CIS1610AP01

Anchurus-GEN: Generador de código PHP a partir de modelos ISML

Frank Sebastián Franco Hernández

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
CARRERA DE INGENIERIA DE SISTEMAS
BOGOTÁ, D.C.
2016

CIS1610AP01

Anchurus-GEN: Generador de código PHP a partir de modelos ISML

Autor:

Frank Sebastián Franco Hernández

MEMORIA DEL TRABAJO DE GRADO REALIZADO PARA CUMPLIR UNO DE LOS REQUISITOS PARA OPTAR AL TITULO DE INGENIERO DE SISTEMAS

Director

Jaime Andrés Pavlich Mariscal

Jurados del Trabajo de Grado

Luis Guillermo Torres Ribero

Lina María Consuelo Franky de Toro

Página web del Trabajo de Grado

http://pegasus.javeriana.edu.co/~CIS1610AP01/index.html

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
CARRERA DE INGENIERIA DE SISTEMAS
BOGOTÁ, D.C.
Junio 2016

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
CARRERA DE INGENIERIA DE SISTEMAS

Rector Magnífico

Jorge Humberto Peláez Piedrahita, S.J.

Decano Facultad de Ingeniería

Ingeniero Jorge Luis Sánchez Téllez

Director de la Carrera de Ingeniería de Sistemas

Ingeniera Mariela Josefina Curiel Hurtado

Director Departamento de Ingeniería de Sistemas

Ingeniero Efraín Ortiz Pabón

Artículo 23 de la Resolución No. 1 de Junio de 1946

"La Universidad no se hace responsable de los conceptos emitidos por sus alumnos en sus proyectos de grado. Sólo velará porque no se publique nada contrario al dogma y la moral católica y porque no contengan ataques o polémicas puramente personales. Antes bien, que se vean en ellos el anhelo de buscar la verdad y la Justicia"

AGRADECIMIENTOS

Antes que nadie, agradecer a Dios por darme licencia de llegar hasta este punto de la carrera, y guiarme en la recta senda siempre. También agradecer a mi madre que siempre ha estado ahí con su apoyo, comprensión y cariño y de soporte en los momentos difíciles.

Agradezco a la Casa Editorial El Tiempo, porque gracias a la beca que me concedieron por el Concurso de Ortografía, pude estudiar la gran mayoría de la carrera en esta universidad. A Diana Fernández, directora de Promoción Institucional, por su amabilidad en el recibimiento en mi primer día en la Javeriana y por su buena disposición cuando vine a buscar su apoyo. A Fernando Lozano, director del Programa Contacto, por darme las adecuadas orientaciones que me dirigieron a elegir esta carrera. A Edward Martínez, por hacer parte de mi proceso de formación cuando fui parte del equipo de difusión de la oficina de Promoción Institucional.

Agradezco a todos los profesores por los cuales he pasado en mi proceso en la Javeriana, en especial a Edwin Murcia, Javier López, Julio Carreño, Ángela Carrillo y Alexandra Pomares, por todos los conocimientos que de ellos he recibido, y por las enseñanzas adicionales para la vida que de ellos pude haber recibido. Mención especial a mi director de tesis, Jaime Andrés Pavlich Mariscal, por haber confiado en mí en la labor de realizar este trabajo de grado y la propuesta que le antecedió.

Agradezco también a Juan José Martínez Moritz, compañero de Ingeniería Electrónica con el que he convivido ratos agradables y también me ayudó a abrirme puertas hacia el mundo. Así mismo agradezco a todos los compañeros que han confiado en mí en los trabajos en grupo a lo largo de la carrera y los hemos sacado juntos, como también agradezco a aquellos que me dejaron solo, puesto que de ellos he aprendido el valor de la resiliencia y a no desfallecer en los momentos más complicados.

En resumen, a todas aquellas personas por las cuales he pasado en el proceso de mi carrera, como diría el fallecido músico Gustavo Cerati, GRACIAS TOTALES, por el aporte, poco o mucho, que han dejado en mi vida.

CONTENIDO

CONTENIDO v

INTRODUCCIÓN 1

I - DESCRIPCIÓN GENERAL 2

1. OPORTUNIDAD, PROBLEMÁTICA, ANTECEDENTES 2
   - *1.1. Formulación del problema que se resolvió* 3
   - *1.2. Justificación del problema* 3
   - *1.3. Impacto Esperado* 4

2. DESCRIPCIÓN DEL PROYECTO 4
   - *2.1. Objetivo general* 4
   - *2.2. Objetivos específicos* 4

3. METODOLOGÍA 4
   - *3.1. Concepción* 5
     - Método 5
     - Actividades 5
     - Resultados 6
   - *3.2. Construcción* 6
     - Método 6
     - Actividades 6
     - Resultados 7
   - *3.3. Documentación y pruebas* 7
     - Método 7
     - Actividades 7
     - Resultados 7

II – MARCO TEÓRICO 8

1. MARCO CONCEPTUAL 8
   - *1.1. Ingeniería dirigida por modelos (MDE)* 8
   - *1.2. Information System Modeling Language (ISML)* 9
   - *1.3. EMF* 10
   - *1.4. Xtext* 10
   - *1.5. Xtend* 11
   - *1.6. Acceleo* 11

1.7. PHP *12*
1.8. Los tres marcos de trabajo de PHP más populares *12*

1.8.1. Laravel 12

1.8.2. Symfony 13

1.8.3. Nette 14

2. TRABAJOS RELACIONADOS 14
2.1. PHPrunner *15*
2.2. PHPMaker *15*
2.3. ScriptCase *15*
2.4. PHP MySQL Wizard *16*
2.5. CodeCharge Studio *16*
2.6. AppGini *16*
2.7. SynApp2 ..... *16*
2.8. Spring Roo *17*
2.9. Comparación entre las soluciones ya hechas y lo que se construyó *17*

III – ANÁLISIS 19

1. ALCANCE Y PERSPECTIVA 19

2. RESTRICCIONES 19

3. FUNCIONES DEL SISTEMA 20

4. MODELO DE DOMINIO 21

5. REQUERIMIENTOS 25
5.1. Requerimientos funcionales *25*
5.2. Requerimientos no funcionales *29*

IV – DISEÑO 31

1. VISTA FÍSICA DEL SISTEMA 31

2. VISTA LÓGICA DEL SISTEMA 33

3. DISEÑO DETALLADO DEL SISTEMA 35
3.1 Clase `AnchurusLaravelGenerator` *35*
3.2 Paquete `co.edu.javeriana.isml.generator.common` *36*
3.3 Paquete `co.edu.javeriana.anchurus.generator.laravel.generators` *38*
3.4 Paquete `co.edu.javeriana.anchurus.generator.laravel.templates` *41*
3.5 Paquete `co.edu.javeriana.anchurus.generator.laravel.utils` *47*

V – DESARROLLO DE LA SOLUCIÓN 49

1. APLICACIÓN DE LA METODOLOGÍA DAD 49

2. ESTÁNDARES 49

3. FUNCIONALIDADES DE LA APLICACIÓN 50

3.1. Generación de código desde página ISML a página PHP 50

3.2. Generación de código desde controlador ISML a controlador PHP y archivo de rutas 52

3.3. Generación de código de entidad ISML a modelo y migración PHP 54

3.4. Archivo de configuración 56

VI – RESULTADOS 56

COMPARACIONES ENTRE CÓDIGO GENERADO Y MODELOS ISML 66

VII – CONCLUSIONES 68

1 ANÁLISIS DE IMPACTO DEL DESARROLLO 68

2 CONCLUSIONES Y TRABAJO FUTURO 69

VIII- REFERENCIAS Y BIBLIOGRAFÍA 70

IX – ANEXOS 78

ABSTRACT

Nowadays, web application development has a big learning curve for the programming languages used in this field.

Anchurus-GEN is born as an alternative to existing code generators which allows to generate PHP code from a high-level model, multiplatform and free. This tool was built using the Eclipse platform, through the Xtext plugin and allows to generate pages, migrations, models, controllers and the route archive. To validate the proper functioning of the application, system tests were executed. It is expected that this application will increase developers' productivity, because it will only be necessary to write the ISML model.

