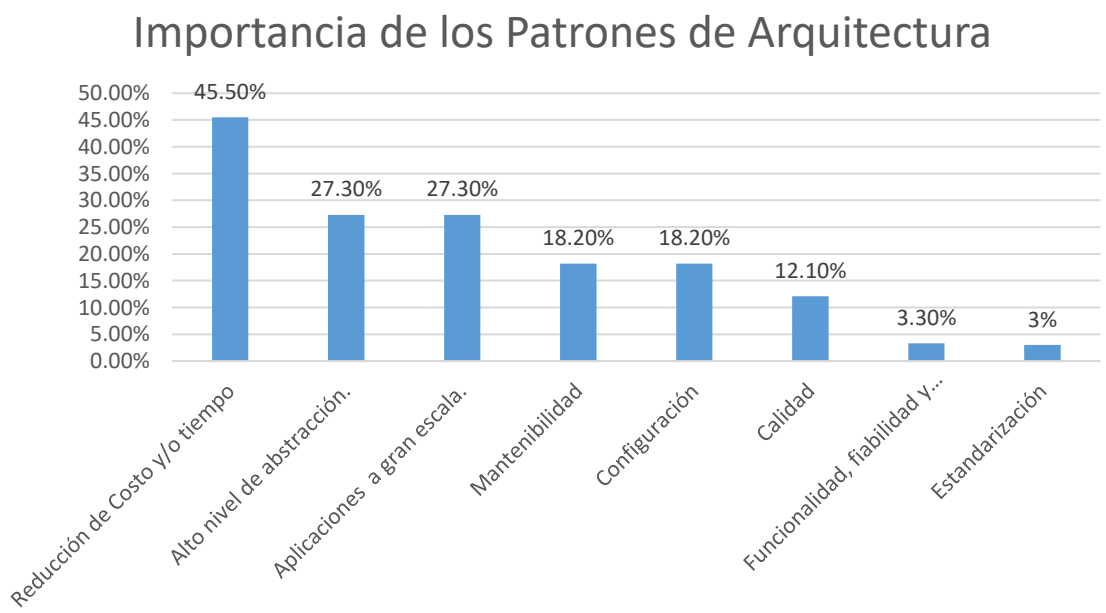

Ilustración 12- Importancia de los patrones de arquitectura

De los estudios revisados, se ha encontrado que los patrones de arquitectura se aplican en sistemas de software grandes y complejas, en un 27.3%; permiten un alto nivel de abstracción, lo cual implica un alto nivel de análisis, diseño y planificación antes de empezar a codificar los programas, con un 27.3%; al identificar y planificar detalladamente los requerimientos, recursos y costos, permitirá reducir los costos y el tiempo de entrega de los proyectos, con un 45.5%; otras características importantes que se han encontrado es que los patrones de arquitectura mejoran la calidad, funcionalidad, usabilidad, mantenimiento y configuración de los sistemas que se van a desarrollar. La estandarización del código es otro factor influyente para el uso de patrones de arquitectura.

Análisis y discusión

Los patrones arquitectónicos de software, como todos los patrones, capturan soluciones genéricas para problemas recurrentes en contextos específicos, poseen conocimiento que se puede reutilizar en un dominio de aplicación particular y se basan en alternativas de diseño [26].

Tomando en cuenta el nivel de abstracción que se necesita para el desarrollo de software, además de la reusabilidad y la estandarización del código, los patrones de arquitectura ofrecen soluciones a problemas de arquitectura de software, están orientados a la parte de la composición u organización de un sistema informático, es decir tiene que ver con la organización, partes, componentes, capas o estructura interna de un sistema informático [2] .

Establecer patrones de arquitectura para un sistema informático, nos permitirá crear sistemas modulares, lo cual tiene su ventaja, porque cuando se realizan actualizaciones o cambios no es necesario cambiar y/o compilar todo el sistema, si no bastará con modificar o actualizar solo aquel paquete o capa que requiera modificarse [3]. Es conveniente que antes de desarrollar un sistema, se defina el patrón de arquitectura, para evitar la continua modificación en la codificación,

configuración, utilización de librerías, recursos humanos, etc.; es decir, dependiendo del patrón elegido, el alcance, tiempo y costo del proyecto puede cambiar [12].

Otra razón importante para hacer uso de un patrón de arquitectura está relacionado al hecho de hacer uso de buenas prácticas al momento de desarrollar, estandarizar la codificación de los programas, generar aplicaciones rápidas y que consuman pocos recursos de hardware [3]. Así mismo, en [7] se ha encontrado que existe una relación entre los patrones de arquitectura y algunas características y atributos comunes de calidad del aplicativo, como son la funcionabilidad, usabilidad, mantenibilidad, configuración, fiabilidad, etc.

5.1.3Patrones de Arquitectura (PI1c)

¿Qué arquitectura o patrón de arquitectura utilizan los GCF?

El objetivo de esta pregunta es identificar los patrones de arquitectura que se están aplicando para la generación de código fuente.

Patrones de Arquitectura	**Descripción**	**Estudio**	**Conteo**	**%**
MVVM	Modelo Vista Vista-Modelo.	S1	01	3%
MVC	Modelo Vista Controlador.	S2, S11, S13, S15, S26, S27, S30, S19	08	24.2%
Multicapa	Arquitectura basada en capas o multicapas.	S4, S11, S13, S16 S19, S25, S28, S29, S31, S33	10	30.3%

Patrones de Arquitectura	Descripción	Estudio	Conteo	%
REST	Transferencia de estado representacional, adopta la arquitectura de internet y tecnologías web en general. No depende de la plataforma. Acepta JSON	S9, S24, S32	03	9.1%
Naked Object	Basada en el patrón multicapas. (Presentación, control, dominio y persistencia)	S14	01	3%
***MDA**	Arquitectura Dirigida por modelos.	S2, S4, S5, S6, S7, S8 S11, S14, S16, S22, S23, S25, S32	13	39.4%
***SOA**	SOA (Arquitectura orientada a servicios) es un enfoque para estructurar sistemas en torno a servicios.	S6, S12, S32	03	9.1%
CBSE	Arquitectura basada en componentes	S8, S12	02	6%
oAW	Open Architecture Ware	S23	01	3%

Tabla 28 Principales Patrones de Arquitectura de los GCF

Resultados:

Del 100% de los estudios se encontró que el 84.85 % utiliza alguna arquitectura o patrón arquitectónico para generar código. El 15.15 % no especifica patrón de arquitectura porque la generación de código se realiza a través de reglas, plantillas o solo se aplica a una capa en particular.

En la **Tabla 28** se presenta una lista con las arquitecturas que se usan para la GCF entre ellos están los patrones de arquitectura Modelo Vista Controlador, que representan el 24.2% (08 estudios) que son S2, S11, S13, S15, S26, S27,

S30, S19; también está el patrón MVVM, estudio S1, que representa el 3% del total de estudios S1; el patrón multicapas que representa el 30.3% de los estudios (10 estudios) que son S4, S11, S13, S16 S19, S25, S28, S29, S31, S33.

El patrón Arquitectónico REST representando el 9.1 % del total de estudios. Se encontró estudios que incluyen Arquitectura Orientada a Servicios (SOA) que representa también el 9.1% (03 estudios); además el estudio (S14) presenta un patrón de arquitectura *Naked Object* (objetos desnudos), que está basada en el patrón multicapa.

El 39.4% del total de estudios utilizan MDA (Arquitectura dirigida por modelos) que representan 13 estudios, estos son: S2, S4, S5, S6, S7, S8 S11, S14, S16, S22, S23, S25, S32; el destalle de estas arquitecturas está en el Anexo 04.

Análisis y Discusión:

La arquitectura de software tiene que ver con todo el proceso del modelado o diseño de la aplicación desde la etapa de los requerimientos hasta la etapa de configuración y puesta en operación (normalmente comprende todo el ciclo de vida del software) y tiene un nivel de abstracción muy alto; en cambio, los patrones de arquitectura están referido a la parte de cómo está estructurado el sistema o que componentes forman parte del software, está orientado más a la parte de ejecución o de desarrollo [16].

Los estudios se orientan al uso de la Arquitectura Dirigida por Modelos (MDA) propuesta por la OMG, en (S4) [17] se menciona que la abstracción es la característica más sólida del desarrollo basado en modelos, este enfoque utiliza diferentes patrones y *frameworks* para el desarrollo de código.

Respecto a los patrones de arquitectura más emergentes para el desarrollo de aplicaciones web se consideran MVC y MVVM, siendo el segundo el que presenta mayores ventajas con respecto al primero[1]; entre sus principales

[1] Comparación entre los patrones MVC y MVVM (https://stackoverrun.com/es/q/309975)

ventajas podemos mencionar algunas de ellas: El MVVM permite una mejor vinculación entre la interfaz del usuario y las estructuras de datos, la validación de datos y reglas de negocio se realizan en el lado del cliente y la carga de procesos al servidor se disminuye, haciendo que las aplicaciones usando el modelo MVVM sean más rápidas [24]. Además, este patrón está relacionado con el patrón REST cuya característica es que permite el procesamiento de imágenes y video.

5.1.4 Generadores de Código Fuente (PI1d)

¿Qué GCF se han desarrollado?

El objetivo de esta pregunta es identificar los principales GCF encontrados que evidencien el uso de patrones de arquitectura.

Resultados:

El 39.4% del total de estudios revisados, se caracterizan por tener el nombre de la aplicación que genera código, se tomó estos estudios para mostrar la relación entre GCF y el patrón que utilizan: S1, S2, S9, S10, S14, S16, S24, S23, S24, S25 S29, S30, S31.

En el 64% de los estudios, los GCF no especifican un nombre; describen el proceso de generación de código, patrón de arquitectura, así como los *frames* y herramientas que se utilizan para la generación de código; los cuales se especifican más adelante.

Nombre del Generador	**Descripción**
Travel Guide Application (S1)	Genera código para la implementación de aplicaciones web portátiles utiliza patrón de Arquitectura MVVM a partir del diseño de un diagrama de transición de estado.
***WebGen* (S2)**	Genera código de forma automática desde el *front-end* hasta el *back-end*.
***Titan Architect Mobile (*S9)**	Genera código para el desarrollo de aplicaciones web y móviles. Sigue el patrón REST.
Visual Paradigm(S10) (*)	Generador de código fuente para diferentes lenguajes de programación. Se integra con IDE como NetBeans, Eclipse, Laika.
***JustBusiness* (14)**	Desarrollo de aplicaciones Android utilizando patrones de arquitectura *Naked Objects*, Navegación y operaciones CRUD.
OpenCLGen	Es un servicio web, es legible y puede integrarse en una aplicación existente o ejecutarse de forma independiente.
Codagen (S16)	Permite a las empresas generar automáticamente código fuente a partir de modelos UML, basado en MDA y patrón multicapa.
***Lyo code generator* (S24)**	Utiliza un estándar OASIS OSLC (Open Services for Lifecycle Collaboration) para la interoperabilidad de las herramientas de software. Sigue el patrón arquitectónico REST.
***JEEWiz* (S25)**	Genera sistemas de aplicaciones web casi completos a partir de modelos XML. Utiliza MDA y patrón multicapa.
***CreaCod* (S31)**	Generador de Código Fuente para controlar base de datos MySQL, SQL Server y Access en lenguajes JAVA, PHP y ASP
VULCAN(S10) (*)	Para aplicaciones móviles.
XMovil(S10) (*)	Para aplicaciones móviles.
Laika(S10)	Para aplicaciones móviles.

Nombre del Generador	Descripción
Adobe DreamWeaber(S10)	Permite el modelado de la lógica de negocios.
Power designer (S10) (*)	Generador de código para diferentes lenguajes de Programación.
***DOMMLite* (S23)**	Desde un modelo *DOMMLite* se puede generar una aplicación completa con navegación y operaciones CRUDS Utiliza patrón de arquitectura en capas.
***NHibernateMapper* (S29)**	Desarrollo de aplicaciones WEBGIS. *NHibernateMapper* se desarrolla principalmente para su aplicación en el campo de los Sistemas de información geográfica. Utiliza patrón multicapa.
WebRatio(S30)	Generador de código para aplicaciones WEB, entrada modelo

Tabla 29 Principales Nombres de los GCF.

Análisis y discusión:

En la tabla 29 se puede observar que cada GCF está relacionado con algún patrón de arquitectura; no todos los GCF de estudios encontrados tienen un nombre en particular, por eso no están incluidos en la tabla; sin embargo, describen el proceso de generación de código, así como los patrones, *frameworks* y las herramientas que utilizan. Algunos GCF están orientados exclusivamente para aplicaciones móviles; se ha encontrado un solo GCF que genera código del frontend y del backend, en los otros estudios solo generan código para la parte del backend o el frontend, pero no ambos. En algunos casos los GCF resultan siendo frameworks ya desarrollados, con diferentes grados de madurez, permiten la generación de código a partir del modelado de la solución.

5.2 Herramientas o componentes de los GCF (PI2)

¿Qué Herramientas se utiliza para la generación de código fuente?

EL objetivo de la pregunta de investigación 2 es identificar los lenguajes de programación usados para la generación de código(*input)* y los lenguajes de programación del código generados(*output*), base de datos, herramientas de modelado y las librerías o componentes de desarrollo utilizados en los estudios relevantes.

5.2.1 Lenguajes de programación para la construcción de GCF (PI2a)

¿Qué lenguaje de programación se ha utilizado para desarrollar el generador de código fuente?

El objetivo de esta pregunta es identificar los leguajes de programación que se utilizan para construir un generador de Código fuente. En base a la información obtenida, se ha identificado los principales lenguajes de programación utilizados para los generadores de código fuente, la **Tabla 30** muestra los principales leguajes utilizados en los estudios.

Lenguaje de programación del GCF	**Estudios**	**Conteo**	**%**
JavaScript	S1, S2	2	6.1%
C++	S4, S12, S17, S18, S22, S28, S30	7	21.2%
PHP	S19, S26, S28, S30	4	12.1%
Java	S2, S3, S4, S6, S7, S8, S9, S10, S11, S12, S13, S14, S15, S16, S18, S19, S20, S22 S24, S25, S27, S28, S29, S30, S33	25	75.5%
Python	S18, S21	2	6.1%

Lenguaje de programación del GCF	**Estudios**	**Conteo**	**%**
Perl	S18	1	3%
C#	S1, S4, S16, S19, S22, S28, S29, S32	8	24.2%
Visual Basic	S16, S22, S31	3	9.1%

Tabla 30 Lenguajes de programación para la Generación de Código fuente

Resultados

De la investigación realizada se ha encontrado que existe mayor desarrollo de generadores de código fuente en el lenguaje de programación abierto Java, con un 75.5% del total de estudios; en segundo lugar se encuentra el lenguaje de programación C# con 24.2%, en tercer lugar se encuentra el C++, con un 21.2%; también se ha encontrado desarrollos en otros lenguajes de programación como Python con un 6.1%, PHP con 12.1%, Visual Basic 9.1%, Perl con 3% y JavaScript con 6.1% del total de estudios.