RESUMEN

El problema del desarrollo de aplicaciones web en la actualidad es la gran curva de aprendizaje para los lenguajes de programación utilizados en dicho campo.

Anchurus-GEN nace como una alternativa a generadores de código ya existentes que permite generar código en PHP desde un modelo de alto nivel, multiplataforma y gratuita. Esta herramienta fue construida utilizando la plataforma Eclipse, mediante el plug-in Xtext, permite generar páginas, migraciones, modelos, controladores y el archivo de rutas. Para validar el correcto funcionamiento de la aplicación se ejecutaron pruebas de sistema y se espera que esta aplicación aumente la productividad de quienes vayan a usarla, porque solamente será necesario escribir el modelo de la página.

INTRODUCCIÓN

Este documento de memoria contiene todo el proceso de desarrollo del Trabajo de Grado intitulado "Anchurus-GEN: Generador de código PHP a partir de modelos ISML", en adelante llamado el Trabajo de Grado. En el transcurso de este Trabajo de Grado se plantea la problemática que existe alrededor del desarrollo de aplicaciones en PHP y el cómo se podría subsanar dicha problemática mediante el uso de la herramienta que se desarrollaría en el Trabajo de Grado.

El Trabajo de Grado se dividió en siete secciones: descripción general, marco teórico, análisis, diseño, desarrollo de la solución, resultados y conclusiones.

En la sección I - DESCRIPCIÓN GENERAL, se encontrará una descripción sucinta de todo lo que dio origen a este Trabajo de Grado: problemática planteada, pregunta generadora e impacto esperado.

En la sección II – MARCO TEÓRICO, se dará contexto a todos aquellos conceptos que sean necesarios para la adecuada comprensión del Trabajo de Grado.

En la sección III – ANÁLISIS, se podrá observar cómo fue el proceso previo que terminó por dar lugar a la estructuración de Anchurus-GEN, mostrada de forma detallada en la sección IV – DISEÑO, donde se podrá ver el diseño del sistema, desde su despliegue en el hardware (capítulo 1 de la sección) hasta su estructuración en paquetes, clases y métodos (en el capítulo 3).

En la sección V – DESARROLLO DE LA SOLUCIÓN, se verá cómo se puso en práctica la metodología implementada en el Trabajo de Grado, los estándares utilizados y detalle de las funcionalidades de la aplicación.

En la sección VI – RESULTADOS se mostrará detalladamente cómo fueron las pruebas de Anchurus-GEN y también se detallará la comparación entre la escritura de modelos ISML y el código generado.

En la sección VII – CONCLUSIONES, se verá a qué conclusiones se llegaron en este Trabajo de Grado y qué desarrollos adicionales pudiera tener en un futuro Anchurus-GEN.

Para mayores detalles, en especial, sobre las secciones técnicas (Análisis y Diseño), remitirse a los anexos entregados con este documento.

I - DESCRIPCIÓN GENERAL

1. Oportunidad, Problemática, Antecedentes

Teniendo en cuenta el auge que ha tenido el tema del desarrollo de aplicaciones web en los últimos años [1], ha surgido el siguiente hecho problemático: construir aplicaciones web es dispendioso por la curva de aprendizaje que tienen los diversos marcos de trabajo. Por ejemplo, Java Enterprise Edition (JEE) posee un total de 2011 clases, repartidas en 139 paquetes [2], sin contar el aprendizaje adicional que implica aprender PrimeFaces o frameworks similares que sean compatibles con el lenguaje para hacer la aplicación web (en el caso de PrimeFaces, son cerca de 145 tipos de etiquetas diferentes [3], y según la aplicación tienen que utilizarse unos cuantos tipos). Este enorme volumen de información, por lo tanto, dificulta en gran manera el aprendizaje de este tipo de desarrollo.

La ingeniería dirigida por modelos, o *Model Driven Engineering* (MDE), tiene el potencial de resolver este inconveniente que se afronta a la hora de desarrollar aplicaciones web. El objetivo de MDE es definir modelos, métodos y herramientas aptas para la representación precisa y eficiente de sistemas de software. Así mismo, intenta englobar el ciclo de vida entero del sistema desde diversas dimensiones, como los requerimientos, funcionalidades, datos, procesamiento, dependencias, arquitectura e infraestructura [4].

MDE es una buena solución porque permite abstraer las aplicaciones web hacia modelos y reducir los niveles de complejidad relacionados con la especificación de la aplicación web. Este enfoque, así mismo, provee la posibilidad de automatizar la transformación del modelo hacia la aplicación [5].

En proyectos anteriores, un tándem [6] Universidad-empresa conformado por un equipo de trabajo de la Pontificia Universidad Javeriana y otro de la empresa Heinsohn Business Technology trabajó sobre un entorno de MDE conocido como ISML, que se concretó en el marco de trabajo, conocido dentro del mismo proyecto como Lion Framework. ISML son las siglas de *Information Systems Modeling Language*, en español: "Lenguaje de Modelado de Sistemas de Información", es un lenguaje textual de dominio específico, o *Domain Specific Language* (DSL) que permite expresar aplicaciones web en términos de modelo, sin importar la plataforma en la que se desarrollarán. [7]

ISML es un lenguaje textual sencillo en comparación a Java o C#, que utiliza el patrón modelo-vista-controlador (MVC) [8], que se aplica en partes del desarrollo de las aplicaciones web. Los elementos principales de ISML son las entidades (equivalente directo a las clases, solo que estas se guardan en un medio como una base de datos), las páginas (lo que se le mostrará al usuario), los controladores (los que manejan los elementos de la página) y los servicios (funcionalidades que tendrán los controladores). [7]

ISML permite, en primera instancia, reutilización de componentes, además de proveer modelado parcial de la aplicación. En segundo lugar, este lenguaje se empaqueta dentro de un conjunto de plugins de Eclipse, para que éstos últimos puedan hacer verificaciones de sintaxis y navegar entre elementos de un proyecto, y por último, y no menos importante, permite la

automatización en el desarrollo de los proyectos de software, mediante la generación de código [7].

Hasta el momento, se ha realizado la definición del lenguaje ISML en su totalidad y el desarrollo de un transformador de código que recibe como entrada un modelo ISML y devuelve en respuesta, un esqueleto de aplicación empresarial de Java (porque en muchas ocasiones, las funciones que tiene un servicio las tiene que implementar el desarrollador). También se ha desarrollado un transformador que recibe de entrada un modelo ISML que contiene únicamente entidades y sus relaciones, y devuelve un modelo que ya involucra los casos de uso CRUD para las entidades mostradas. [7]

Una de las plataformas en la que es deseable generar código para aplicaciones web es PHP. Por esta razón, la intención es extender las capacidades de la herramienta que ya están en Java EE; asimismo, se espera tener resultados en la generación de código parecidos a los obtenidos con Java EE: una línea de código del modelo equivale aproximadamente a 4 líneas y media de código Java EE, y esta proporción se eleva a 115 cuando se utiliza el transformador de CRUD [7].

1.1. Formulación del problema que se resolvió

¿Cómo acelerar el proceso de desarrollo de aplicaciones PHP utilizando ISML?

1.2. Justificación del problema

La problemática a afrontar es lo dispendioso que resulta desarrollar una aplicación en PHP mediante los métodos tradicionales y para ello se utilizará el ambiente MDE conocido como ISML. Pero, ¿por qué generar desde ISML hacia PHP?

Se desea generar código desde ISML hacia PHP por las siguientes razones: en primera instancia, ISML es un lenguaje textual bastante sencillo en comparación a PHP, en segundo lugar, porque fuerza a seguir una disciplina determinada para la creación de las páginas, en este caso, la obligación de crearlas sobre el patrón MVC, por lo mismo que ISML está dotado de más estructura que PHP, y también, que PHP al ser tan flexible, tiene el riesgo de generar código poco estructurado.

La generación de código se hace hacia PHP y no otro lenguaje, por lo siguiente: en primer lugar, PHP es un lenguaje portable, tiene *frameworks* que permiten su desarrollo en múltiples plataformas. Así mismo, existen generadores en proceso para lenguajes que son de código abierto que los desarrollan estudiantes de la Maestría de Ingeniería de Sistemas y Computación de la Pontificia Universidad Javeriana, para lenguajes distintos de PHP [9]. Por último, PHP es el sexto lenguaje más popular del mundo según el índice TIOBE. [10], y en relación con el desarrollo web, es el cuarto, detrás de Java, Python y C# [11].

Dadas las razones anteriores, el lenguaje destino del generador no es conveniente que sea Java porque el ya existente convierte modelos ISML a código Java EE. En ninguno de los tres C (C, C++ y C#) es práctico hacerlo por las siguientes razones: porque es complicado usarlo

en entornos web (en el caso de C y C++) o porque el lenguaje es propietario (caso de C#, que es de Microsoft). El caso de Python es explicable porque solo existe un marco de trabajo parcialmente conocido (Django, [12]) que permite el trabajo con aplicaciones web.

1.3. Impacto Esperado

A corto plazo se espera que este proyecto pueda continuar con la buena labor realizada con Zoe-GEN (generador de código hacia Java EE). En medio plazo se esperaría que este proyecto pueda tener una difusión moderada y haya empresas que utilicen esta herramienta para generar sus aplicaciones, y en el largo plazo se esperaría que grandes compañías usen Anchurus para sus aplicaciones; sin descontar la posibilidad de publicar en una gran base de conocimientos como lo son la IEEE, la ACM o Springer-Verlag.

2. Descripción del Proyecto

Las siguientes dos secciones mostrarán los objetivos que orientaron el proyecto que dio lugar a este documento de memoria. En la sección 2.1 se mostrará el objetivo general y en la 2.2, los objetivos específicos.

2.1. Objetivo general

Construir un generador de código bajo el marco de trabajo de Eclipse [13], que convierta un modelo ISML (Information System Modeling Language) a código en PHP.

2.2. Objetivos específicos

- Especificar los requerimientos del generador de código.
- Diseñar el producto basado en los requerimientos previamente obtenidos.
- Implementar el diseño provisto anteriormente.
- Probar la implementación que se diseñó con diferentes modelos ISML.

3. Metodología

El proyecto se orientó eminentemente por la metodología conocida como *Disciplined Agile Delivery* (DAD). Se trata de un método ágil para el desarrollo de software adaptable a la organización que lo vaya a acoger, posee un ciclo de vida orientado por metas y está regido por los riesgos que se tomen [14]. El DAD se divide en las siguientes fases que se muestran en el siguiente gráfico [14]:

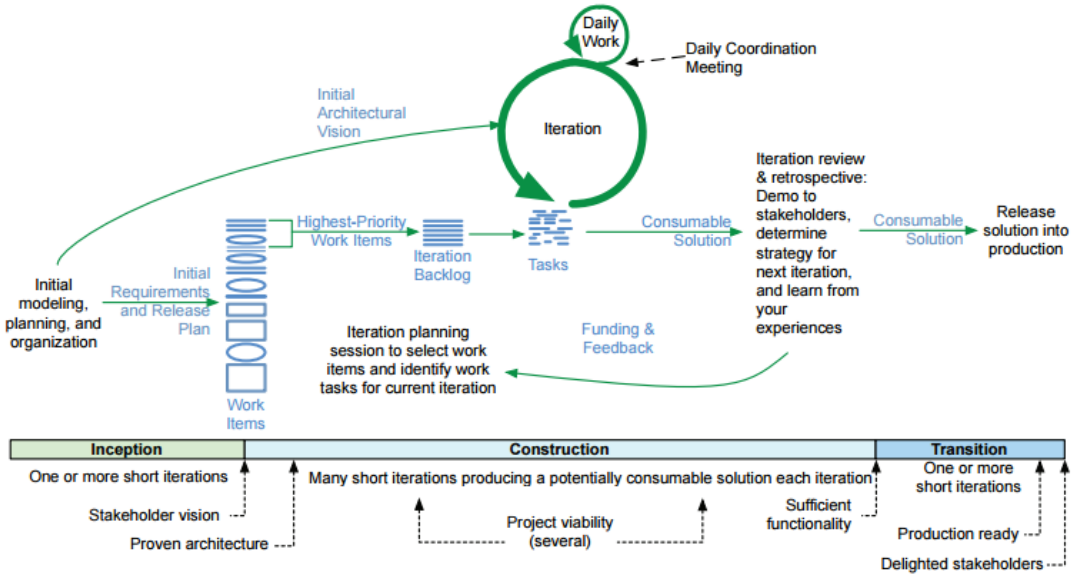


Ilustración 1 Proceso original de DAD

Aunque el proceso original contemplaba tres fases: concepción, construcción y transición, no se incluyó la de transición, sino que en su lugar se hizo una de documentación y pruebas.

A continuación se detallan los resultados de cada una de las fases:

3.1. Concepción

En esta fase, básicamente, se construyó el embrión de lo que fue el trabajo de grado, con la propuesta de trabajo de grado, en adelante, el Documento. Este se llenó de manera progresiva durante el semestre de Seminario, con gran parte de las actividades que se plantearon y trabajo al margen de dichas actividades.

Método

La mayoría de las actividades contempladas en esta fase fueron basadas en lo realizado en la clase de Seminario de Metodología de la Investigación. Adicionalmente, se construyó el documento de visión y una primera versión de la especificación de requerimientos, cuya información se obtuvo de las conversaciones con el cliente que fue el Director de Trabajo de Grado.

Actividades

Las actividades que compusieron la primera parte de esta fase fueron un total de seis, desde la segunda en adelante todas tuvieron una íntima relación con el Trabajo de Grado, la primera fue para detectar el área donde uno tiene talento y pasión. La segunda parte de esta fase la compusieron cuatro actividades, que fueron las más determinantes para el éxito de toda la fase. Una tercera fase la compusieron las actividades inmediatamente anteriores a la fase de Construcción.

La descomposición de las actividades fue la siguiente:

1. Primera parte: inicio de construcción del Documento
    a. Desarrollo de la Encuesta de Hermann y de la ficha de Gustos/Habilidades.
    b. Localización de información concerniente al tema de investigación.
    c. Formulación de la pregunta generadora
        i. Descripción de la problemática
        ii. Formulación de la pregunta
        iii. Justificación
    d. Definición del objetivo general del Trabajo de Grado
    e. Definición de los objetivos específicos del Trabajo de Grado
    f. Definición de la metodología del Trabajo de Grado
2. Segunda parte: presentación al público y consolidación del Documento
    a. Feria de Posters
    b. Definición de la gestión del proyecto
    c. Concreción del marco teórico y bibliografías pertinentes
    d. Entrega del Documento.
3. Preparación para la fase de Construcción
    a. Elaboración del documento de visión
    b. Versión inicial de la especificación de requerimientos

Resultados

Los resultados de la fase de concepción fueron: la entrega del Documento y la primera versión de la especificación de requerimientos.

3.2. Construcción

En esta fase se ejecutó lo planteado en el Documento, y se empezó a diseñar y desarrollar el generador de código sobre la base construida en la primera especificación de requerimientos.

Método

Se definió un número finito de ítems de trabajo, que a efectos de este proyecto, serían los requerimientos obtenidos en la especificación de requerimientos, sobre los cuales se trabajó durante toda la fase de construcción. Se organizaron según una prioridad definida y, con los más prioritarios, se desarrolló un generador que satisfizo esos ítems de trabajo, en una primera iteración. Una vez se construyó el primer generador, se definieron los pasos a seguir para lo que vino en la siguiente iteración, que comenzaría de manera parecida, definiendo prioridades y desarrollando el mismo generador de las anteriores iteraciones, así hasta la penúltima iteración. En la última iteración, salió el generador de código con su respectiva documentación (porque ya no quedarían más ítems en los que trabajar).