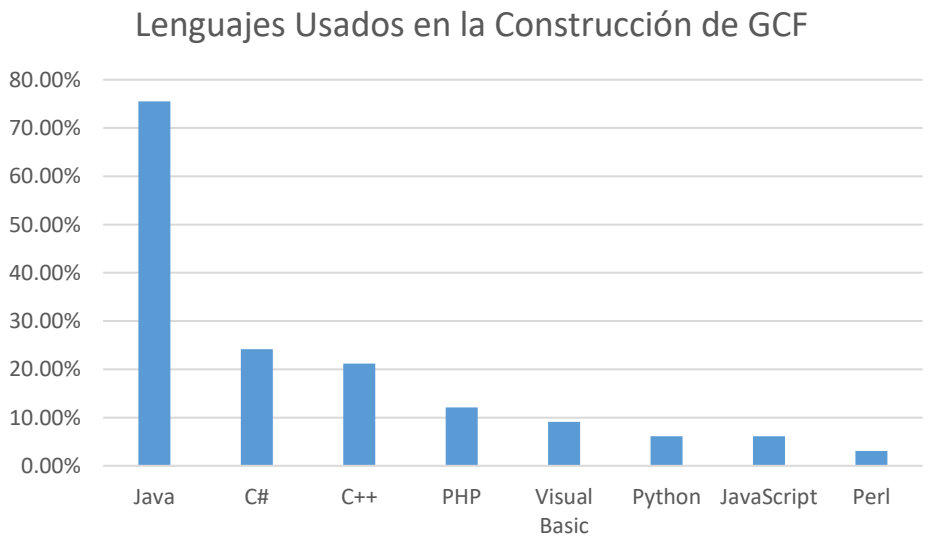

Ilustración 13- Importancia de los patrones de arquitectura

Análisis y Discusión

De acuerdo con la información obtenida, se puede concluir que el lenguaje de programación preferido para desarrollar GCF es Java, tiene muchas ventajas en comparación a otros lenguajes de programación [47]; entre las ventajas del Java podríamos mencionar en primer lugar que es un lenguaje de código abierto, es independiente de la plataforma (funciona en diferentes sistemas operativos), está orientada a objetos (código reutilizable), las aplicaciones graficas son independientes del dispositivo, posee un kit de desarrollo gratuito, esta soportada por una gran cantidad de desarrolladores en continuo crecimiento, el lenguaje tiene un gran número de librerías gratuitas que permiten conectividad con dispositivos móviles.

La elección del lenguaje de programación está muy relacionada con la elección de la arquitectura y/o el patrón de arquitectura utilizado en la construcción del GCF, es decir, si la tendencia es usar herramientas CASE de modelado y que estas puedan usarse en cualquier sistema operativo, entonces el lenguaje de programación será Java; si se quiere utilizar las ventajas de funcionalidad de Windows, entonces es recomendable utilizar un lenguaje de programación de la familia de Microsoft.

5.2.2 Lenguajes de Programación del código fuente generado (PI2b).

¿Cuáles son los lenguajes de programación generados por los GCF?

El objetivo de esta pregunta es identificar los lenguajes de programación que son generados por el GCF.

Los generadores de código fuente, pueden generar código para diferentes lenguajes de programación [10]. De acuerdo a los estudios se ha elaborado **Tabla 31**, que muestra los principales lenguajes de programación generados:

Lenguaje de programación generado por el GCF	Estudios	Conteo	%
JavaScript	S1, S9, S17, S28	3	12.1%
C++	S2, S4, S12, S18, S28	5	15.2%
Java	S2, S3, S4, S6, S7, S8, S10, S12, S13, S14, S15, S16, S17, S18, S19, S20, S24, S25, S27, S28, S29, S31, S33, S30	22	72.7%
C#	S4, S19, S28, S29, S32	4	15.2%
NesC	S5	1	3%
Python	S15, S18, S21, S24, S28	3	15.2%
Perl	S18, S28	1	6.1%
PHP	S19, S26, S31, S28, S30, S31	3	18.2%
Visual Basic	S22	1	3%
ASP	S31	1	3%

Tabla 31 Lenguajes de programación generados.

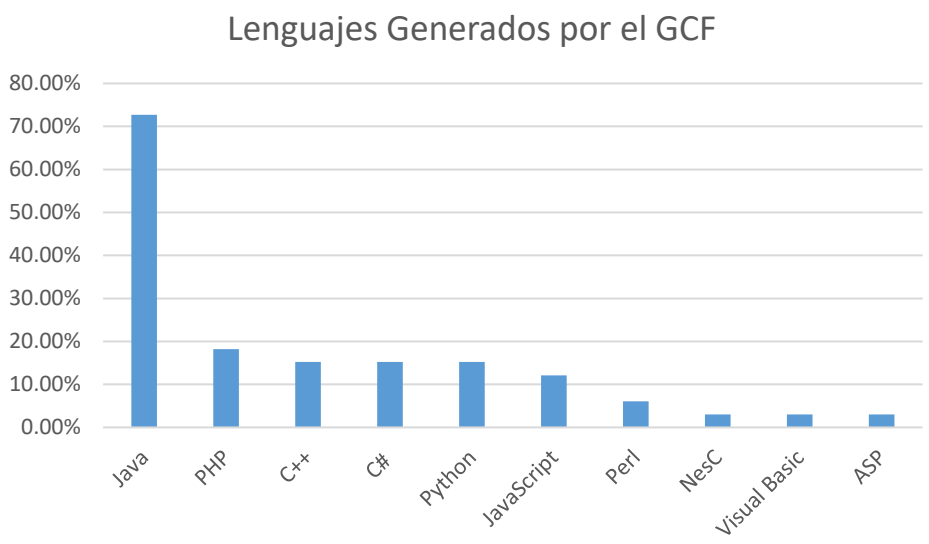

Ilustración 14- Lenguajes de programación generados por el GCF

Resultados

De la investigación realizada se ha encontrado que los GCF generan código para Java con un 72.7%, en segundo lugar se encuentra C++ con 16.1%, en tercer lugar se encuentra C# con 12.1%, en cuarto lugar se encuentran los lenguajes de programación PHP, Python; finalmente se ha encontrado que también existen generadores de código fuente que generan código para lenguajes de programación como ASP, Visual Basic, Perl y *NesC* en porcentajes menores al 3%.

Análisis y Discusión

Los GCF pueden generar código para diferentes lenguajes de programación, tomando en cuenta el resultado de los estudios se observa una preferencia por el lenguaje de programación Java.

Otro detalle que debe ser tomado en cuenta, está relacionado a la interfaz de desarrollo del lenguaje de programación, el nivel de madurez y facilidad de uso; así mismo, a nivel de estándares, existe más variedad de metodologías, *frameworks*, librerías y componentes para java, lo que puede ser una desventaja al momento de querer integrar y estandarizar la programación; la idea es que se tenga más opciones a la hora de elegir el lenguaje de programación.

5.2.3Gestores de Base de Datos usados en la GCF(PI2c)

¿Para qué gestores de base de datos se genera código fuente?

El objetivo de esta pregunta es identificar los gestores de base de datos asociados y utilizados en la generación de código fuente.

Se ha encontrado la siguiente información:

Gestores de base de datos	**Estudios**	**Conteo**	**%**
TinyDB	S5	1	3%
PostgreSQL	S9, S29, S30, S32	3	12.2%
SQLite	S9, S14	1	6.1%
Access	S19, S31, S33	3	9.1%
Oracle	S6, S21, S25	1	9.1%
SQL Server	S22, S29, S31	2	9.1%
Mysql	S29, S30, S31	2	9.1%

Tabla 32 Principales gestores de bases de datos encontradas.

Resultados

A nivel de herramientas de base de datos; se ha encontrado que los GCF generan código fuente que pueden integrarse con un determinado motor de base de datos, de los 33 estudios analizados 26 de ellos (79%) están orientados a la interacción con bases de datos, el otro grupo de GCF están orientados a la generación de código para áreas como el internet de las cosas (IoT), líneas de producción de software (SPL) y otros. El 36% de los GCF investigados permiten la generación de código que permite la integración con cualquier base de datos relacional, pudiendo ser una base de datos propietaria o una base de datos de código abierto; el 21.2% de los estudios están orientados a bases de datos de código abierto (PostgreSQL, MySQL, SQLite), y el 18.2% están orientados a bases de datos propietarias (Access, SQL Server, Oracle); el 3% está orientado a bases de datos orientado a aplicaciones móviles (*TinyDB*).

Análisis y Discusión:

Se ha tomado en cuenta la generación de código a nivel de Script SQL que genera el GCF para la creación de tablas y para la generación de código de operaciones básicas CRUD (*Create, read, update, delete*), para diferentes gestores de bases de datos, así como la generación de código para los principales lenguajes de programación.

A nivel de base de datos, el 79% de los GCF se han orientado a aplicaciones que utilicen gestores de bases de datos, con una preferencia por bases de datos relacionales múltiples(36%), y en el caso de un solo gestor de base de datos, existe preferencia por las bases de datos de código abierto o libres (21.2%) y en menor proporción (18.2%) a bases de datos propietarias o pagadas; por lo que en general podemos concluir que existe una preferencia por múltiples bases de datos de origen libre.

5.2.4Herramientas de modelado y librerías (PI2d).

¿Qué Herramientas y librerías son usados para desarrollar generadores de código fuente?

El objetivo de esta pregunta es identificar otras herramientas, librerías y lenguajes de marcado utilizados por los GCF, encontrándose la siguiente información:

Herramientas de Modelado	**Estudios**	**Conteo**	**%**
Visual Paradigm	S10	1	3%
Power Designer	S10	1	3%
VULCAN	S10	1	3%
Laika, XMobile	S10	1	3%
Together UML	S15	1	3%
Poseidon for UML	S15	1	3%
Objecteering UML	S15	1	3%
Enterprise Architecture	S16	1	3%
Rational Software	S3, S15, S17, S20, S24, S26, S33	7	21.2%
StarUML	S31	1	3%
MagicDraw UML	S6	1	3%
SysML	S7	1	3%
WebML	S30	1	3%

Herramientas de Modelado	Estudios	Conteo	%
Titan Architect	S9	1	3%
Eclipse's Papyrus	S5	1	3%
AUML Tool	S6	1	3%
State Graph Manipulators	S7	1	3%
XisUseCase,	S8	1	3%
Eden	S10	1	3%
CoSMIC (an MDA generative tool)	S5	1	3%

Tabla 33 Principales herramientas de modelado.

A nivel de librerías de desarrollo se ha encontrado la siguiente información.

Librerías / componentes	Estudios	Conteo	%
JDBC	S25, S27, S33		
NHibernateMapper	S29		
IPS (Instantiated Pattern Structure) and **AC (Application Component)**	S3	1	3%
RPNParser	S2	1	3%
Atlas Transformation **Eclipse Acceleo plugin** **Sun SPOTs**	S4	1	3%
ArchWiSeN	S5	1	3%

Librerías / componentes	Estudios	Conteo	%
OpenMP **Intel Threading Building Blocks**	S7	1	3%
IntelliJ, ApiGen, Laika, XMobile, VULCAN, Automatic Coder using Artificial Intelligence (ACAI)	S10	1	3%
Jade-Leap **Sun SPOTs**	S11	1	3%
iPOJO,	S12	1	3%
JBPM	S13	1	3%
Poseidon	S15	1	3%
Codagen	S16	1	3%
XML schema	S18	1	3%
ANTLR library.	S19	1	3%
MetaEdit **EJB**	S20	1	3%
Acceleo	S23	1	3%
EJB	S24	1	3%
FUJABA	S26	1	3%
jABC:	S29	1	3%
NHibernateMapper	S32	1	3%

Tabla 34 Librerías de Desarrollo encontradas.

Resultados

Respecto a las herramientas utilizadas en la generación de código fuente, y de la agrupación realizada se ha encontrado los siguientes resultados:

A nivel de herramientas de modelamiento; se ha encontrado 15 estudios que mencionan el uso de herramientas de modelamiento (S3, S5, S6, S7, S8, S9, S10, S15, S16, S17, S20, S24, S26, S31, S33); de los estudios analizados se mencionan a 18 herramientas de modelamiento, las cuales en la mayoría de los casos generan directamente el código a nivel de base de datos, es decir permiten crear las tablas, columnas e índices en la base de datos; en algunos casos estas herramientas de modelamiento se comportan como GCF, es decir generan código fuente para un determinado lenguaje de programación, en otros casos generan un archivo XML o archivo de texto el cual es utilizado como input por otras aplicaciones del tipo GCF, los cuales toman como referencia el archivo XML y luego generan código fuente para un determinado lenguaje de programación. La herramienta de modelamiento a la que más se hace referencia es Racional Software, representando el 21.2% de todos los estudios analizados, después las otras herramientas no tienen mayor representatividad, sólo representan el 3% del total de estudios analizados.

A nivel de librerías o componentes de desarrollo; se ha encontrado 21 estudios que mencionan el uso de este tipo de programas (S1, S2, S3, S4, S5, S6, S7, S10, S11, S12, S13, S15, S16, S18, S19, S20, S23, S24, S26, S29, S32);estas librerías o componentes se agregan a las herramientas de modelado para generar código fuente, o son incorporados a los entornos de desarrollo para lograr la conexión a las herramientas de modelado y/o los archivos XML que estas generan; de lo revisado estas librerías y componentes se han usado y/o desarrollado para cada estudio en particular, ninguna de estas librerías o componentes ha sido tomado en cuenta por otro estudio.

Análisis y Discusión:

En el caso de las herramientas de modelado, se ha encontrado que son herramientas muy útiles y necesarias para hacer una representación de la solución que se pretende implementar, además estas herramientas permiten la generación de código para bases de datos o para un determinado lenguaje de programación, en este campo las herramientas más utilizada son de Rational Software, que permite hacer el modelamiento considerando todo el ciclo de vida del software, sin embargo al ser un producto propietario se debe considerar el costo de las licencias para utilizarlo en el desarrollo de aplicaciones empresariales y personalizables; por otra parte, existe una serie de *frameworks* y librerías que facilitan la generación de código fuente, estas librerías y componentes necesariamente se integran con un determinado entorno de desarrollo integrado (IDE), y permiten de alguna forma conectar los diagramas de modelo y los IDE, para finalmente generar código fuente personalizable y orientado a las aplicaciones empresariales.

5.2.5 Entornos de Desarrollo Integrado IDE (PI2e).

Según los estudios, ¿Que Entornos de Desarrollo se utilizan para el desarrollo de GCF?

Se encontraron los siguientes IDE:

IDE	Estudios	Conteo	%
IntelliJ IDEA	S10	1	3%
Eclipse	S6, S8, S10, S13, S14, S24, S26, S23	5	15.2%
Rational Application Developer	S7	1	3%
JetBrain	S10	1	3%
NetBeans	S10	1	3%

IDE	Estudios	Conteo	%
Microsoft Visual Studio	S4 16 19 25 28 29		

Tabla 35 Entorno de Desarrollo Integrados.

Resultados:

En relación a los Entornos de Desarrollo Integrado (IDE), se ha encontrado 5 herramientas de desarrollo, siendo Eclipse la herramienta que más se utiliza, con una participación de 15.2%, otras herramientas que se utilizan son: Java Studio, IntelliJ, IBM *Rational application Developer, jetBrain*, todas estas funcionan en la plataforma de Java; para la plataforma de .Net Framework de Microsoft, el IDE utilizado para hacer desarrollo es el Microsoft Visual Studio.

Análisis y Discusión:

Las librerías y componentes funcionan en un determinado entorno de desarrollo integrado (IDE), y permiten de alguna forma conectar los diagramas de modelo y los IDE, para finalmente generar código fuente personalizable y orientado a las aplicaciones empresariales. Al respecto, aún no existe un GCF que genere código a partir de los modelos y que genere código para el backend y frontend, por lo que en cada estudio se ha tratado de tener una herramienta que facilite esta labor.

5.3 *Frameworks* (PI3)

¿Qué *Frameworks* se han empleado en la generación de código fuente?

El objetivo de esta pregunta es describir los principales *frameworks* usados para generar código fuente. Como se menciona en el estudio (S10) [10], la generación de código fuente es una parte importante de la Ingeniería de software y se sugiere utilizar diferentes métodos, técnicas y enfoques para desarrollarlos. Se

ha demostrado que el uso de marcos y componentes es eficaz para mejorar la productividad y la calidad del software S12 [32].

El anexo 01 contiene información con los principales *frameworks* encontrados. A continuación, la

Frameworks	Estudio	Conteo	%
PhoneGap	S1	1	3%
AndroMDA	S6, S11, S22	3	9.1%
Microsoft NET Framework.	S4, S8, S16, S19, S25, S28, S29	7	21.2%
J2EE (Hibernate, **Spring, struts, JSP for client)**	S2, S4, S8, S11, S13, S16. S25	7	21.2%
ArchWiSEN	S5	1	3%
VERTAF / Multi-Core (VMC)	S7	1	3%
Titan Framework	S9	1	3%
iPOJO-OSGi Framework.	S12	1	3%
FraCC Framework	S12	1	3%
JustBusiness framework	S14	1	3%
Django framework	S15, S39	2	6.1%
NHibernateMapper	S29	1	3%
SCT dynamic frames	S18	1	3%
openArchitectureWare (oAW)	S23	1	3%
(OPRF) Framework.	S33	1	3%

Tabla 36 muestra los Framework encontrados para la generación de código fuente.

Frameworks	Estudio	Conteo	%
PhoneGap	S1	1	3%
AndroMDA	S6, S11, S22	3	9.1%

Microsoft NET Framework.	S4, S8, S16, S19, S25, S28, S29	7	21.2%
J2EE (Hibernate, **Spring, struts, JSP for client)**	S2, S4, S8, S11, S13, S16. S25	7	21.2%
ArchWiSEN	S5	1	3%
VERTAF / Multi-Core (VMC)	S7	1	3%
Titan Framework	S9	1	3%
iPOJO-OSGi Framework.	S12	1	3%
FraCC Framework	S12	1	3%
JustBusiness framework	S14	1	3%
Django framework	S15, S39	2	6.1%
NHibernateMapper	S29	1	3%
SCT dynamic frames	S18	1	3%
openArchitectureWare (oAW)	S23	1	3%
(OPRF) Framework.	S33	1	3%

Tabla 36 Principales Frameworks para la GCF.

Resultados:

Los *frameworks* que más se usan para el desarrollo de aplicaciones web según los estudios son: J2EE y .NET con 21.2% seguido por AndroMDA que alcanza 9.1 %; los demás presentan un framework por cada estudio. Entre los principales *frameworks para generar aplicaciones* encontrados podemos mencionar los siguientes:

PhoneGap, es un *framework* de desarrollo de código abierto, crea aplicaciones hibridas para dispositivos móviles multiplataforma con tecnologías HTML5, CSS3 y JavaScript, que son portables e independientes de la plataforma (S1) [24]. En el mismo estudio se menciona el *framework Titanium* que tiene las mismas características del marco PhoneGap.

JustBusiness, un marco para desarrollo de aplicaciones Android utilizando el patrón *Naked Objects*. Ofrece beneficios como la generación automática de interfaces de usuario y de código de CRUD, agilizando así el proceso de desarrollo de una aplicación móvil (S14) [33].

En el framework J2EE se puede implementar aplicaciones con diferentes patrones de arquitectura como multicapas o MVC; en el estudio (S2) presenta la arquitectura de una aplicación web típica conformada por: *front-end, middletier y back-end;* en este sentido. *l*os estudios (S2), (S4), (S8), (S11), (S13), (S16) y (S25) que también están basados en el framework J2EE, utilizan otros *frameworks* de código abierto como: *struts, spring, hibernate* para la implementación de la plataforma J2EE. Struts se utiliza para la capa de vista web. Spring es responsable de la configuración y gestión de toda la aplicación. *Hibernate* hace el trabajo de mapeo de base de datos ORM (CRUD empaquetado) (S13) [15].

El .NET Framework (.Net) se añade al sistema operativo Windows de Microsoft, este *framework* incluye componentes para la interfaz de usuario, acceso a datos,

reglas de negocio, entre otros. Es usado en los estudios (S4), (S16), (S19), (S28) y (S29).

Otro framework que ha sido identificado es *AndroMDA*, partir de modelos UML, se pueden transformar componentes que se despliegan en plataformas como J2EE o .NET. AndroMDA permite generar métodos que luego se pueden implementar con la lógica empresarial de la aplicación; sin embargo, no se genera el cuerpo de esos métodos, este marco se basa en la MDA; los estudios (S6) [28],(S11), (S22) y (S23) Presentan este enfoque.

La GCF en el estudio (S25) [42] se realiza a través de la herramienta JEEWIZ, también basado en MDA, permite generar aplicaciones web a partir de modelos; y utiliza los *frameworks* para .Net o J2EE.

El marco ArchWiSeN se utiliza para generar código fuente en una plataforma WSAN[1], (S5) [27]; este marco puede generar código para aplicaciones en dominios diferentes.

En el estudio (S9) [18] se aplicó el *Titan Framework*, para aplicaciones móviles, que implemento dos tareas: un bus de servicio, que contempla la comunicación entre las aplicaciones web y sus aplicaciones móviles correspondientes y la implementación de un generador de código fuente para aplicaciones móviles utilizando arquitectura *Titan Architect Mobile.*

Entre los diversos *frameworks* web basados en el patrón MVC, Django admite el desarrollo rápido de programas con un diseño realista. El *framework* Django, basado en Python, es ligero y más rápido en procesamiento de objetos solicitados frente a otros marcos. En el marco de Django, el módulo Controlador es parte del marco que maneja la invocación de las funciones Python requeridas, la mayor parte del trabajo se lleva a cabo en los modelos, plantillas y vistas (S23).

El marco SCT presentado en S18 [37], define tres componentes: Especificación, Configuración y un conjunto de Plantillas, se utiliza Python para la

[1] Wireless sensor and actuator networks

implementación del generador estos tres componentes se presentan en forma de listas de Python.

En (S12) [32] se aprecia el proceso de desarrollo propuesto para aplicaciones CBSE[1], basado en MDA; el estudio realiza dos casos de estudio, en el primero aplica *iPojo frameworks* y en el segundo *FracCC*.

Genesys es un *framework* orientado a servicios, para la construcción de generadores de códigos de alto nivel. Este enfoque orientado a servicios, hace generadores de código robustos y de alta calidad al confiar en una gran biblioteca de bloques de construcción altamente reutilizables [44].

El estudio (S7) presenta VERTAF / Multi-Core (VMC), un marco de aplicación para el desarrollo de software integrado multi-core. Adopta un enfoque basado en modelos con generación automática de código a partir de modelos SysML. El código generado por VMC utiliza las API de Quantum framework y la biblioteca de bloques de creación de subprocesos de Intel junto con un sistema operativo que admite procesadores de varios núcleos como Linux. VMC muestra lo fácil que es desarrollar software integrado para procesadores de varios núcleos. Utilizamos un ejemplo del mundo real, a saber, un sistema de grabación de video digital (DVR) [48].

openArchitectureWare (oAW) es un conjunto de herramientas y componentes que ayudan al desarrollo de software basado en modelos, basado en MDA, implementado en Java que admite modelos, metamodelos y formatos de salida (código). Las herramientas de apoyo (como editores y navegadores de modelos) se basan en la plataforma Eclipse. openArchitectureWare es una herramienta para crear herramientas MDA; se utiliza en todo tipo de dominios, como aplicaciones web basadas en J2EE, aplicaciones basadas en Spring, aplicaciones basadas en Eclipse en varios dominios, sistemas basados en C ++, Java, Python, Aplicaciones .NET, arquitectura SOA Enterprise, etc. [6].

[1] Component-based software engineering.

NHibernateMapper se convierte en una herramienta que permite el rápido desarrollo de la capa de acceso a datos para soluciones GIS interoperables (S29) [44]. NHibernateMapper se ha implementado completamente en C # con Microsoft .NET Framework. Como resultado, se crea la biblioteca de clases NHibernateMapper.dll. Todas las clases de programación en la biblioteca podrían dividirse en dos grupos principales: clases para recuperación de esquema de base de datos y clases para la manipulación de archivos de mapeo, la herramienta es capaz de atravesar automáticamente el esquema de la base de datos y la biblioteca de clases de dominio proporcionados para coincidir y sugerir correspondencias. La GUI interactiva ofrece la capacidad de aceptar o rechazar correspondencias sugeridas, así como de definir manualmente las asignaciones para podarlas en una forma final.

El Framework OPRF consta de: el componente de ejecución de SQL, que proporciona la conexión de la base de datos y la conversión de datos; el componente de conexión de la base de datos, que proporciona un mecanismo para adjuntar y desconectar a o desde un servidor de aplicaciones web sin cambios en el código fuente; y el generador de código fuente de acceso a la tabla, que genera los componentes de acceso a la tabla (S33) [46].

Análisis y discusión.

Se ha encontrado gran variedad de frameworks relacionados a la generación de código fuente, es conveniente diferenciar los Frameworks de aplicación de los frameworks para generar aplicaciones para no confundirlos.

Los *frameworks* de aplicación, son infraestructuras orientados a brindar una plataforma para que funcionen las aplicaciones en un determinado sistema operativo, tal es el caso de .Net Framework, el cual ha sido desarrollado por Microsoft y funciona sobre los diferentes sistemas operativos Windows; existe también otro *framework* muy utilizado como es el J2EE, el cual fue desarrollado por Sun Microsystems, tiene la particularidad de que es multiplataforma (funciona en varios sistemas operativos) y está orientado al desarrollo de aplicaciones de código abierto.

Los *frameworks* para generar aplicaciones, permiten generar código fuente a partir de un modelo gráfico, generalmente elaborado con una herramienta CASE; también se ha encontrado estudios donde se ha generado código fuente a partir de una estrada como un archivo de formato XML.

De los estudios revisados, se intenta hacer un resumen de los procesos más comunes que se ha seguido para la creación o planteamiento de los GCF, podemos considerar las siguientes etapas:

1. Identificación o definición de la solución (Dominio).
2. Modelamiento de los requerimientos, haciendo uso de herramientas CASE que permiten diseñar el modelo del dominio en UML.
3. Transformación o explotación del modelo del dominio a diagramas de clases y modelos de datos.
4. Generación de código fuente.

Se toma como input dos tipos de modelos: Modelos o diagramas de datos orientados a la generación de código SQL [6] y diagramas o modelos de clase, orientados a la generación de código para un determinado lenguaje de programación [37]. La generación de código se puede realizar de varias formas: Haciendo uso de herramientas CASE las cuales permiten no solo diseñar si no también generar código SQL para la base de datos y código para un determinado lenguaje de programación, sin embargo, debido a las limitaciones de personalización y a la necesidad de integrar el código generado con librerías o componentes externos, resulta que estas herramientas no se utilizan en la práctica [10]; para suplir esta necesidad, algunos estudios plantean la creación o utilización de un software GCF intermedio, estos tipos de programa se caracterizan por tomar como input un archivo XML o texto que es generado por una herramienta CASE [30], a partir de este archivo se genera el código fuente para un lenguaje de programación en particular, sin embargo este tipo de aplicaciones también presenta algunas limitaciones, ya que está orientado a la parte de generación de clases y no considera la generación de código SQL, otra

desventaja es la necesidad de conocer la lógica de funcionamiento del aplicativo en detalle.

Uno de los temas que se ha encontrado en los GCF está relacionado a la parte del mantenimiento del programa, es decir, cuando el sistema ya está en producción, el modelo de datos y clases de las herramientas CASE no está sincronizado con las tablas de la base de datos, por lo que se podría asumir que los GCF tienen ventaja en la parte inicial de la construcción de un sistema empresarial, pero no tiene mucho uso en la parte del soporte y mantenimiento [32]; los f*rameworks* no se pueden modificar o actualizar por el usuario.

5.4 Aplicaciones de los GCF (PI4)

De acuerdo a los estudios seleccionados los GCF tienen aplicaciones diversas tanto en la industria como en área académica; la mayoría de los estudios están orientados a desarrollo de software tomando en cuenta la arquitectura en la que están diseñados.

Según los estudios, ¿Cuáles son las principales aplicaciones de los GCF como herramienta para desarrollo de Sistemas informáticos?

Estudio	**Aplicación**
S1, S9, S14, S33	Desarrollo Móvil multiplataforma. Implementación de aplicaciones web portátiles
S2	Generación de prototipos modificable para una aplicación web.
S4	Reutilización de código.
S5	Desarrollo de aplicaciones WSAN [1]genéricas desde dos puntos de vista: dominio y red.

[1] Wireless sensor and actuator networks.

Estudio	Aplicación
S6	Integración y Cooperación de Amazon para la Modernización del Monitoreo Hidrológico. Este proyecto se encuentra actualmente bajo el desarrollo del Instituto Brasileño de Tecnología Aeronáutica.
S7	Construcción de aplicaciones integradas.
S8	Desarrollo de aplicaciones interactivas.
S11	Desarrollo de servicios GRID.
S12	Desarrollo de aplicaciones basadas en componentes.
S16, S19, S18	Sistemas de aplicaciones web casi completos a partir de modelos (o especificaciones)
S13	Transformación de un formulario web estático en un formulario web dinámico. Desarrollo de aplicaciones a gran escala.
S15	Generación automática de programas OpenCL para dispositivos de GPU.
S17, S27	Desarrollo de aplicaciones de comercio electrónico. de empresa a empresa. Dichas aplicaciones se pueden clasificar como informativas, transaccionales o colaborativas. Aplicaciones *e-commerce.*
S23, S28, S30	Generación de una aplicación a gran escala/alto nivel con navegación y operaciones CRUDS.
S24	Generación de aplicaciones web para encuestas.
S25, S26, 27	generar sistemas de aplicaciones web casi completos a partir de modelos (o especificaciones).
S31	Para gestión de información.
S29	Desarrollo de Aplicaciones GIS
S32	Servicios web
S22	Genera código para la capa de acceso a datos.

Tabla 37 Aplicaciones de los generadores de código.

Resultados.

De los estudios el 81.8% evidencian la aplicación de los GCF en las distintas áreas del desarrollo académico e industrial. Mientras que 18.2% solo muestra metodologías o procesos para la GCF.

Según los estudios la aplicación de GCF para la construcción de software, está inmersa en varias áreas del desarrollo científico e industrial, algunas de las aplicaciones son: Los estudios (S1) [17], (S9) [18]) , (S14) y (S33) evidencian la aplicación de GCF en la construcción de aplicativos móviles, el estudio (S7) [48] en la construcción de aplicaciones multiplataforma**.** Los estudios (S6) [28] y (S11) [31] para la construcción de sistemas multi agente (IoT), en los estudios (S17) [36]y (S27) [11] los GCF se aplican para el desarrollo de comercio electrónico, (S32) [45] en la construcción de aplicaciones GIS; también están los estudios (S23), (S28), (S29), (S30) para aplicaciones a gran escala considerando las operaciones CRUD.