Actividades

Esta fase se compuso de cinco actividades, de las cuales cuatro se hicieron de manera iterativa, hasta que no quedaron ítems de trabajo por hacer.

1. Obtención de los ítems de trabajo
2. Organización por prioridad de los ítems (faltantes en caso de no estar todos)
3. Desarrollo de ese generador según esos ítems prioritarios.
4. Entrega de la solución consumible
5. Definición de la estrategia para la siguiente iteración (si existen aún ítems de trabajo) o para la siguiente fase (si ya acabaron).

Resultados

Los resultados de la fase de construcción fueron: el generador de código terminado, el documento de diseño y la especificación de requerimientos final.

3.3. Documentación y pruebas

Esta parte es externa a DAD, se adaptó porque no se hizo etapa de transición por la razón expuesta en la introducción al proceso.

Método

Básicamente, con el generador ya hecho, se realizaron pruebas con modelos preexistentes y construidos para probar el funcionamiento del programa. Igualmente, se precisó hacer los manuales para los usuarios del generador, hacer la memoria del trabajo de grado y un sitio web en el servidor Pegasus de la Pontificia Universidad Javeriana que albergó todo lo realizado durante todo el proceso.

Actividades

Son un total de cuatro actividades definidas que en algunos casos fue necesario ejecutarlas en paralelo. Son las siguientes:

1. Elaboración del manual de usuario final y de desarrollador
2. Ejecución de las pruebas sobre el generador de código
3. Elaboración del documento de memoria del Trabajo de Grado
4. Elaboración del Sitio web en Pegasus.

Resultados

Los resultados de esta fase fueron: los manuales de usuario y desarrollador, el reporte de pruebas, este documento de memoria y el sitio web en Pegasus.

II – MARCO TEÓRICO

1. Marco Conceptual

A continuación se mostrarán los conceptos básicos para comprender la terminología mostrada en este trabajo de memoria. Los conceptos de tipo teórico están en las secciones 1.1 y 1.2, y las secciones restantes están destinadas para los conceptos de tipo programático, que orientarán sobre las herramientas utilizadas.

1.1. Ingeniería dirigida por modelos (MDE)

MDE es un paradigma dentro de la ingeniería de software que aboga por el uso de los modelos y las transformaciones que se den entre ellos, como piezas claves para orientar las actividades relacionadas por la ingeniería de software. [15]

Complementando esta definición dada por los profesores de la UOC en la citación anterior, Rodrigues da Silva en su artículo *Model-driven engineering, a survey supported by the unified conceptual model*, dice que MDE es una aproximación de la ingeniería de software que considera a los modelos no solamente como documentación sino como ciudadanía de primera clase, que pueden usarse en cualquier disciplina de ingeniería y en cualquier dominio de aplicación [16]. Como se dijo antes, es paradigma de ingeniería de software y aquí complementa diciendo que, como tal, no tiene soporte de herramientas a utilizar [16].

Estando MDE en la cima de la abstracción de las aproximaciones "dirigidas por modelos", las que le siguen (y se contienen en MDE), ya bajando el nivel de abstracción son MDD, MBT y MDA.

MDD son las siglas en inglés de *Model-driven Development*, Desarrollo dirigido por modelos en español. Esta aproximación se centra en las disciplinas de construcción de requerimientos, análisis, diseño e implementación. Las aproximaciones concretas de MDD tienden a definir lenguajes de modelado para especificar el sistema en estudio en diferentes niveles de abstracción, proveyendo transformaciones de modelo a modelo y de modelo a texto para mejorar la productividad y la calidad del proceso y el sistema de software final [16].

MBT son las siglas en inglés de *Model-based Testing*, Prueba basada en modelos en español. Se centra esta aproximación en la automatización de la disciplina de pruebas. Los modelos de prueba se utilizan para representar los comportamientos deseados del sistema en pruebas, representar estrategias de prueba y representar el entorno de prueba. Un modelo de prueba describiendo el sistema en pruebas es normalmente una representación abstracta y parcial del comportamiento del comportamiento deseado del sistema en pruebas. Los casos de prueba derivados desde ese modelo son pruebas funcionales en el mismo nivel de abstracción que el modelo, y pueden ser pasados a pruebas ejecutables comunicables con el sistema en pruebas bajo herramientas de prueba y marcos de trabajo específicos [16].

MDA son las siglas en inglés de *Model-driven Architecture,* arquitectura dirigida por modelos. Aproximación propuesta por el Object Management Group (OMG) [17] en los primeros

años de la década del 2000, está dirigida de forma primordial en la definición y transformación de los modelos. MDA soporta definición de modelos a diferentes niveles de abstracción, a saber: *Computational Independent Models* (CIM), *Platform Independent Models* (PIM) y *Platform Specific Models* (PSM). Las transformaciones consideradas son las siguientes: CIM a CIM, CIM a PIM, PIM a PIM, PIM a PSM, PSM a PSM y de PSM a material textual [16].

1.2. Information System Modeling Language (ISML)

Se trata de un lenguaje textual de dominio específico que permite expresar modelos de aplicaciones web sin importar las tecnologías donde se vayan a desenvolver. El metamodelo [18] que describió este lenguaje se escribió utilizando la herramienta de código abierto conocida como Xtext [19] [7].

ISML se caracteriza por ser un lenguaje sencillo, comparado a lenguajes como Java o C#, utilizar el patrón MVC [8], uso de servicios y reutilización de componentes preexistentes, facilidades para modelar parcialmente una aplicación web, uso de los plugins de Eclipse y la posibilidad de transformar los modelos construidos en ISML hacia código fuente que represente aplicaciones web [7].

Las aplicaciones web expresadas en el lenguaje ISML se inspiran parcialmente en el patrón MVC [8], donde se contemplan cuatro tipos de elementos: **entidades** de negocio que se guardan en una base de datos y representan el Modelo de la aplicación, **páginas** que es lo que se provee al usuario mediante Vista de las entidades, los **Controladores**, que definen lo que se hará con las peticiones de los usuarios en las páginas, definen las reglas de navegación e invocan **servicios**, que son los ejecutantes de la lógica de negocio y manejan las entidades en la aplicación [7].

Entrando a los rudimentos del lenguaje, una entidad se define bajo la palabra reservada **entity,** de manera similar a una clase en Java, indicando sus atributos. La diferencia aquí es que se pueden indicar, mediante las palabras reservadas **must be,** las restricciones que puede tener dicho atributo. Los atributos de relación entre entidades se indican mediante la palabra reservada **opposite,** que indica que lo que se tiene a la derecha de esa palabra lo tiene que indicar lo que está en la izquierda (ejemplo: si en una entidad Concesionario se tiene un arreglo de entidades Vehículo **opposite** concesionario; en cada uno de los vehículos debe indicarse que tiene un concesionario) [7].

Las páginas se definen bajo la palabra reservada **page.** Poseen los elementos típicos de una página web, como etiquetas, cajas de texto, botones, tablas entre otros. Cada página se controla mediante un controlador (que los disparan normalmente botones), de la misma manera que una clase implementa métodos de una interfaz (utiliza la palabra reservada **controlledBy** entre la definición de la página y el nombre del controlador) [7].

Cada controlador posee unos métodos que van a llamar las páginas que controla, y tiene así mismo, servicios subyacentes que les ayudan a hacer el trabajo. Se definen bajo la palabra reservada **controller** y los servicios se importan mediante la directiva **has** [7].

Los servicios, como ya se dijo, los invocan los controladores y se encargan por completo de la lógica de negocio. Se definen mediante la palabra reservada **service** y pueden tener un funcionamiento parecido al de los *templates*. Los servicios tienen los métodos de negocio en el modelo, pero eso no significa que en la implementación vaya a ser igual. Al marcar con **native** un método de negocio, no lo va a implementar desde el modelo sino se espera que el desarrollador se encargue de ese trabajo [7].