Análisis y discusión.

Los GCF se están utilizando en el desarrollo de aplicaciones de múltiples áreas, desde aplicaciones convencionales para equipos de cómputo, aplicaciones móviles, aplicaciones orientados al internet de las cosas y aprendizaje automático (*machine learning*).

Los GCF siguen evolucionando, existe una tendencia creciente de desarrollar aplicaciones para que personas que no son programadores, puedan generar su propia aplicación. Así como existen GCF orientados a no programadores, también existe una serie de herramientas orientados a programadores permitiendo generar código en diferentes lenguajes de programación, sin embargo, hasta la fecha y según los estudios revisados no se ha encontrado una herramienta comercial o libre que genere código para un determinado patrón de arquitectura considerando la parte del backend y el frontend, por lo que aún existe una serie de necesidades para que los GCF orientados a los programadores se sigan desarrollando y evolucionando.

6 Trabajos futuros y conclusiones

6.1 Conclusiones.

Se ha planteado y desarrollado el protocolo de revisión para presentar los resultados de la revisión sistemática de la literatura sobre Generadores de código fuente y su relación con los patrones de arquitectura, para ello se ha realizado una síntesis de la situación actual de los generadores de código fuente, incluyendo ítems sobre su importancia, los patrones de arquitectura que utilizan y/o generan, los principales *frameworks*, las herramientas y su aplicación en el mundo real.

Luego de realizar la selección de los estudios relevantes, en siete bases de datos electrónicas, se obtuvo 33 estudios que cumplieron con los criterios de inclusión y exclusión; también se realizó la evaluación de la calidad aplicando las preguntas planteadas en el protocolo de revisión.

Se llega a la conclusión de que los GCF permiten automatizar la generación de código, facilitando el desarrollo y mantenimiento de aplicaciones, reduciendo el tiempo y minimizando errores de programación, por lo tanto, generando código de calidad; permitiendo además reutilizar la herramienta en otros proyectos de desarrollo. Las ventajas de utilizar los GCF son: La reutilización (45.5%), alto nivel de abstracción (67.3%), la aplicación en sistemas a gran escala (67.3%), reducción de costo y/o tiempo (45.5%). La aplicación de los GCF en el mundo real abarca muchos campos entre ellos se puede mencionar: Desarrollo de aplicaciones móviles, implementación de aplicaciones web portátiles, aplicaciones *e-commerce*, desarrollo de aplicaciones GIS, programación multi core, gestión de información, construcción de sistemas multi agente, sistemas de información a gran escala, entre otros. (PI11)

El uso de patrones de arquitectura en la construcción de aplicaciones es fundamental para obtener software de calidad, mejoran la planificación y diseño de la estructura de las aplicaciones, ya que facilitan la organización del código

de una manera estandarizada, facilitando su programación, mantenimiento y pruebas; es bajo este escenario donde se puede sacar mayor provecho el uso de los generadores de código fuente. Así mismo, se ha encontrado que existe una relación entre los patrones de arquitectura y algunas características de calidad del aplicativo, como son la funcionabilidad, usabilidad, mantenibilidad, configuración, fiabilidad, etc. (PI2)

En cuanto a los patrones de arquitectura con mayor porcentaje de uso encontrados están el patrón multicapa (30,2%), MVC (24.2%); se encontraron evidencias de arquitecturas emergentes como, REST, MVVM, SOA, y microservicios (PI1C).

Respecto a las herramientas utilizadas en la GCF, se ha identificado que el lenguaje de programación con mayor porcentaje de uso para la construcción de GCF es Java (75.5%); así mismo, el lenguaje del código fuente más generado es Java (72.7%). Entre los gestores de bases de datos, PostgreSQL (12.2%) tiene mayor porcentaje de aceptación; la herramienta de modelamiento preferida es *Rational Software*, la librería o componente más utilizado para la conexión con la base de datos es el JDBC (9%), finalmente el entorno de desarrollo integrado preferido es el Eclipse (15.2%). (PI2)

Los frameworks de aplicación al que están más orientados los GCF son .Net Framework (18%, plataforma Microsoft) y J2EE (18%, plataforma Java), una diferencia importante que se ha encontrado es que .Net Framework trabaja con Visual .Net (producto con librerías integradas) y J2EE puede utilizar diversos entornos de desarrollo (diferentes productos con librerías estándar que permiten su integración). Existen otros frameworks de aplicación Web, los cuales se caracterizan porque permiten la generación de código siguiendo una arquitectura o patrón de arquitectura, entre los más resaltantes podemos encontrar a AndroMDA(15%), framework de generación de código fuente abierto que sigue el paradigma de la Arquitectura Dirigida por Modelos (MDA), genera código a partir del diseño o modelado de los componentes, esto quiere decir que el modelamiento se puede hacer para cualquier patrón de arquitectura; existen

también trabajos independientes que permiten generar código a partir de un modelo o mediante una conexión directa a la base de datos, sin embargo estos productos aun no son maduros ya que se han aplicado para una solución en particular.(PI3)

De acuerdo al objetivo central de la investigación se ha desarrollado una RSL, con lo que se concluye de que el desarrollo de aplicaciones hacen uso de herramientas como patrones de arquitectura, GCF, y frameworks que ayudan a disminuir los problemas en la estructura, funcionamiento e integración de los componentes de software.

6.2 Trabajos futuros.

En los estudios revisados, a la fecha no se ha encontrado evidencia de que algún *framework* o GCF genere código para todas las capas de un determinado patrón de arquitectura, en el estudio (S1) menciona que genera código para el patrón de Arquitectura MVVM; pero está limitado a plataformas móviles, la implementación se realiza a través del *framework* de desarrollo *Phonegap* y está orientado principalmente al *frontend* o la capa de presentación; otros GCF y *frameworks,* por ejemplo los estudios (S30 y S31), generan código fuente para un patrón de arquitectura MVC, pero solo a nivel de *backend* para las capas de acceso a datos y reglas del negocio; no generan código para la capa de interfaz o *frontend*; no se ha encontrado un GCF o *framework* que genere el código para el *frontend* y *backend* de manera integrada y para un patrón de arquitectura determinado.

Desde el punto de vista de la ingeniería de software, es recomendable la creación de un GCF o *framework* que permita la generación de código a partir de la integración del modelo de clases y el diseño de interfaces, para un determinado patrón de arquitectura de una manera automatizada, considerando todas las capas del *frontend* y *backend* de una manera integrada, con el objetivo de reducir el tiempo de programación, y dejar que las personas se concentren en la optimización de los procesos, en el análisis y una adecuada planificación de la solución de un problema se software.

La generación automática de código fuente para aplicaciones web, podría resultar más útil con el uso de tecnologías emergentes como: servicios web (REST o SOA), para la parte del *backend* que utiliza diferentes librerías y componentes web de diferentes fabricantes y que se actualizan continuamente; en este sentido los GCF también deberían tener una base de datos de librerías que permitan la actualización automática, similar al funcionamiento de los antivirus.

Por otra parte, continuamente aparecen nuevos requerimientos que deberían ser soportadas por los GCF; como, generar código que ofrezca una visualización adecuada de la información en diversos dispositivos electrónicos conocido como “web responsive”; o que tengan la posibilidad de funcionar offline (sin internet), desplegar notificaciones, etc. Sería interesante investigar y buscar una solución adecuada en este campo.

Desde el punto de vista de la arquitectura de software, se debería estandarizar las diferentes capas de los patrones de arquitectura, establecer un conjunto de conceptos y buenas prácticas para el desarrollo de aplicaciones diferenciándolas por tipo de aplicación (aplicaciones web, móviles, orientadas al internet de las cosas, etc.), se debe buscar integrar y facilitar la transición entre el modelado de procesos con el modelado de software, así mismo, se debe integrar y mejorar la notación de modelado entre el modelo de clases y el diseño de las interfaces.

Finalmente, se debe considerar la realización de un catálogo de herramientas de desarrollo y arquitectura de manera simple, organizada, clasificada y de fácil entendimiento, para que pueda ser utilizado por los diferentes actores que integran el grupo de desarrollo se software.

7 Bibliografía.

[1] Microsoft, «MSDN Microsoft,» Microsoft, 11 2018. [En línea]. Available: https://msdn.microsoft.com/es-es/hh144976.aspx. [Último acceso: 15 05 2018].

[2] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerland, Michael Stal, Pattern-Oriented Software Architecture., Germany: JOHN WlLEY & SONS , 2001.

[3] J. T. Mora, *Arquitectura de software para aplicaciones Web-Tesis,* Mexico: Unidad Zacatenco, 2011.

[4] Eduardo Chávez, Edgar Hermoza y César Villacís., «Generador de Código Fuente para Gestión de Información de MySQL, SQL Server y Access para JAVA, PHP y ASP,» *Revista" GEEKS"-DECC-Report,* vol. 4, nº 1, p. 45, 2013.

[5] Daniel Strmečki, Ivan Magdalenić and Danijel Radosević, «A Systematic Literature Review on the Application of Ontologies in Automatic Programming,» *International Journal of Software Engineering and Knowledge Engineering,* vol. 28, nº 05, pp. 559-591, 2018.

[6] Igor Dejanović, Gordana Milosavljević, Branko Perišić, and Maja Tumbas, «A domain-specific language for defining static structure of database applications,» *ComSIS ,* vol. 7, nº 3, pp. 410-440, 2010.

[7] D. d. J. M. Acosta, «Herramienta para la generación automática de código fuente para aplicaciones con arquitectura MVC bajo el desarrollo dirigido por modelos textuales(MDD,» Universidad Nacional de Colombia , Bogota, 2014.

[8] Vitor E. Silva Souza, Ricardo A. Falbo, Giancarlo Guizzardi, «Designing Web Information Systems for a Framework-based Construction.,» *IGI Global,* vol. 4, p. 34, 2010.

[9] Magdalenić Ivan, Danijel Radošević, and Dragutin Kermek, «Implementation model of source code generator,» *Journal Comunications of Software and Sistems,* vol. 7, nº 2, pp. 71-79, 2011.

[10] Rosales-Morales, Viviana Yarel ; Alor-Hernandez, Giner ; Garci¬a-Alcaraz, Jorge Luis ; Zatarain-Cabada, Ramon ; Barron-Estrada, Maria Luci¬a, «An analysis of tools for automatic software development and automatic code

generation,» *Revista Facultad de Ingeniería Universidad de Antioquia,* vol. 10, nº 77, pp. 75-87, 2015.

[11] V. Tarau, «A framework for automatic generation of web based data entry applications based on XML,» de *Proceedings of the 2002 ACM symposium on Applied computing*, Alemania, 2002.

[12] Havva cetmer AL TIP ARMAK ,Busra TOKGOZ ,Okkes Emm BALCICEK - , Ashhan OZKA Y A ,Prof. Dr. Ahmet ARSLAN, «Source Code Generation For Large Scale Applications,» de *2013 The International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE)*, Turkey, 2013.

[13] B. Kernighan and P. Plauger, Software tools in Pascal, Boston, USA: Addison-Wesley, 1981.

[14] O. P. a. J. Štastný, Autmatic Generation of Programs, Advances in Computer Science, Czech Republica : Dr. Matthias Schmidt, 2011.

[15] Song, Yong Chao ; Wu, Bu Dan ; Chen, Jun Liang, «The Design and Implementation of Code Generation Based on J2EE in the Development of JBPM Workflow System,» *Applied Mechanics and Materials,* Vols. %1 de %2263-266, pp. 1961-1968, 2013.

[16] C. B. Reynoso, «Introducción a la Arquitectura de Software,» *Universidad de Buenos Aires,* vol. 1, p. 245, 2004.

[17] C. O. Morales, «A Heuristic Approach to Architectural Design of Software-Intensive Product Platforms,» de *Advances in Product Family and Product Platform Design* , USA, Springer, 2014, pp. 647-681.

[18] Carromeu, Camilo ; Barroso Paiva, Debora ; Cagnin, Maria Istela, «From e-Gov Web SPL to e-Gov Mobile SPL,» de *International Conference on Computational Science and Its Applications*, Brasil, 2016.

[19] K. B., Guidelines for performing Systematic Literature Reviews in Software Engineering, Keele University: EBSE Technical Report, 2007.

[20] PETTICREW, M.; ROBERTS, Systematic Reviews in the Social Sciences: A Practical Guide., Blackwell, 2006.

[21] Webster, J. And Watson, «Analyzing the past to prepare for the future:Writing a Literature Review,» *MIS Quarterly,* vol. 26, nº 2, pp. MIS Quarterly Vol. 26 No. 2, pp. xiii-xxiii/June 2002, 2002..

[22] I. Crombie, The Pocket Guide to Appraisal, BMJ Books, 1996.

[23] A. Fink, «Conducting Research Literature Reviews. From the Internet to Paper.,» de *Sage Publications.*, 2005..

[24] Francese, R. and Risi, M. and Tortora, G. and Scanniello, G., «Supporting the development of multi-platform mobile applications,» de *15th IEEE International Symposium on Web Systems Evolution (WSE)*, Eindhoven, Netherlands, 2013.

[25] Jia Zhang, Jen-Yao Chung,Carl K. Chang, «Towards Increasing Web Application Productivity,» de *Proceedings of the 2004 ACM symposium on Applied computing*, USA, 2004.

[26] Uwe van Heescha, Paris Avgeriou, Uwe Zdun, Neil Harrison, «The supportive effect of patterns in architecture decision recovery-A controlled experiment,» *Science of Computer Programming,* Vols. %1 de %2Volume 77, , nº 5, pp. 551-576, 1 May 2012.

[27] Rodrigues, Taniro and Delicato, Flavia C. and Batista, Thais and Pires, Paulo F. and Pirmez, Luci, «An approach based on the domain perspective to develop WSAN applications,» *Software & Systems Modeling,* vol. 16, nº 4, p. 949–977, 2017.

[28] Breno Lisi Romano, Gláucia Braga e Silva, Adilson Marques da Cunha and Walter Itamar Mourão, «Applying MDA development approach to a Hydrological Project,» de *Seventh International Conference on Information Technology: New Generations*, USA, 2010.

[29] Shih, Chihhsiong ; Wu, Chien-Ting ; Lin, Cheng-Yao ; Hsiung, Pao-Ann ; Hsueh, Nien-Lin ; Chang, Chih-Hung ; Koong, Chorng-Shiuh ; Chu, William C., «A Model-Driven Multicore Software Development Environment for Embedded System,» de *International Computer Software and Applications Conference*, USA, 2009.

[30] Rodrigues da Silva, Alberto ; Saraiva, Joao ; Silva, Rui ; Martins, Carlos, «XIS-UML Profile for eXtreme Modeling Interactive Systems,» de *Workshop on Model-Based Methodologies for Pervasive and Embedded Software (MOMPES'07)*, Portugal, 2007.

[31] Mizuta, S. ; Runhe Huang, «Automation of Grid Service Code Generation with AndroMDA for GT3,» de *International Conference on Advanced Information Networking and Applications (AINA'05)* , Taiwan, 2005.

[32] Alonso, Diego ; Sanchez-Ledesma, Francisco ; Sanchez, Pedro ; Pastor, Juan A ; Alvarez, Barbara, «Models and Frameworks: A Synergistic

Association for Developing Component-Based Applications,» *The Scientific World Journal,* p. 17, 2014.

[33] Fabiano Freitas, Paulo Henrique M. Maia, «JustBusiness: A Framework for Developing Android Applications using Naked Objects,» de *IX Brazilian Symposium on Components, Architectures and Reuse Software*, Brasil, 2015.

[34] Deepika H.V, Mangala N, N. Sarat Chandra Babu, «Automatic Program Generation for Heterogeneous Architectures,» de *Intl. Conference on Advances in Computing, Communications and Informatics (ICACCI*, India, 2016.

[35] Codagen Technologies Corp, *Codagen Technologies Launches Tool to Lower Total Cost of Ownership for Software Development Projects,* Canada, 2002.

[36] R. H. Grenier, «An object-oriented model for interorganizational collaborative planning,» Southeastern, 2000.

[37] Danijel Radošević, Ivan Magdalenić, «Python Implementation of Source Code Generator Based on Dynamic Frames,» de *Proceedings of the 34th International Convention MIPRO*, Croacia, 2011.

[38] Hafidhoh, Nisa'ul and Liem, Inggriani and Azizah, Fazat Nur, «Source code generator for automatic business rule implementation,» de *International Conference on Data and Software Engineering*, Indonesia, 2016.

[39] Dong Hyuk Park and Soo Dong Kim and Park, Dong Hyuk and Kim, Soo Dong, «XML rule based source code generator for UML CASE tool,» de *Proceedings Eighth Asia-Pacific Software Engineering Conference*, China, 2001.

[40] Fayaz Ali Dharejo ; Shaukat Hayat ; Parinya Suwansrikham ; She Kun ; Mutiullah Shaikh,Fayaz Ali Dharejo ; Shaukat Hayat ; Parinya Suwansrikham, «Data transformation of UML diagram by using model driven architecture,» de *IEEE 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, China, 2018.

[41] J. E--khoury, «Lyo code generator: A model-based code generator for the development of OSLC-compliant tool interfaces,» *SoftwareX,* vol. 5, pp. 190-194, 2016.

[42] Fowler, Matthew and van Niekerk, Brahm, «Meta-programing for the Real World,» de *OOPSLA (Object-Oriented Programming, Systems, Languages & Applications) Conference*, USA, 2004.

[43] Carlos Mario Zapata Jaramillo,Gloria Lucía Giraldo Gómez,John Jairo Chaverra Mojica, «Generacion Automatica de Codigp Bajo el Patron MVC a partie de Esquemas Preconceptuales,» de *XIX LACCEI Latin American and Caribbean Conference , Engineering for a Smart Planet, Innovation, Information*, Colombia, 2011.

[44] Sven Jörges ; Tiziana Margaria ; Bernhard Steffen, «NHibernateMapper - A Tool for Rapid Development of Data Access Layer for Interoperable GIS Solutions,» *Electronics,* vol. 15, nº 1, pp. 62-66, 2011.

[45] Colin Atkinsona, Philipp Bostana, Dirk Draheimb, «Foundational MDA Patterns for Service-Oriented Computing,» *In Journal of Object Technology,* vol. 14, nº 1, p. 1–30., 2015.

[46] Hiroshi Suganuma Norikazu Kijima Takeshi Nii Kinya Nakamura, «Preliminary case study on software reuse with object persistency framework,» de *Proceedings 26th Annual International Computer Software and Applications*, UK, 2002.

[47] S. J. Chapman, Java for Engineers and Scientists, NJ, USA: Prentice-Hall, 2003.

[48] Shih, Chihhsiong ; Wu, Chien-Ting ; Lin, Cheng-Yao ; Hsiung, Pao-Ann ; Hsueh, Nien-Lin ; Chang, Chih-Hung ; Koong, Chorng-Shiuh ; Chu, William C., *A Model-Driven Multicore Software Development Environment for Embedded System,* USA, 2009.

[49] Ohtsuki, M. and Yoshida, N. and Makinouchi, A., *A source code generation support system using design pattern documents based on SGML,* Kyushu Japan, 2000.

[50] P. B. Kruchten, «The 4+1 view model of architecture.,» *IEEE Software,* vol. 12, nº 6, pp. 42-50, 1995.

[51] Carmen Penades, Patricio Letelier Torres, «Metodologias agiles para el desarrollo de software.,» *DialNet,* vol. 5, nº 26, 2006.

[52] P. e. al., Systematic Reviews in the Social Sciences, Blackwell Publishing, 2006.

[53] Magdalenić Ivan, Danijel Radošević, and Dragutin Kermek, *Implemetation model of source code generator,* Croatia, 2011.

[54] Havva Çetiner Altiparmak ; Büşra Tokgöz ; Ökkeş Emin Balçiçek ; Aslıhan Özkaya ; Ahmet Arslan, *Source code generation for lage scale application,* Turkey, 2013.

[55] B. Kitchenham, *Guidelines for performing Systematic Literature Reviews in Software Engineering,* Keele: Department of Computer Science, 2007.

[56] Gloria Arcos-Medina, Jorge Menéndez, Javier Vallejo, «Comparative Study of Performance and Productivity of MVC and MVVM design patterns,» de *(Ibero-American Symposium on Computer Programming)*, 2018.

[57] P. H. M. M. Fabiano Freitas, «A Naked Objects based Framework for Developing Android Business Applications,» de *ICEIS 2016 Proceedings of the 18th International Conference on Enterprise Information Systems*, Italia, 2016.

ANEXOS

Anexo 01: Procesos de los *Frameworks*.

Estudio	**Frameworks**	**Procesos/fases**	**Input/output / Otros**
S1	PhoneGap	• Fase Definición de la aplicación. • Fase composición del diagrama de estado. ○ GCF • Fase de integración y despliegue(phonegap)	Diagrama de transición de estados/GCF Aplicación móvil.

S2	Framework MDA/MDD	• GUI Environment, Model, database, Converter Components, Symbolical components, power systems patterns, power systems application.
		• Simulación y modelado de sistemas de energía intuitivos y fáciles de usar.
		  ○ **(Modelo de sistema de la energía.)**
		• Validación de modelos y generación de código.
		  ○ Convertidor del modelo.
		  ○ **Generación de código.**
		• Aplicación de sistemas de energía con módulo de comunicación de datos.
		  ○ Código Autogenerado.
		  ○ Librerías.
		  ○ **Aplicación final.**
			Modelo/ Aplicación.

S4	"Microsoft Foundation Classes (MFC®).
	
	Microsoft NET Framework®.
	
	Java EE® y marcos para (GUI) como Qt, Motif, Swing®, o Adobe Flash®"
		• Modelamiento:
		  ○ Análisis, Construcción, refinamiento del modelo de dominio
		• Diseño:
		  ○ Analizar restricciones y adaptabilidad.
		  ○ Mapa del conocimiento a la arquitectura.
		  ○ Mapear objetos de arquitectura a patrones de diseño.
		  ○ Completo diseño de software detallado.
		  ○ **Generación de código arquitectónico**.
			Modelo ejecutable.

S5	ArchWiSEN Proceso de desarrollo basado en MDA para	• habilitar el diseño de aplicaciones a través de modelos de alto nivel de abstracción; • proporcionar una metodología específica para desarrollar aplicaciones WSAN; y	Modelo. (para aplicaciones SWAN)

	aplicaciones WSAN	• Ofrecer una infraestructura MDA compuesta por PIM, PSM y programas de transformación para respaldar este proceso
		  ○ Crear un perfil UML independiente de la plataforma (Perfil UML SWAN).
		  ○ Crear un perfil UML especifico de la plataforma (Perfil UML tinyOS).
		  ○ Crear programa de transformación M2M (WSAN to TinyOS).
		  ○ Crear programa de transformación M2T (TinyOS to code).
		• **Generación de Código fuente.**	

S6	AndroMDA	• Definición de la aplicación Multiagente. • Modelamiento del diagrama con AUML. • **Generación de código fuente.**	Modelo AUML (agent)/GCF
S7	VERTAF / Multi-Core (VMC) (Basado en uml). VMC es un IDE para una arquitectura de software integrada de varios núcleos	• Describir sus requisitos de sistema con SysML[1]. utilizando este entorno • Modelar el diseño con notación estándar SysML. • Aplicar automáticamente una estructura de patrón en su diseño para un sistema integrado de múltiples núcleos de alta calidad. • **Generar fuente código a través de un modelo bien diseñado**. • asignarse a una arquitectura de hardware diferente según lo asignado por el modelo. • Finalmente probar el código.	Modelos SysML
S8	ProjectIT Architect	• Definir un modelo UML adecuado. • Definir “Plantillas de transformación Model2Model” para producir nuevos modelos. • Definir "Plantillas de transformación Model2Text". • **Finalmente generar el código fuente.**	Modelo UML
S9	Titan Framework	• Definir la aplicación. • composición del modelo de arquitectura basado en PLUS (Product line UML software). • **Generación de Código Fuente.**	Archivos de configuración XML y SQL
S10	Tom, Apigen, vulcan, acai	• Generadores de código a través de: herramientas case. • A través de IDE. • A través de Framework. (herramientas)	Estudio de análisis de herramientas de GCF.

S11		• Ingeniería del Dominio:
		  ○ Requerimientos y análisis del dominio.
		  ○ Modelamiento de la variabilidad del sistema.
		  ○ Arquitectura del sistema.
			Modelo de diseño.

[1] System Modeling Language.

		• Ingeniería de Aplicación:
		  ○ Requerimiento de la aplicación.
		  ○ Configuración del producto.
		  ○ Configuración de la arquitectura.
		  ○ Arquitectura final.	

S12	iPOJO-OSGi Framework.
	
	The FraCC Framework
		• Requerimientos funcionales y no funcionales.
		• Transformación de modelo a modelo.
		  ○ Generación de modelos.
		• Transformación del modelo a código.
		  ○ Generación de código Fuente.
		• Apelación final.
			Requerimientos basados en componentes.

S13	Strut (Capa de vista).
	
	Spring (gestión de app).
	
	Hibernate base de datos (CRUD)
		• Flujo de trabajo JBPM 1:
		  ○ patrón de diseño (Plantillas).
		  ○ Campos meta data.
		  ○ Regla de Negocios.
		  ○ Generación de código fuente.
			modelo de datos.

S14	MDD/MDA	Para la generación de código de formularios para la capa de presentación.	Modelos UML
S15	AndroMDA Framework	Operación de Generación de código. Integridad de un esquema de comportamiento. Ejecución de un esquema de comportamiento.	Diagrama de clase enriquecido. UML Models
S16	J2EE or .NET framework	transformar virtualmente cualquier modelo UML en código de aplicación utilizando un marco J2EE o .NET. Al utilizar el entorno estandarizado basado en XML de la herramienta, los arquitectos de software y los desarrolladores clave utilizan plantillas que sirven como bloques de construcción para transformar su propia arquitectura de software en un activo reutilizable.	UML Models
S17		El modelo fue desarrollado siguiendo el proceso de Rational Unified. Se ajusta a la arquitectura 4 + 1 y se expresa en la notación U M L. El software de modelado visual Racional Rose se utilizó para crear el modelo y generar código fuente prototípico. Durante la fase de elaboración de este proyecto, se desarrollaron artefactos de cuatro vistas arquitectónicas. Solo las vistas de caso lógico, componente y uso son relevantes para la validación del modelo.	UML models

[1] Java Business Process Model.

S18	SCT dynamic frames.	Specification (S) Configuration (C) Templates (T) • Controlador: Prepara los inputs para el generador. • Clases CST: o Especificación: Nombre del atributo, valor del atributo. o Configuración: Conexión, atributo de especificación y plantilla. o Plantilla: Contiene la plantilla base del generador.	Modelo de marcos dinámicos SCT
S19	.Net Framework	• Proceso de Análisis. • El siguiente proceso es asignar la regla de negocio transformada a los elementos del código fuente. • El proceso de generación de código Fuente. o MappedCode (GetTheParameter) o LoadVarDictionary (GetGeneratedRule, GeneratedCode, GetVarType) o GenerateCode	Gramática, diccionario de negocios y plantilla.
S20	GCF basado en la regla XML para Herramienta Case UML.	• Modelo de diseño extractor. • Mapeo del modelo de diseño al código fuente. o descriptor de regla de mapeo. o Interprete de reglas. o Diagrama de clases y diagrama de secuencias para el generador de código. o Generar código fuente.	Modelo UML.(.xml)
S21	Modelo de Implementación de GCF. (IMSCG)	• Plantillas de código de programa • La especificación de la aplicación • Configuración del generador de código fuente. • **Generación de Código Fuente.**	las plantillas, la especificación y la configuración del GCF.
S22	GCF basado en un lenguaje de especificación propietario	• La meta-base o El metamodelo básico o Descripción del lenguaje de programación. • El lenguaje de especificación o Especificaciones y declaraciones o Plantillas de código fuente. o Declaraciones para manejar los metadatos • La generación de aplicaciones o Generación de base de datos o Ejecución de la integridad referencial o **Generación de código fuente** o Generación de interfaz de usuario.	Meta modelo básico (base de datos).

S23	openArchitectureWare framework. oAW Eclipse Modeling Framework EMF marco web Django (en phyton)	(MDE) approach Seleccionar DSL para definir la estructura estática de las aplicaciones de base de datos. Definir plantillas para la GCF. Configuraciones. Generación de código fuente.	UML, MOF.DOMMLite Model.
S24	Lyo OSLC4J SDK	• Configure el proyecto OSLC4J: cree manualmente un proyecto Eclipse vacío. • Adaptador del modelo OSLC4J: instanciación gráfica del meta modelo del adaptador. • Valide el modelo del adaptador. • Generar código de adaptador. • Implementación completa del adaptador. • Ejecutar adaptador.	una instancia del meta modelo OSLC como entrada. produce un código Java compatible con OSLC4J
S25	.Net, J2EE -ASP.Net, Code Behind, Ado.Net. VB. -JSp, Struts, Entity EJBs with CMP, Ant, JDBC RAD	• Especificación de requerimientos. • Construcción de meta programas. • Lógica del negocio. • Despliegue de la aplicación.	DSL (modelos), XSL-T plantillas
S27	Three tier framework: wizard web-based data entry application. Wizard framework. struts, ant, J2EE	View(jsp) Controller (Servlet) Business logic (action) (Collect, navigate and store) Models(beans) Especificaciones: datos y validación. Framework Vista Implementación.	Textual specification in form of XML application

S28	.net platform: Nhibernate, JPA,
	
	top link, dotnorm, entity framework, OBJ.net and propel,
	
	while hibernate can be given as an example for java.
		• Enfoque de ORM estático.
		• Arquitectura multicapa.
		• Diseño Tecnológico Detrás De SCG.
		  ○ XML, XSLT.
		  ○ Entradas y salidas
		  ▪ GCF de la tabla.
		  ▪ GCF de SP.
		  ▪ GCF de la vista.
		  ▪ GCF del escenario de negocios.
			Definición de la tabla: XML, Store procedure definition XSLT documentation.