1.3. EMF

El proyecto Eclipse Modeling Framework (Marco de Trabajo de Modelado de Eclipse) es un marco de trabajo de modelado y un centro de generación de código para herramientas de construcción y otras aplicaciones basadas en un modelo de datos estructurado. Desde una especificación de modelo descrita en XMI [20], EMF provee herramientas y soporte en tiempo de ejecución para producir un conjunto de clases Java para el modelo, junto con un conjunto de clases adaptadoras que permiten visión y edición basada en comandos del modelo y un editor básico [21].

El núcleo EMF es un estándar común en el que se basan modelos de datos, tecnologías y otros marcos de trabajo. Esto incluye soluciones de servidor, marcos de trabajo de persistencia, frameworks de interfaz de usuario y soporte para transformaciones. Consiste en tres piezas fundamentales, EMF a secas, EMF.Edit y EMF.Codegen.

EMF a secas incluye un metamodelo (conocido como Ecore) para describir otros modelos y soporte en tiempo de ejecución para los modelos, que incluye notificación de cambios, soporte de persistencia con serialización XMI por defecto y una muy eficiente API reflexiva para manejar genéricamente los objetos EMF [21].

EMF.Edit incluye clases genéricas reutilizables para construir editores para modelos EMF. En el caso de EMF.Codegen, este centro de generación de código es capaz de generar todo lo necesario para construir un editor completo para un modelo EMF. Incluye una interfaz gráfica de usuario desde donde las opciones de generación pueden especificarse y pueden invocarse los generadores. El centro de generación se aprovecha de las herramientas de desarrollo de Java de Eclipse [21].

1.4. Xtext

Xtext es un marco de trabajo para desarrollar lenguajes de programación y lenguajes de dominio específico. Cubre todos los aspectos de una infraestructura completa de lenguaje, desde el analizador gramatical, enlazador, compilador o intérprete hasta la integración completa en el entorno de desarrollo integrado, o IDE [22] (siglas en inglés de *Integrated Development Environment)* de Eclipse [13]. Xtext cuenta con grandes estándares en todos esos aspectos que al mismo tiempo pueden ser adaptados para las necesidades individuales [19].

Xtext provee una colección de lenguajes de dominio específico e interfaces de programación de aplicaciones o APIs (siglas en inglés de *Application Programming Interfaces*) modernas que describen los diferentes aspectos del lenguaje de programación a realizar. Basado en esa información, Xtext da una implementación de dicho lenguaje sobre la máquina virtual de

Java. Los componentes de compilador de dicho lenguaje son independientes de Eclipse o de OSGi (siglas de *Open Services Gateway initiative*) [23] y pueden ser utilizadas en cualquier entorno Java. Incluyen componentes como el analizador gramatical, el árbol de sintaxis abstracto, el serializador y el formateador de código, y el enlazador, chequeo de compilador y validación, así como el intérprete [19].

Estos componentes, en tiempo de ejecución, se integran y están basados en el *Eclipse Modeling Framework* (EMF) o Marco de Trabajo de Modelado de Eclipse en español que permite usar Xtext con otros componentes EMF como por ejemplo el *Graphical Modeling Framework* (GMF) [24] [19].

1.5. Xtend

Es un lenguaje de programación de tipado estático que se traduce a código fuente comprensible de Java. Sintáctica y semánticamente Xtend tiene raíces en Java (es un dialecto, a fin de cuentas) pero lo mejora en muchos aspectos: métodos de extensión, expresiones Lambda, anotaciones activas, sobrecarga de operadores, estructura de control switch más poderosa, envío múltiple (conocido también como invocación de métodos polimórfica), expresiones plantilla, no hay sentencias (todo es una expresión en este lenguaje), propiedades, inferencia de tipos, soporte completo para lo genérico de Java y traducción a Java, como se dijo anteriormente [25].

Xtend carece de problemas de interoperabilidad con Java, contrario a otros lenguajes que utilizan la máquina virtual de Java. Al mismo tiempo, Xtend es más conciso, legible y expresivo. La librería de Xtend es una capa delgada que provee utilidades y extensiones encima del JDK [25].

Es posible llamar los métodos de Xtend desde Java, en una forma transparente. Además, Xtend provee un IDE basado en Eclipse íntimamente integrado con las herramientas de desarrollo de Java en Eclipse, incluyendo características como jerarquías de llamado, reestructuración de renombrado, depuración y muchas más [25].

1.6. Acceleo

Se trata de una implementación pragmática del estándar del OMG [17] conocido como MOF *(Meta Object Facility)* [26] Model to Text Language (MOFM2T) [27]. Apoya al desarrollador con muchas de las herramientas que pudieran ser esperadas de un entorno de desarrollo de generación de código de alta calidad, como por ejemplo, sintaxis sencilla, generación eficiente de código y características al nivel de las herramientas de desarrollo de Java [28].

Las características de Acceleo incluyen: resaltado de sintaxis, autocompletado, plegado de código, reconstrucción, plantillas de código personalizables, soporte para Maven [29] y para Ant [30], generación incremental de código, compilador, detección de errores, depurador entre muchas otras. Aunque es un plugin de Eclipse, aún no es altamente integrado en Java [28].

1.7. PHP

Acrónimo de "PHP, Hypertext Processor", es un lenguaje de *scripting* de propósito general y de código abierto que está pensado específicamente en el desarrollo web y puede ser incrustado en páginas HTML. Su sintaxis es parecida a la de C, Java y Perl, haciéndolo fácil de aprender. El objetivo principal del lenguaje es permitir al desarrollador web escribir dinámica y rápidamente páginas web generadas, aunque se puede hacer más con el lenguaje [31].

Las páginas de PHP contienen HTML con código incrustado que hace algo. Ese código que hace algo está encerrado entre las etiquetas de comienzo y fin "<? php" y "?>" [31].

PHP es un lenguaje que se ejecuta del lado del servidor, es decir, que el usuario recibe un código HTML producto de ese PHP que el cliente nunca sabrá qué lleva por debajo.

Aunque PHP se utilice para desarrollo web, tiene otros usos también. Como por ejemplo el uso de scripts desde línea de comandos (por ejemplo, para ejecutar cierto trabajo cada X tiempo con los cron jobs) y desarrollar aplicaciones de escritorio mediante PHP-GTK, que es un fork de PHP que implementa GTK [32] [31].

PHP es un lenguaje portable, multiplataforma, compatible con los sistemas operativos principales, como Windows, Linux, MacOS X y variantes de Unix como HP-UX [33], Solaris y OpenBSD [34], lo que da pie a tener virtual libertad de elegir qué servidor web se usará para la ejecución del lenguaje. Y este lenguaje se puede escribir de manera procedimental o con estilo de programación orientada a objetos [31].

La generación de lenguaje que tiene inherente PHP no está limitada a HTML sino a la generación de cualquier tipo de texto, como XHTML [35] o archivos XML [36]. Y si vamos más lejos, PHP puede crear imágenes, PDF y Flash, sobre la marcha. Una de las características más importantes de este lenguaje es el soporte que tiene para una amplia gama de bases de datos, como lo son MySQL y Oracle, mediante las extensiones propias de cada base de datos, usando la capa de abstracción PDO [37] o usando ODBC [38] [31].

PHP cuenta con el soporte para comunicarse con otros servicios, como LDAP [39], IMAP [40], SNMP [41] entre muchos otros. Se pueden también crear sockets de red puros e interactuar utilizando otros protocolos. También incluye características de procesamiento de texto, como las expresiones regulares de Perl y análisis de documentos XML [31].

1.8. Los tres marcos de trabajo de PHP más populares

Según Sitepoint.com [42], estos son los marcos de trabajo de PHP más populares para el desarrollo:

1.8.1. Laravel

Laravel, como los otros dos, son marcos de trabajo para desarrollar aplicaciones sobre PHP. Su filosofía está basada en que el desarrollo debe ser una experiencia disfrutable y creativa, para ser verdaderamente enriquecedora. Su meta es facilitar la mayoría de las tareas que se

realizan en un proyecto web, como la autenticación, el enrutamiento, las sesiones y el manejo de la caché [43].

En este marco de trabajo, como ya se dijo, se pretende que el desarrollador disfrute de la experiencia, pero sin sacrificar la funcionalidad. En ese intento, Laravel integró lo mejor de lo que ya se ha visto en otros marcos de trabajo web, como Ruby on Rails [44], ASP.NET MVC [45] y Sinatra [46] [43].

A pesar de ser un marco de trabajo accesible, es poderoso, proveyendo herramientas necesitadas para grandes y robustas aplicaciones. Posee contenedor de controles, un sistema de migración expresivo (mediante comandos) y unidad de pruebas [43].

El enrutamiento entre páginas PHP se maneja en este marco de trabajo con métodos de una clase conocida como Route. Dicha ruta puede servir al método POST o al GET de HTTP, siendo posible que sirva a ambas, o incluso que esa ruta se distribuya de manera segura, HTTPS. Las rutas pueden tener parámetros, ser nombradas, hacer un filtro para que solamente bajo ciertas condiciones, se pueda acceder a ellas, agruparlas (para no hacer el mismo filtro para varias rutas, por ejemplo), fijarlas a un modelo existente, o también, arrojar el error 404 desde ellas en caso de no existir el recurso [43].

La obtención de las cookies, respuestas y solicitudes al servidor se hacen también de forma metódica, es decir, llamando a métodos de las clases correspondientes, Cookie, Response y Request. En PHP típico esto se hace llamando a los arreglos asociativos que corresponden a las cookies y los métodos HTTP [43].

El manejo de las bases de datos, en términos de consultas y operaciones de administración de la misma, como no podría ser de otra manera, es metódico, usando los métodos de la clase DB. Las consultas pueden ser también construidas utilizando los constructores de consultas que provee el mismo lenguaje. Para efectos de construir clases que persistan en la base de datos, Laravel provee el mapeo objeto-relacional [47] conocido como Eloquent, el cual es una implementación del patrón Active Record [48] [43].

1.8.2. Symfony

En palabras de sus creadores, SensioLabs, "Symfony es un conjunto de componentes PHP, un marco de trabajo de aplicación web, filosofía y comunidad, trabajando juntos en armonía". La filosofía es que cada quien pueda crear sus módulos a fin de mejorar el marco de trabajo de Symfony, y es posible hacerlo, pues Symfony está bajo una licencia de código abierto [49].

Este marco de trabajo cuenta con 34 componentes, que sirven para diferentes tipos de propósito. Debug, por ejemplo, se encarga de las tareas de depuración del código. Config, de los valores de configuración y sus validaciones. Form, del procesamiento de los formularios HTML (creación, uso y reutilización). Guard es el módulo de seguridad. Intl es lo que se usa cuando falla o falta la extensión intl en PHP típico. Validator y ClassLoader son complementarias: una extensión valida las clases y la otra, las sube al sistema si son válidas (esto es, si siguen determinados estándares PHP). Stopwatch sirve de disparador de eventos y funciona

como un cronómetro: tiene métodos para iniciar, parar y crear una vuelta (esto es, parar y de inmediato volver a comenzar el evento) [49].

En el caso de los enrutamientos, este marco de trabajo utiliza controladores (y métodos en el interior de ellos) para hacerlos, asociados con el archivo de enrutamiento que está en formato YAML [50] [49].

La construcción de objetos que persistirán en la base de datos, análogamente a Laravel, usa un mapeo objeto-relacional, pero este se conoce con el nombre de Doctrine. Las operaciones de Doctrine no son metódicas como son las de Eloquent, sino que son manejadas mediante la línea de comandos [49].

1.8.3. Nette

Se trata de otro marco de trabajo para PHP cuya filosofía radica en el centrarse en lo más importante de la aplicación, sin tener que hacer tareas repetitivas. Cuenta con una sintaxis eficaz y comprensible, un depurador innovador y características de seguridad líderes en la industria [51].

Este marco de trabajo se caracteriza por tener un sistema de construcción de plantillas sobresaliente, herramientas de diagnóstico que dicen, son insuperables, una capa de base de datos que es eficaz, como se dijo anteriormente, cuenta con características de seguridad líderes en la industria, es compatible con HTML5, AJAX y es amistoso con el posicionamiento en motores de búsqueda. También se caracteriza por tener un desempeño sobresaliente, utilizar la orientación a los objetos que provee PHP 5, combinarse con otros marcos de trabajo y tener una oferta creciente de añadidos [51].

Hablando ya de generalidades del marco de trabajo, Nette puede crear tablas en una base de datos en una facilidad conocida como Adminer provista dentro del mismo marco de trabajo. Las plantillas en Nette se crean de manera parecida a como se crean en Laravel, aunque tienen una sintaxis bastante diferente, y no se guardan en la misma extensión que un PHP típico. Aparte, las plantillas en Nette permiten incluir atributos adicionales dentro de las etiquetas HTML tradicionales, conocidas como "n: atributos", que permiten ahorrar espacio que habitualmente se malgastaría en validaciones dentro de los bloques de código en los archivos Latte (aquellos que funcionan de plantilla) [51].

2. Trabajos Relacionados

En esta sección se mostrarán trabajos relacionados con generación de código en PHP. Las primeras ocho secciones (de la 2.1. PHPrunner a la 2.8. Spring Roo) muestran el trabajo relacionado y una pequeña descripción sobre él. La sección 2.9. Comparación entre las soluciones ya hechas y lo que se construyó mostrará la comparación que se hizo con estos trabajos relacionados y la herramienta construida y cuál es el valor agregado que tiene sobre las demás.

2.1. PHPrunner

Es un generador de páginas web que construye interfaces web, a partir de bases de datos, sean remotas o locales, que vengan desde servidores MySQL, Access, SQL Server, Postgres y Oracle [52].

Aparte del soporte para múltiples bases de datos, se caracteriza por ofrecer plantillas para el negocio para el que se vaya a utilizar la herramienta, como por ejemplo páginas específicas para avisos clasificados o para un concesionario de vehículos. Tiene así mismo un editor visual donde uno puede arrastrar y poner los componentes que sean necesarios en la página, todo desde una plantilla que el mismo PHPRunner propone para el trabajo. Entre otras características, tiene también la posibilidad de ver la vista preliminar de cómo quedará la aplicación antes de ser publicada, edición de consultas SQL, soporte multilenguaje y controles para edición de texto enriquecido [52].

Aunque es una herramienta que es bastante versátil, tiene la grave desventaja de que solamente funciona en Windows y en Internet Explorer. Su versión completa tiene un alto costo también, de 499 dólares [52].

2.2. PHPMaker

Análogamente a PHPrunner, Maker puede generar páginas PHP desde bases de datos de similares orígenes a PHPrunner, MySQL, Access, Postgres, SQL Server y Oracle [53].

Una de sus características es la integración que tiene con Bootstrap [54] y por lo tanto, que desde este generador se pueden hacer sitios web para teléfonos móviles. Las tablas que se hagan dentro de este programa son exportables a diversos formatos de archivo (como archivos de Excel (xls), pdf, csv, xml o inclusive imprimirla) [53].

Es una herramienta que también cuenta con gran versatilidad, aunque solamente es descargable para Windows, sin embargo, el servidor puede ser Linux o Windows sin ningún problema. La versión que no es de prueba, tiene un costo de 299.90 dólares, incluyendo la mejora que se le realizó [53].

2.3. ScriptCase

Con Runner comparte la multiplicidad en soporte de sistemas de bases de datos. Pero es de las pocas cosas que comparten. ScriptCase está más orientado hacia las altas esferas de la organización, porque provee muchas herramientas orientadas al análisis, como los gráficos, lista de chequeo sobre lo que hay que hacer y un calendario. Es así mismo de las pocas que proveen integración hacia un sistema de pagos como lo es Paypal [55].