S30	Eclipse Frame work	• Análisis del dominio del problema y selección de artefactos a generar. • Identificación de patrones dentro de la implementación de referencia. • Características de la herramienta y creación de la arquitectura de la solución. • Diseño e implementación del Lenguaje de Dominio Específico. • **Plantillas de generación de código automático.** • Pruebas sobre el DSL construido.	Domain Specific Language (DSL),
S31	Marco CreaCod.	• Seleccionar y configurar la base de datos a ser conectada. • Seleccionar el lenguaje de programación en el que se creará el código fuente. • Llenar los atributos de cada una de las tablas de la base de datos. • Llenar los atributos de cada uno de los campos de las tablas. • Seleccionar y llenar los atributos de la plantilla de administración GUI del proyecto. • Generar el código fuente resultante.	Base de Datos.
S29	NHibernateM apper. Microsoft .NET Framework	• Aplicación • programming classes, database and mapping file. • Definir tres consultas adicionales que recuperen la información del esquema de la base de datos. Estas consultas (GetTables, GetColumns y GetConstraints). • Proveedor de datos (en forma de DLL) ofrecido por el producto DBMS. • Cambie el archivo de configuración para que incluya la declaración del proveedor de datos agregado.	Base de Datos. Dominio especifico. En forma de clases de programa. De aplicación DDL nhibernate-mapping.xsd

		Genera código Fuente par a la capa de acceso a datos.	
S33	Framework (OPRF)	• Archivo de definición del diseño de la tabla. • DB generador de código fuente de acceso • Generación de código fuente de acceso a tablas. • Componente de acceso a tabla. • Componente de ejecución de SQL. • Componente de conexiona base de datos.	Archivo de definición de diseño de la tabla. (modelo de datos). Lógica del negocio. Archivo de configuración.

Tabla 38 Frameworks.

Anexo 02: Descripción de los procesos y métodos utilizados en la GCF.

(S1) [24] PhoneGap es un marco de desarrollo de código abierto y basado en estándares para crear aplicaciones móviles multiplataforma con tecnologías HTML5, CSS3 y JavaScript. Es compatible con varias plataformas móviles, incluyendo iPhone / iPad, Google Android, Symbian, BlackBerry y Windows Mobile. PhoneGap concreta la vista del patrón MVVM. La Vista Modelo es implementado por una Máquina de Estado que controla y maneja los estados (es decir, el Modelo) de la aplicación móvil. La comunicación entre los estados se implementa mediante la adopción de una técnica de paso de mensajes. El proceso de desarrollo se compone de tres fases: definición de la aplicación, composición del diagrama de estado e integración y despliegue.

En el estudio [49] se plantea un sistema de soporte de generación de código fuente para apoyar el diseño y ayudar a comprender los códigos fuente. Su objetivo es utilizar patrones de diseño como componentes de manera eficiente mediante la automatización para integrar patrones de diseño en el diseño de una aplicación tanto como sea posible. También tiene como objetivo permitir a los usuarios comprender los significados de las estructuras en los códigos fuente al relacionarlos con los patrones de diseño en los que se basan. El sistema de soporte de generación de código fuente debe generar todos los códigos fuente que pueden generarse automáticamente. La **Ilustración 15** muestra el Proceso de generación de código Fuente.

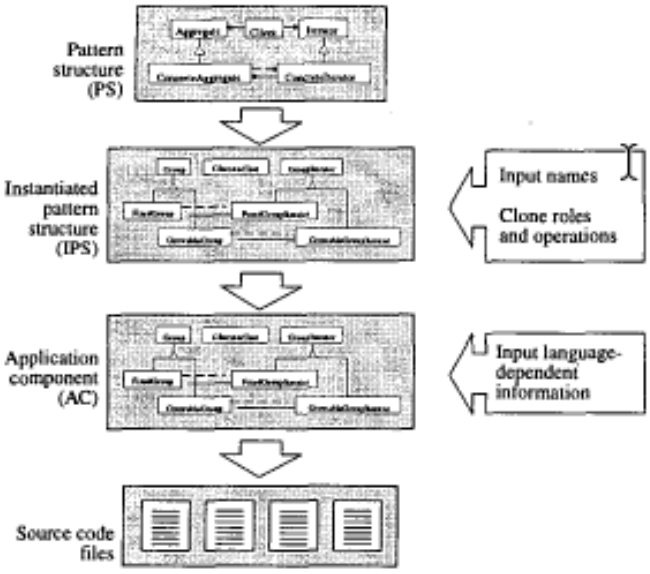

Ilustración 15 - Proceso de generación de Código Fuente.

En (S5) [27] Se genera código fuente para una plataforma WSAN[1], utilizando el marco ArchWiSeN.

ArchWiSeN generara código para aplicaciones en muchos dominios diferentes, el código representa la lógica, la configuración y el enrutamiento Si se cumplen todos los requisitos de la aplicación, la infraestructura MDA (ArchWiSeN) realizará la actividad "Aplicar transformación M2M". Dicha actividad toma como entrada la instancia de PIM (modelo UML) para generar como salida una instancia de PSM que representa la realización de la aplicación en la plataforma específica elegida por el experto en redes. Finalmente, se realiza la actividad "Aplicar transformación M2T". Esta actividad tiene dos entradas: (1) el modelo PSM refinado por el experto en redes y (2) las plantillas de código de plataforma elegidas. Genera, como salida, el código fuente de la aplicación que se implementará en los nodos del sensor. El código generado es luego refinado por ambos desarrolladores (cada uno con respecto a su conocimiento específico) en la actividad "Refinar Código" para agregar mejoras tales como como funciones

[1] Wireless sensor and actuator networks

específicas de la aplicación o parámetros de protocolo no generados automáticamente por la transformación MDA.

En el estudio (S6) [28], se puede apreciar el proceso de generación de código a través del framework MDA que consiste en:

abrir el archivo ".xml" en la herramienta MagicDraw, que se ocupa de la parte del modelado para toda la aplicación; tomando en cuenta que AndroMDA no puede leer los modelos de MagicDraw directamente; Se exporta a otro formato de archivo en este caso EMFUML. Este el archivo que será procesado por AndroMDA. Siguiendo la definición del modelo, la generación del código de la aplicación se logra ejecutando un comando. Por lo tanto, todas las clases correspondientes al modelo, el sistema colaborativo se crea y el desarrollador puede acceder a ellas y modificarlas fácilmente, donde tiene la capacidad de implementar sus operaciones en el código generado.

Según (S9) [18] Titan Framework Mobile genera código para una aplicación móvil, para esto, este generador escanea los archivos en lenguaje de marcado (XML) desde la aplicación web (Servidor), responsable de la parametrización de los componentes de Titan Framework, generando un código Java que compone el proyecto de la aplicación móvil. En particular, un bus de servicio que contempla la comunicación entre las aplicaciones web y las aplicaciones móviles correspondientes también fue implementado con titán framework.

(S12) Proceso de desarrollo propuesto para aplicaciones CBSE[1], basado en MDA para diseñar y validar aplicaciones, y en marcos para proporcionar el soporte requerido en tiempo de ejecución; el estudio realiza dos casos de estudio donde utiliza dos frameworks, el primero FracCC y segundo IPOJO.

En (S13) [15] se aplica frameworks de código abierto struts, spring, hibernate como la implementación liviana de la arquitectura J2EE. Struts se utiliza para la capa de vista web. Spring es responsable de la configuración y gestión de toda

[1] Component-based software engineering.

la aplicación. Hibernate hace el trabajo de mapeo de base de datos ORM. Este diseño en capas ayuda a desacoplar los módulos de software del sistema que dividen el desarrollo del software. Y es ventajoso para el diseño del generador de código para mejorar la eficiencia de generación de código. El generador de código solo necesita generar el código fuente de las capas J2EE de acuerdo con la característica de las capas.

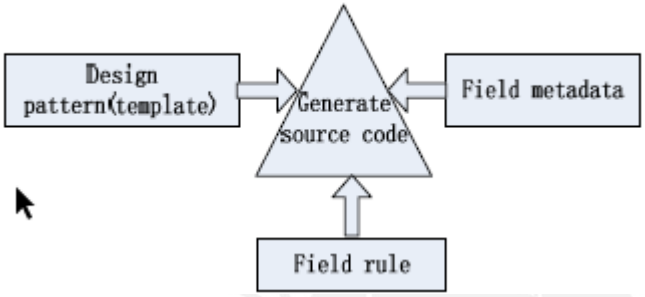

Ilustración 16 Modelo de generador de código.

Se utiliza POJO como modelo de dominio que mantiene un buen estilo de diseño orientado a objetos facilita las pruebas y la reutilización.

En la capa de acceso a datos, se introduce el modo de acceso a objetos de datos DAO, que deja la lógica de negocios y la lógica de acceso a datos. En la ayuda de Hibernate ORM framework, las operaciones CRUD a los datos de la base de datos están empaquetadas. En esta capa, se diseñan la interfaz de acceso a la base de datos denominada DAO del modelo de dominio y la clase de implementación de Hibernate.

En el Estudio (S15) [34] se encontró el framework AndroMDA, un marco de generación de código fuente abierto. Los modelos de herramientas UML se transforman en componentes desplegables de plataformas como J2EE, Spring o .NET. AndroMDA genera métodos en blanco para implementar la lógica empresarial de la aplicación, pero no ayuda en la generación automática del cuerpo de esos métodos.

En S18 [37] se presenta los marcos SCT, este marco define tres componentes: Especificación, Configuración y un conjunto de Plantillas, se utiliza Python para

la implementación del generador estos tres componentes se presentan en forma de listas de Python. Estas listas son estructuras de datos muy flexibles.

Además, se hace uso de un controlador cuya función es preparar entradas para el generador (objeto SCT) y recopilar y guardar las salidas del generador. El controlador define un nuevo objeto SCT y lo inicializa. También encuentra cada nombre de archivo de salida en particular y la parte apropiada de la Especificación, así como la plantilla base apropiada de la Configuración. El controlador invoca el generador y guarda el código generado en el archivo de salida de destino.

(S19) [38] Se desarrolla un generador de reglas de negocios para generar códigos fuente basados en un script DSL para expresar las reglas de negocios. Se muestra que el generador puede ayudar a la implementación de las reglas comerciales en los códigos fuente y que el código generado puede usarse en aplicaciones empresariales, utiliza un Framework MVC. El estudio también muestra un prototipo del generador de reglas de negocios, basado en el desarrollo de una interfaz de usuario para ayudar a un usuario a administrar las reglas de negocios llamado Business Rule Generator (BRuGen). El usuario puede administrar (para agregar, editar, eliminar o mostrar) las reglas comerciales mediante la interfaz de usuario. BRuGen se creó sobre .NET Framework y el uso de la biblioteca ANTLR.

S20 El proceso de generación de código utiliza un árbol de generación y un mapa generador primitivo. El proceso de generación consta de tres partes:

Primero, el generador de nodos se obtiene del árbol de generación. Y luego el generador de nodos obtiene un generador primitivo apropiado del mapa del generador primitivo. Y luego el generador de nodos colabora con el generador primitivo para generar código con un contenedor de datos modelo.

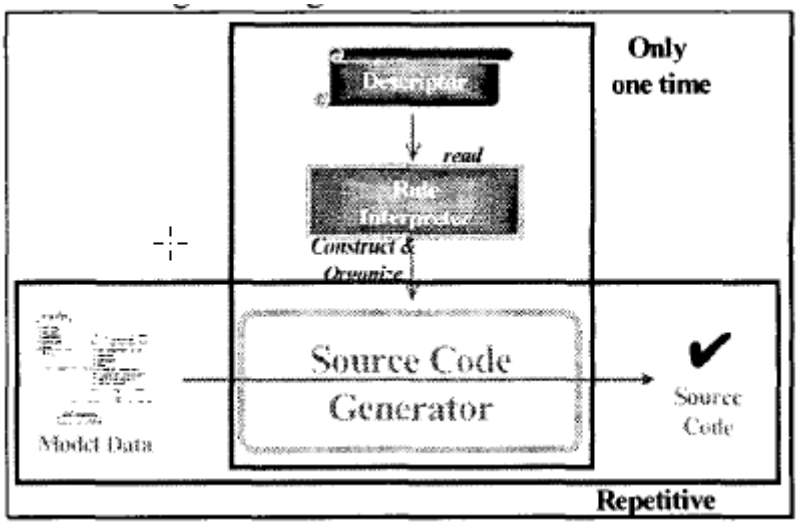

Ilustración 17 Proceso completo para la generación de código fuente.

En el estudio (S21) [9] el proceso de generación de código la especificación de la aplicación y la configuración del generador de código fuente definen el proceso de generación de código fuente al elegir la plantilla de código de programa adecuada y al proporcionar valores concretos para reemplazar marcas en las plantillas de código de programa. La configuración del generador de código fuente define qué tiene que ver el generador de código fuente con la sustitución de marcas en las plantillas de código de programa. Las entradas al generador de código fuente son las plantillas de código de programa, la especificación de la aplicación y la configuración del generador de código fuente. La salida del generador de código fuente es una representación de cadena del código del programa.

En (S23) [6] se presenta DOMMLite, un lenguaje extensible específico del dominio (DSL) para la definición de estructura estática de aplicaciones orientadas a bases de datos. Se ha utilizado el enfoque de ingeniería dirigida por modelos (MDE). La estructura del lenguaje se define por medio de un metamodelo complementado por reglas de validación basadas en el lenguaje de verificación y las extensiones basadas en el lenguaje extendido, que forman parte del marco de openArchitectureWare. El metamodelo se ha definido junto con la sintaxis textual, que permite la creación, actualización y persistencia de modelos DOMMLite usando un editor de texto común. La semántica de ejecución

DSL se ha definido mediante la especificación y la implementación del generador de código fuente para una plataforma de destino con una semántica de ejecución ya definida. Para habilitar la edición de modelos, también se ha desarrollado un editor de texto de Eclipse. DSL, definido de esta manera, tiene la capacidad de generar código fuente completo para formularios GUI con CRUDS (Crear-Leer-Actualizar-Eliminar-Búsqueda) y operaciones de navegación.

En (S24) [41] se detalla el proceso de desarrollo de un adaptador OSLC4J [1] utilizando el generador de código Lyo. Primero, Crear de forma manual un proyecto Eclipse vacío que esté configurado para desarrollar cualquier adaptador OSLC4J. luego especificar la funcionalidad deseada del adaptador mediante una instanciación gráfica del meta modelo del adaptador. Utilizar el mecanismo de validación proporcionado para garantizar que todas las propiedades del modelo requeridas se definan como se espera. Generar código para producir casi todo el código Java necesario para un servidor de adaptador OSLC4J en ejecución.

En (S25) [42] En la interfaz de usuario, se produce HTML, utilizando ASP.NET en IIS para .NET, y utilizando JSP en la arquitectura J2EE. La retención de datos y la navegación se controlan mediante clases de código subyacente en .NET y mediante Struts en J2EE. La interfaz de servicio se implementa en .NET y por EJB, con estructuras de datos lógicos adjuntas que en ambos entornos son implementadas por clases normales. La persistencia en .NET está 'escrita a mano; en J2EE utiliza la persistencia gestionada por contenedor EJB2.

El estudio (S27) [11] está basado en la arquitectura MVC, La especificación de la aplicación se realiza a través de un documento wiz. xml. Que describe: todos los datos de entrada con su rango de valores y validación. El framework comprende las acciones: recoger, Navegar y almacenar. El servlet lee el archivo wiz.xml y construye una presentación interna de la especificación. Se genera

[1] Open Services for Lifecycle Collaboration for Java.

desde wiz.xml el modelo en forma de java bean y para cada elemento de vista se genera un jsp-Page.

El generador de código genera una página JSP por cada etiqueta de vista en el archivo wiz.xml. estas páginas contienen un elemento forma con sub elementos para cada archivo. El framework wizard es implementado completamente en java utiliza java servlets para el controlador y java server Pagés para las vistas. El corazón de la arquitectura MVC utiliza el framework strut de la fundación de software apache. El entorno de la generación de código es implementado utilizando la herramienta ANT.