Más versátil que los anteriores, por la cantidad de herramientas que se pueden obtener desde este generador, y es también el único multiplataforma (pudiendo ejecutarse en Linux, Mac, Windows y otros sistemas operativos). Así mismo, es también el más costoso, según la licencia puede costar desde 359.10 hasta 629.10 dólares [55].

2.4. PHP MySQL Wizard

Se trata de otra herramienta (únicamente para Windows) que convierte datos en una base de datos construida en MySQL, con la particularidad de que, contrario a las otras herramientas citadas, es de código abierto (en el sentido de que se puede distribuir el resultado de lo realizado en este asistente de construcción) [56].

Como características principales, aparte de la posibilidad de distribuir lo realizado con la aplicación, están la generación automática de PHP para manejar los registros de la base de datos, poder generar una cantidad ilimitada de proyectos, validación de formularios y opciones de seguridad [56].

Aunque es de código abierto, hay que pagar para tener la licencia completa, que vale 48 dólares, aunque tienen una oferta con otras dos herramientas (conversión de Excel y Access a MySQL) que vale 94 dólares [56].

2.5. CodeCharge Studio

Este es un generador desde una base de datos a página web, pero de propósito general, puesto que permite convertir casi cualquier tipo de base de datos (como Access, Oracle o SQL Server) en páginas web de PHP, ASP, JSP, entre otras [57].

Aparte de la ya mencionada característica de convertir una base de datos de cualquier tipo en páginas web de cualquier lenguaje, incluye un IDE y un generador de código y también incluye un set de soluciones preconstruidas [57].

A pesar de la versatilidad entre plataformas de bases de datos y páginas web, programáticamente este generador solo es compatible con Windows. Para este generador también hay que pagar por la versión completa, que fluctúa entre los 140 dólares para la licencia anual en la versión personal (por cada usuario) y los 500 dólares por usuario para la licencia perpetua del producto [57].

2.6. AppGini

Otro generador de páginas web desde una base de datos que es únicamente compatible con Windows y genera código desde MySQL hacia PHP. Sus características son las de reducción en pruebas, tener código seguro (inmune a inyecciones SQL, por ejemplo) y la reducción de costos por tiempo de desarrollo, aparte que permite realizar páginas web para móviles.

Se trata de otra herramienta por la cual hay que pagar por su versión completa, en esta ocasión, 80 dólares. [58]

2.7. SynApp2

Se trata de un generador que convierte los datos provenientes de una base de datos MySQL u Oracle hacia páginas escritas en PHP, pero con el añadido del uso de AJAX. Sus más desta-

cadas características incluyen la rapidez con la que manejan los datos relacionales y que es una herramienta de código abierto [59].

No es una herramienta muy conocida, teniendo muy pocos resultados cuando se busca en google (1550 resultados de búsqueda), aunque es una herramienta gratuita.

2.8. Spring Roo

Se trata de un generador de código para Java Empresarial (JEE) que carece de interfaz gráfica de usuario, puesto que todo lo hace en línea de comandos. Es una herramienta de código abierto y es posible hacer *fork* de esta aplicación a través de GitHub.

No utiliza tiempo de ejecución, no es plugin ni es una librería para procesar anotaciones.

Se trata de una herramienta gratuita y es multiplataforma. [60]

2.9. Comparación entre las soluciones ya hechas y lo que se construyó

Se utilizaron ocho criterios para comparar las soluciones existentes y la que se construyó: si es o no de código abierto, su precio, si genera o no código, si lo generado es código PHP, si posee interfaz gráfica, si es multiplataforma, si es para todos los navegadores, si tiene propósito general y si puede generar código desde un modelo de alto nivel.

Tabla 1 Comparación entre soluciones

	Código abierto	Precio	Genera código	PHP	GUI	Multiplataforma	Para todo navegador	Propósito general	Genera desde un modelo de alto nivel
PHPRunner	No	499 USD	Sí	Sí	Sí	No	No	Sí	No
PHPmaker	No	300 USD	Sí	Sí	Sí	Cliente Windows, servidor Linux o Windows	Sí	Sí	No
ScriptCase	No	359 a 630	Sí	Sí	Sí	Sí	Sí	No	No

		USD según licen-cen-cia							
PHP MySQL Wizard	Si	48 a 94 USD	Sí	Sí	Sí	No	Sí	Si	No
Co-deCharge Studio	No	140 a 500 USD	Sí	Sí	Sí	No	Sí	Sí	No
AppGini	No	80 USD	Sí	Sí	Sí	No	Sí	Sí	No
SynApp2	Sí	Gratis	Sí	Sí	Sí	Sí	Sí	Sí	No
Spring Roo	Sí	Gratis	Sí	No	No	Sí	Sí	Sí	No
Anchu-rus-GEN	Sí	Gratis	Sí	Sí	Si	Sí	Sí	Sí	Sí

La solución construida, Anchurus-GEN, tiene el valor agregado de que construye una aplicación a partir de un modelo de alto nivel como es ISML, así que es mejor que cualquiera existente en el mercado (su competencia más cercana en términos de funcionamiento, Spring Roo, carece de interfaz gráfica y no genera código PHP).

III – ANÁLISIS

En esta sección se explica cómo fue la fase de análisis de Anchurus-GEN. En primera instancia se referenciará el alcance que se tuvo a la hora de construir el producto y a qué se quiso llegar (la perspectiva), en segunda instancia, las restricciones y los requerimientos estarán en tercer lugar. Como último, pero no menos importante, estará el plan de pruebas. Mayores informes sobre el proceso de análisis se pueden encontrar en [61].

1. Alcance y perspectiva

Lo que se pretendió hacer con Anchurus-GEN fue lograr un generador de código integrado a la plataforma de Eclipse como plug-in de la misma, con las funcionalidades definidas en la sección 5.2. Cualquier trabajo adicional que se pretenda hacer sobre este proyecto se considera trabajo futuro, que se definirá en la sección 0 del bloque VII – CONCLUSIONES.

Anchurus-GEN se trata de un producto orientado a los ingenieros de software que conozcan de diseño de aplicaciones web. El producto generará gran parte de los componentes de una página web escrita en PHP bajo el marco de trabajo de Laravel usando como entrada, modelos de un lenguaje de dominio específico conocido como ISML.

Anchurus-GEN entrará a formar parte de una familia de sistemas parecidos donde se encuentra Zoe-GEN [9], que es un generador de código para Java EE, y también hará parte de los plug-ins de Eclipse, lo cual hará que un desarrollador web tenga más de donde elegir cuando acometa un proyecto de software.

2. Restricciones

Anchurus-GEN tiene cuatro tipos diferentes de restricciones: generales, relacionadas con software y hardware y restricciones legales.

La restricción general es que el usuario final requiere tener unos conocimientos medios de inglés para poder utilizar la aplicación.

Restricciones relacionadas con software son las relacionadas con el software huésped (o sea, aquel donde se instalará el plug-in, en este caso, Eclipse), las cuales son que el sistema debe funcionar como un plug-in de Eclipse y que se debe poder instalar en cualquier versión de Eclipse que soporte Xtext 2.8.

Las restricciones de hardware tienen, en este caso, que ver con la pantalla (ha de tener una resolución mínima de 1024*768 píxeles) y la memoria RAM de la máquina donde se ejecute el sistema, la cual debe ser de, por lo menos, 1.3 GB, sugiriéndose más por la presencia de la máquina virtual Vagrant. Y la restricción legal tiene que ver con el licenciamiento, que ha de ser liberado bajo Apache 2.0.