(S28) [12] En este estudio, se ha propuesto un sistema de generación de código fuente (SCG) que proporciona una codificación más rápida y estandarizada para sistemas de software a gran escala. En el modelo propuesto, el diseño de pantalla que incluye escenarios comerciales y objetos de base de datos, como tablas y procedimientos almacenados, que son los objetos utilizados con mayor frecuencia en las instituciones de software, se utilizará como entradas. La **Ilustración 18** muestra el proceso de generación de código fuente.

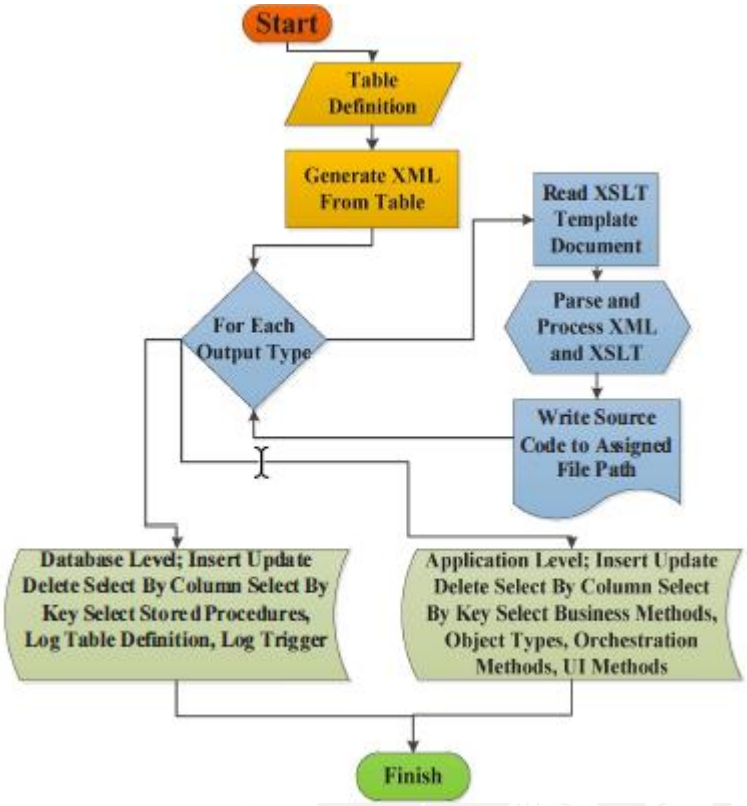

Ilustración 18 Proceso de GCF del estudio nro. 28

En el estudio [44] se presenta *Genesys* un *framework* orientado a servicios para la construcción de alto nivel de generadores de códigos. Se describe cómo este enfoque orientado al servicio conduce a generadores de código robustos y de alta calidad al confiar en una gran biblioteca de bloques de construcción altamente reutilizables. Se comparan dos clases de generadores de código, los Extrusores y los Generadores Puros, que ocurren dentro *Genesys* y objetivos de diferentes tipos de motores de ejecución.

Dado que los generadores de código se construyen dentro del jABC [1](marco para el modelado de procesos), *Genesys* gestiona la construcción de generadores de código para los modelos de jABC. Hasta ahora, *Genesys* se ha utilizado para desarrollar generadores de código para una variedad de plataformas de destino diferentes, como una serie de plataformas basadas en Java, dispositivos móviles, motores BPEL.

[1] Java Application Building Center.

El estudio (S31) [4] trata de un generador de código fuente para controlar la base de datos MySQL, SQL Server y Access, usando los lenguajes de programación ASP, JSP, y PHP que se encuentren previamente desarrollados en plantillas de código fuente, las cuales puedan ser intercambiables entre sí. Los resultados muestran su funcionalidad, dando al programador la posibilidad de gestionar información en pantallas amigables sin necesidad de conocer los detalles de conexión de base de datos y lenguaje de programación.

En (S32), la herramienta es capaz de atravesar automáticamente el esquema de la base de datos y la biblioteca de clases de dominio proporcionados para coincidir y sugerir correspondencias. La GUI interactiva ofrece la capacidad de aceptar o rechazar correspondencias sugeridas, así como de definir manualmente las asignaciones para podarlas en una forma final. *NHibernateMapper* se convierte en una herramienta que permite el rápido desarrollo de la capa de acceso a datos para soluciones GIS interoperables. *NHibernateMapper* se ha implementado completamente en C # con Microsoft .NET Framework. Como resultado, se crea la biblioteca de clases NHibernateMapper.dll. Todas las clases de programación en la biblioteca podrían dividirse en dos grupos principales: clases para recuperación de esquema de base de datos y clases para la manipulación de archivos de mapeo.

(S33) [46] El OPRF consta de (1) el componente de ejecución de SQL, que proporciona la conexión de la base de datos y la conversión de datos, (2) el componente de conexión de la base de datos, que proporciona un mecanismo para adjuntar y desconectar a / desde un servidor de aplicaciones web sin cambios en el código fuente, y (3) generador de código fuente de acceso a la tabla, que genera los componentes de acceso a la tabla. Se aplica la técnica a proyectos de desarrollo de aplicaciones web reales.

Anexo 03 Otros gráficos de la selección inicial de estudios.

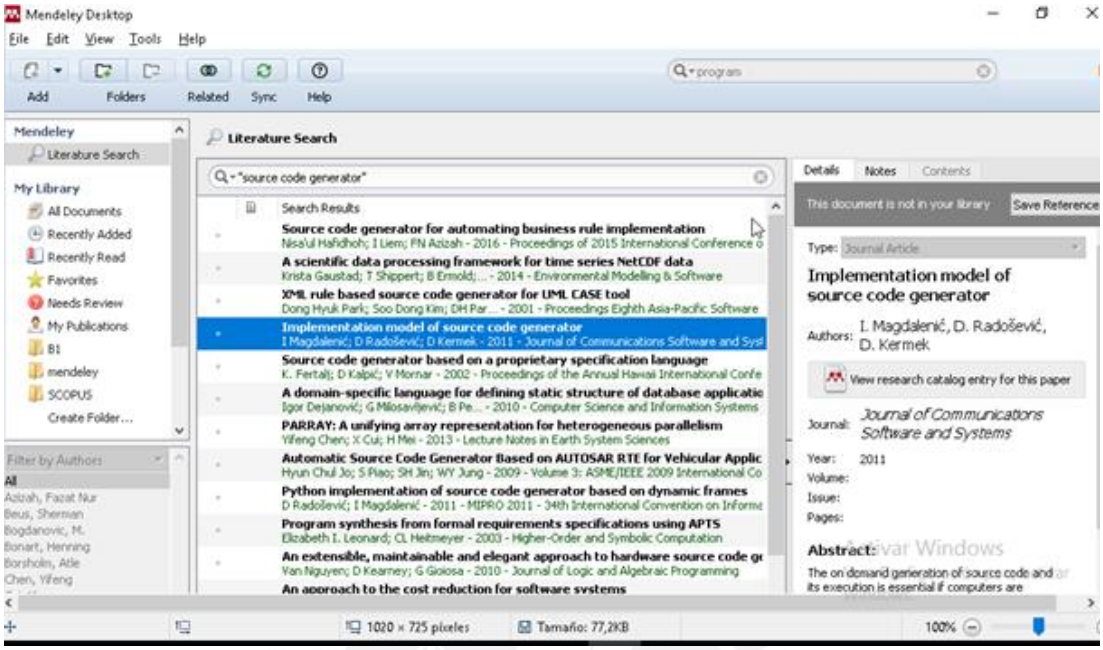

Ilustración 19 Selección inicial con Mendeley.

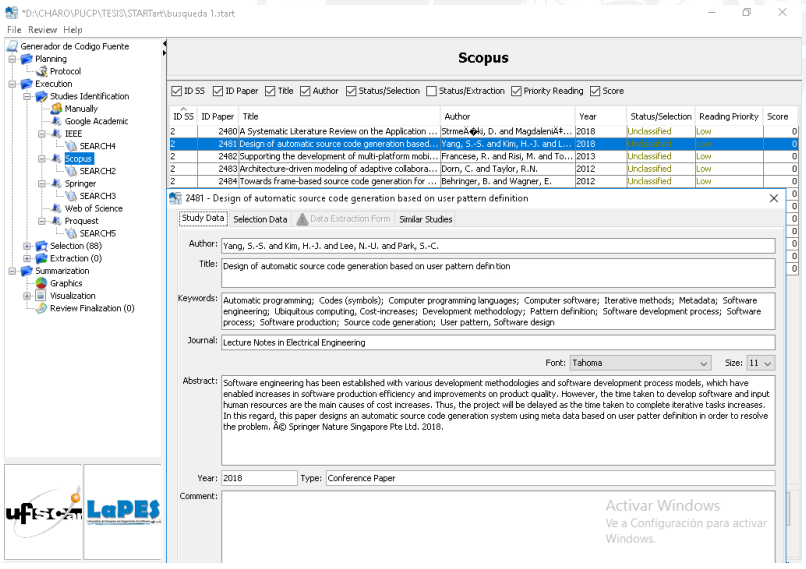

Ilustración 20 Selección inicial de estudios con StArt.

("source code generator" or "automatic generation of source code" or "automatic source code generation") and ((design or architecture) and patterns) and databases and tools and frameworks and methodologies and languages Springer 06/05/2018

("source code generat" or "automatic generation of source code" or "automatic source code generat") and ((design or architecture) and patterns) and databases and methodologies and tools and frameworks and languages ("Source code generat*" or "automatic generation of source code" or "automatic source code generat*") and ((design or architect*) and patterns) AND CRUD ACM 30/05/2018
"source code generator" AND "design patterns" ACM 30/05/2018
("source code generat*" OR "automatic generation of source code" OR "automatic source code generat*") AND ((design OR architect*) AND patterns) AND method* AND languages IEEE
("Source code generat*" or "automatic generation of source code" or "automatic source code generat*") and ((design or architect*) and patterns) and (databases or frameworks or method* or tools or languages)
("Source code generat*" or "automatic generation of source code" or "automatic source code generat*" or SCG or "automatic programming") and ((design or architect*) and patterns) al 22/03/2019
("Source code generat*" or "automatic generation of source code" or "automatic source code generat*" or SCG or "automatic programming") and (architect* or "architect* pattern") and (web or "web application") and (frameworks or databases or languages or IDE or tool) al 22/03/2019.

Tabla 39 Cadenas de búsqueda para la selección de estudios.

Anexo 04: Patrones de Arquitectura Identificados.

En (S1) [24] el código generado usa el patrón arquitectónico MVVM que comprende los siguientes elementos:• Modelo: se refiere a un modelo de dominio que representa el contenido del estado y / o los datos.• Vista: hace referencia a todos los elementos mostrados por los widgets de la interfaz gráfica de usuario, como botones y etiquetas.• Vista Modelo: es una abstracción de la vista que sirve para mediar entre la vista y el modelo. Actúa como un convertidor que convierte la información del modelo en información de vista y pasa los comandos de la vista al modelo. El uso de un enlace de datos declarativo simplifica el desarrollo de la capa de vista a través de la implementación solo de la capa Vista Modelo. La **Ilustración 21** describe este patrón de arquitectura.

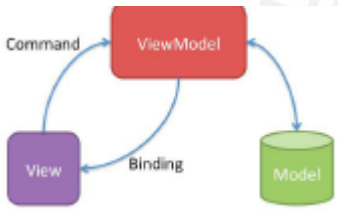

Ilustración 21 El patrón Arquitectónico MVVM [24]

Los Estudios S19, S26, S27 presentan el patrón de Arquitectura MVC (Modelo Vista Controlador); generalmente para separar las reglas del negocio de la funcionalidad de la aplicación; además el código fuente que se genera presenta una arquitectura en capas: Capa de interfaz de usuario, Capa de lógica del negocio y Capa de datos (S26) [43] .La ilustración 22 muestra el funcionamiento de un patrón de arquitectura MVC.

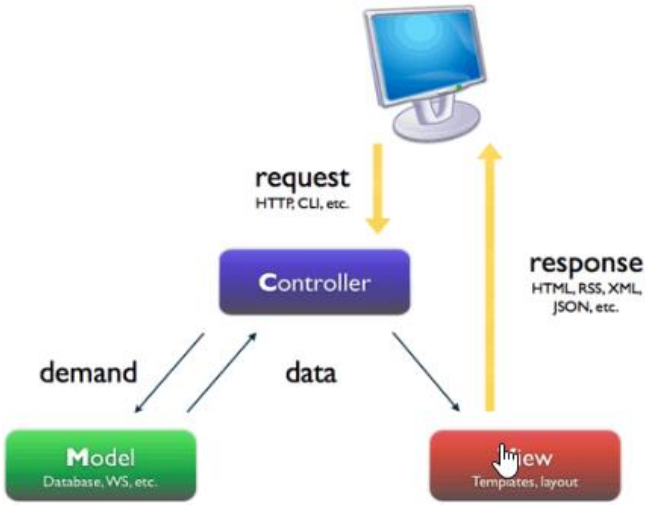

Ilustración 22 Patrón de Arquitectura MVC.

Arquitectura dirigida por modelos propuesta por la OMG. En [17] se menciona que la abstracción es la característica más sólida del desarrollo basado en modelos, también conocida como *Model-Driven Architecture* (MDA). Divide al sistema en distintas perspectivas que muestran su estructura, comportamiento dinámico, interfaces y estados internos, los diseñadores pueden desarrollar un sistema complejo que mantenga la coherencia y la corrección durante todo el ciclo de vida del desarrollo del producto.

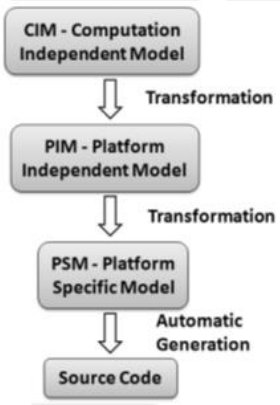

Ilustración 23 Arquitectura dirigida por modelos MDA

En [15] se habla de J2EE, donde se puede implementar una arquitectura en capas de tres niveles para aplicaciones Web. La capa de vista frontal se usa para mostrar la interfaz de usuario, capturar los datos de entrada y procesar la

solicitud del usuario. La capa de servicios del negocio es responsable de que la capa de vista llame procese la lógica de negocios. La capa de acceso a datos se utiliza para realizar operaciones con la base de datos incluida la creación, recuperación, actualización y eliminación. Otros autores consideran cuatro capas, dividiendo la capa de vista frontal en dos partes, obteniendo la siguiente estructura: la capa cliente, la capa web, la capa negocio y la capa datos como se muestra en la figura.

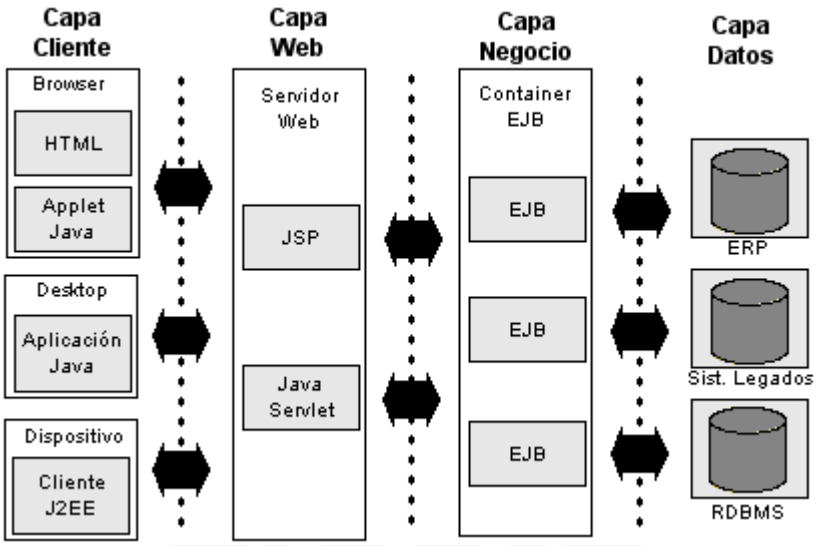

Ilustración 24 Arquitectura J2EE de un sistema de aplicación WEB.

El estudio [36], describe el modelo de vistas de arquitectura 4+1(no es un patrón de arquitectura). Para implementar una arquitectura de software se manejan muchos detalles que son usados por varios actores de la organización; para no sobrecargar el diagrama de arquitectura con demasiados detalles surge la necesidad de usar vistas arquitectónicas, Según *Kruchten* [50] estas vistas son:

- La vista lógica: que representa los requerimientos funcionales del sistema, y es utilizada por el usuario final.
- La vista de procesos: describe los mecanismos de concurrencia y sincronización usados en el sistema; es utilizada por los integradores del sistema para el análisis del rendimiento y la escalabilidad del sistema.

- La vista de desarrollo: muestra el diseño del código en el ambiente de desarrollo; esta vista lo utilizan los líderes del proyecto y programadores, y ayuda en la planeación y evaluación del progreso del proyecto.
- La vista física: describe el mapeo del software en el hardware y refleja los aspectos de la distribución. Los ingenieros de sistemas desarrollan esta vista para determinar la topología del sistema y los requerimientos de comunicación entre distintos componentes.
- El término +1, representa a los escenarios que capturan las características generales del sistema, en esta vista se ilustran las distintas decisiones que son tomadas a lo largo de las cuatro vistas.

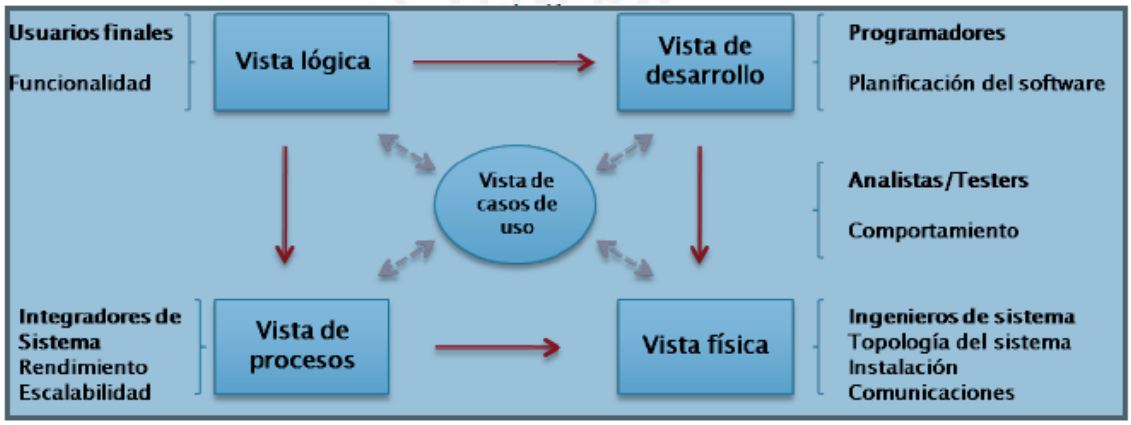

Ilustración 25 Modelo de Vistas de Arquitectura 4+1

Arquitectura de tres capas que puede ser desplegada en un framework J2EE: Capa de lógica del negocio, suele constar de dos partes virtuales: La capa de acceso a datos (donde se realiza la comunicación con la capa de datos) y la capa lógica empresarial (donde se realiza la comunicación entre la capa de IU y la capa de acceso a datos) y la Capa de datos se expresa como el medio de almacenamiento donde se encuentran los datos.

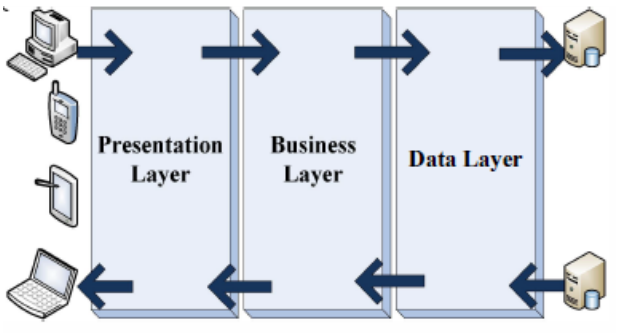

Ilustración 26 Vista de Arquitectura de 3 niveles. [12]

En [44] tenemos la arquitectura *Genesys* está organizado en tres unidades: La Biblioteca, el JABC[1] *plugin* (Java Application Building Center or short *jABC*) y las herramientas de desarrollo tal como se muestra en la **Ilustración 27**. La biblioteca es la parte principal de esta Arquitectura, contiene todos los generadores de código disponibles tanto modelos SLG (Service Logic Graphs) como clases compiladas java; también contiene un conjunto de SIB (Service Independent Building Block) especializados que se pueden usar directamente para generar generadores de código. Las herramientas de desarrollo de Genesys ayudan a los desarrolladores de generadores de código. Estas herramientas contienen un inspector para proporcionar metadatos para un generador de código como (autor, versión, icono, etc.), un editor para modificar las plantillas. y un conjunto de referencia para evaluar y comparar le rendimiento de los generadores de código.

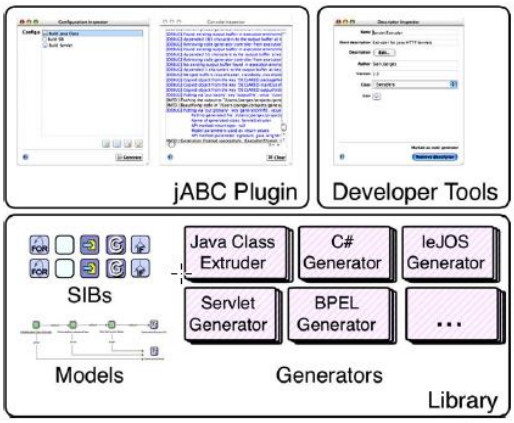

Ilustración 27 Arquitectura Genesys [44]

El Estudio S9 [18] presenta una Arquitectura para dispositivos móviles Titán, que básicamente está formada por el Núcleo y el repositorio de componentes. El núcleo es el responsable de recibir como entrada los archivos de configuración de la instancia (XML y SQL) y generar una aplicación en tiempo de ejecución; además se cuenta con el bus de servicio (*Service bus*) donde los componentes

[1] jabc.cs.tu-dortmund.de

del repositorio se adaptaron para interactuar con la nueva capa de comunicación (función de comunicación).

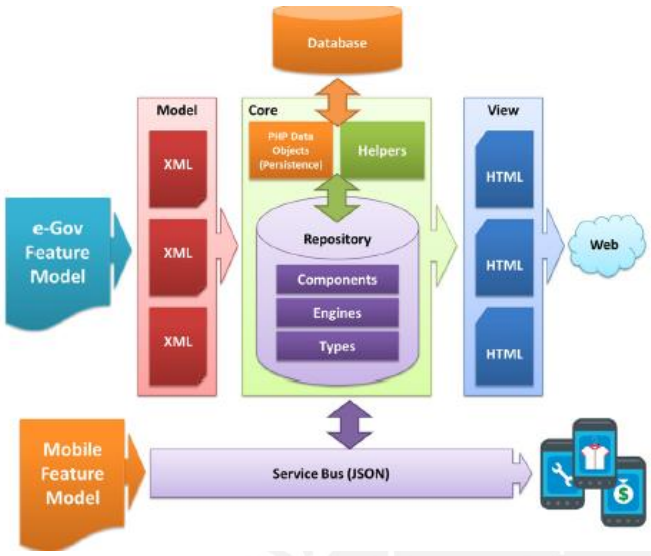

Ilustración 28 Arquitectura Titán para desarrollo de aplicaciones móviles. [18]

En el estudio S32 [45] Se muestra una Arquitectura *NHibernateMapper*, el punto de partida en el proceso de generación de archivos de mapeo *NHibernateMapper* es la base de datos relacional existente, así como el conocimiento de dominio específico en forma de clases de programación de aplicaciones (*dll*). El tercer elemento de entrada es un archivo de mapeo nhibernate-mapping.xsd. *NHibernateMapper* extraerá automáticamente el esquema de la base de datos de la base de datos previamente seleccionada y los atributos de los miembros de clase de la DLL seleccionada. Cuando el usuario finalmente está satisfecho con la definición de asignación, se puede guardar en un archivo de asignación XML de *NHibernate.*

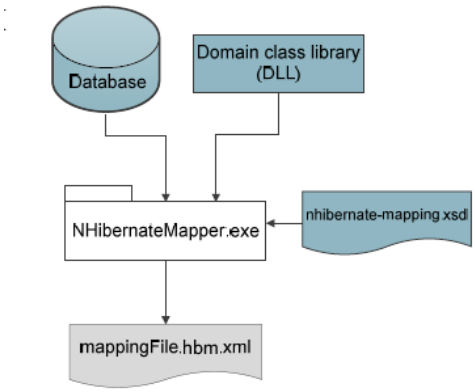

Ilustración 29 Arquitectura NHibernateMapper [45]

El estudio [26], promueve el uso de patrones de diseño mediante el apoyo a la creación de aplicaciones a partir de patrones de diseño (PIML). Se diseña un sistema de soporte de generación de código fuente a partir de patrones de diseño.

El estudio [46], aplica la arquitectura de marco de reutilización de persistencia de objetos OPRF[1] con componentes y un generador de código.

El componente de ejecución de SQL: permite la conexión con la base de datos. El componente de conexión a la base de datos: permite probar la capa de acceso a la base de datos sin cambiar el código fuente para las restricciones del servidor de aplicaciones web. El GCF accede a una tabla de la base de datos, aplica la lógica de negocios y genera la ejecución y conexión de los componentes.

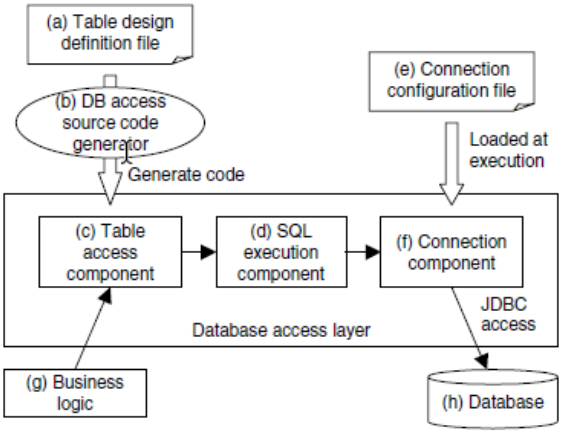

Ilustración 30 Arquitectura OPRF [46]

[1] Object Persistency Reuse Framework.

El estudio S24 que aplica un patrón de Arquitectura REST.

El estudio [35] correspondiente a la Arquitectura *Codagen* es compatible con MDA y utiliza los marcos J2EE y .Net.

openArchitectureWare (oAW) es un conjunto de herramientas y componentes que ayudan al desarrollo de software basado en modelos, basado en MDA, implementado en Java que admite formatos de importación (modelo), metamodelos y formatos de salida (código). Las herramientas de apoyo (como editores y navegadores de modelos) se basan en la plataforma Eclipse. openArchitectureWare es una herramienta para crear herramientas MDA; se utiliza en todo tipo de dominios, los ejemplos incluyen aplicaciones web basadas en J2EE, aplicaciones basadas en Spring, aplicaciones basadas en Eclipse en varios dominios, sistemas en tiempo real basados en C / C, sistemas basados en C ++, Java, Python, Aplicaciones .NET, arquitectura SOA Enterprise, etc. [6].

Anexo 05 Puntaje de Calidad de los Estudios:

La *Tabla 40* muestra el puntaje disgregado de cada estudio (En la **Tabla 11** se define la escala de valores).

Ítem	Pregunta	Puntaje de Calidad (Respuesta: Si= 1, No=0, Parcialmente=0.5)																
		S 1	**S 2**	**S 3**	**S 4**	**S 5**	**S 6**	**S 7**	**S 8**	**S 9**	**S 10**	**S 11**	**S 12**	**S 13**	**S 14**	**S 15**	**S 16**	**S 17**
1	¿El diseño de la investigación es adecuado para llevar a cabo el estudio? [23] [13] [12]	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.5	1	1
2	¿Son creíbles los hallazgos [23]	1	1	1	1	1	1	0.5	1	1	1	0.5	0.5	1	0.5	1	1	1
3	¿Existe relación entre los datos, la interpretación y las conclusiones?	1	0.5	0	0.5	1	1	1	0.5	1	1	0.5	0.5	1	1	1	1	1
4	¿Son claras las suposiciones / perspectivas teóricas / valores que han configurado la forma y el resultado de la evaluación? [13] [23]	1	1	0.5	0	1	0.5	0.5	1	0.5	1	0.5	1	1	1	0.5	1	0
5	¿Las preguntas de investigación están claramente establecidas para los estudios?	1	0.5	1	0.5	1	1	0.5	0.5	1	1	1	0.5	1	0.5	1	1	1
6	**TOTAL**	**5**	**4**	**3.5**	**3**	**4.5**	**4**	**3.5**	**4**	**4.5**	**5**	**3.5**	**3.5**	**5**	**4**	**4**	**5**	**4**

Tabla 40 Puntaje de calidad de los estudios seleccionados (i).

Ítem	Pregunta	Puntaje de Calidad (Respuesta: Si= 1, No=0, Parcialmente=0.5)															
		S 18	**S 19**	**S 20**	**S 21**	**S 22**	**S 23**	**S 24**	**S 25**	**S 26**	**S 27**	**S 28**	**S 29**	**S 30**	**S 31**	**S 32**	**S 33**	
1	¿El diseño de la investigación es adecuado para llevar a cabo el estudio? [23] [13] [12]	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2	¿Son creíbles los hallazgos [23]	0.5	1	1	1	1	1	1	1	1	1	1	1	1	1	0.5	1	
3	¿Existe relación entre los datos, la interpretación y las conclusiones?	0.5	1	0	0.5	1	1	1	1	1	0.5	1	1	0.5	0.5	1	0.5	
4	¿Son claras las suposiciones / perspectivas teóricas / valores que han configurado la forma y el resultado de la evaluación? [13] [23]	0.5	1	1	1	1	1	1	1	0.5	0.5	1	0.5	0.5	0.5	0.5	0.5	
5	¿Las preguntas de investigación están claramente establecidas para los estudios?	1	4	1	1	1	0	1	1	1	1	1	0.5	0.5	1	1	1	
6	**TOTAL**	**3.5**	5	4	**4.5**	**5**	4	**5**	**5**	**4.5**	**4**	5	**4**	**3.5**	4	4	**4**	

Tabla 41 Puntaje de calidad de los estudios seleccionados (ii).