3. Funciones del sistema

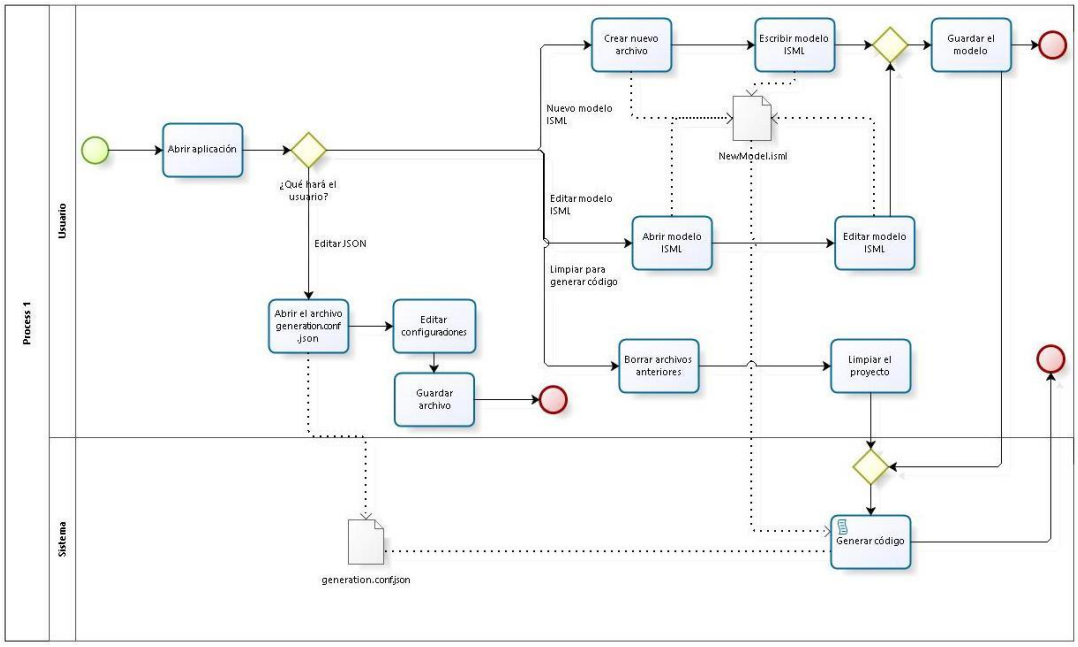


Ilustración 2 Funcionalidades de Anchurus

La anterior imagen es un diagrama BPMN de las funcionalidades de Anchurus-GEN. Primero, un usuario crea y escribe un modelo ISML (o lo edita), luego limpia el proyecto para generar el código y opcionalmente edita el archivo de configuración.

Cuando se va a crear el modelo, se tiene que crear un archivo ISML dentro de la carpeta que se vaya a guardar, se escribe el modelo según lo que se necesite (página, controlador, entidad o servicio), se guarda y el sistema crea el código equivalente al modelo guardado. A la hora de editar primero se abre el archivo, se modifica según conveniencia del usuario y los dos últimos pasos son idénticos a los del proceso de crear.

Cuando se va a limpiar el proyecto, primero se ha de borrar los archivos de migración que se hayan generado en un proceso anterior, luego se ejecuta la limpieza propiamente dicha mediante la opción que provee la plataforma y mientras se ejecuta este proceso, en segundo plano se genera código.

Cuando se quiere editar el archivo de configuración, primero se tiene que buscar dónde está y luego abrirlo, se editan las configuraciones necesarias teniendo en cuenta los nombres puestos en la clase maestra y se guarda dicho archivo después.

4. Modelo de dominio

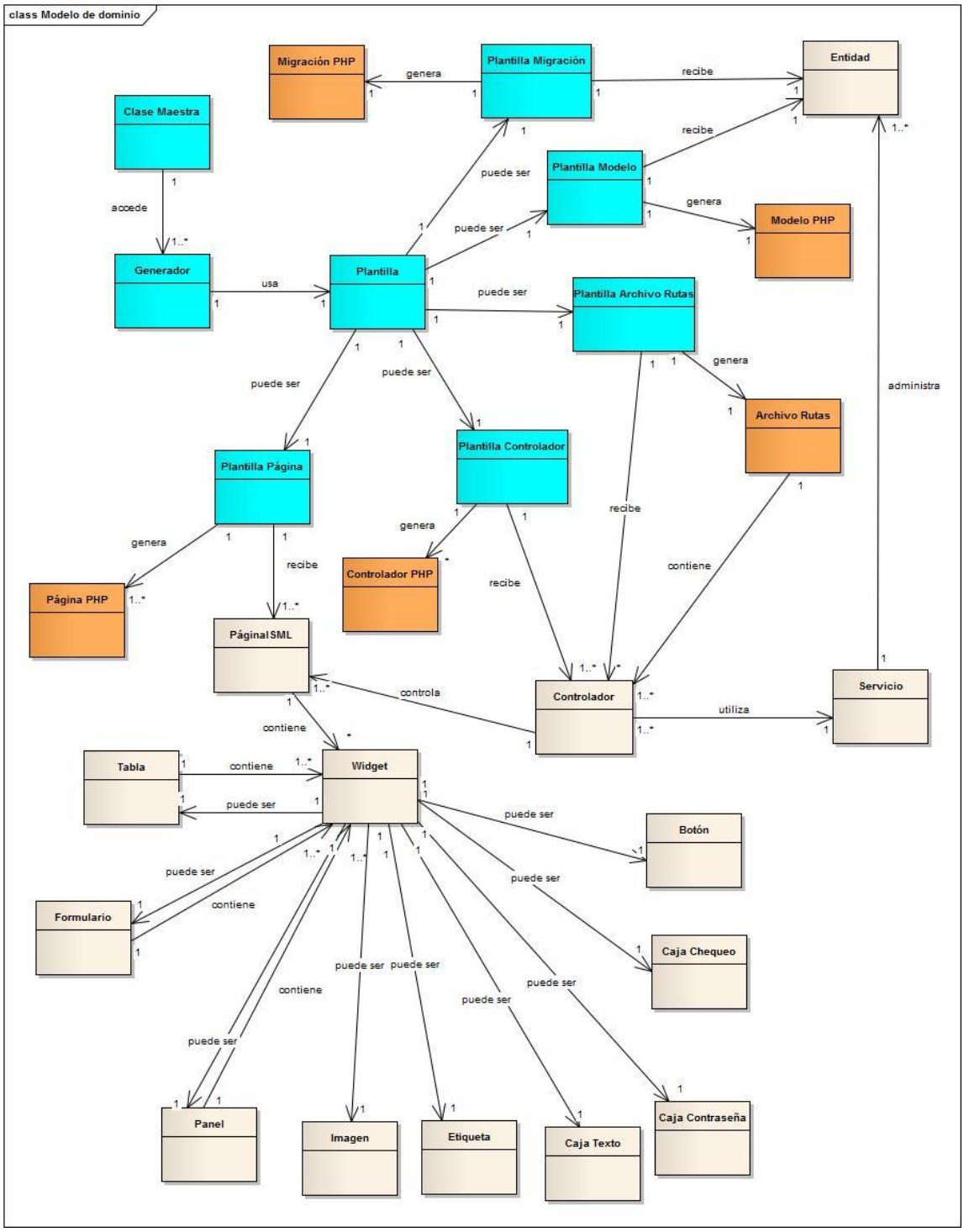


Ilustración 3 Modelo del dominio de Anchurus-GEN

El modelo de dominio del sistema Anchurus-GEN se divide en tres grandes grupos de elementos: elementos de desarrollo, elementos de PHP y elementos de ISML, indicados cada uno por colores diferentes (celeste, naranja y beige respectivamente).

La especificación de cada grupo es como sigue:

- Elementos de desarrollo: aquellos que derivarán en los objetos que se utilizarán para el desarrollo de la aplicación. Son 8 elementos, descritos en la siguiente tabla:

Elemento	Descripción
Clase Maestra	La clase desde donde se accede al sistema y contiene todos los generadores.
Generador	Aquel que, ayudado por una Plantilla, genera archivos de código PHP.
Plantilla	Es el molde de cualquier archivo que se vaya a generar. Para efectos de este sistema, hay cinco plantillas distintas.
Plantilla Migración	Es una Plantilla que transforma Entidades a Migraciones PHP.
Plantilla Modelo	Es una Plantilla que transforma Entidades a Modelos PHP.
Plantilla Página	Es una Plantilla que transforma Páginas ISML a Páginas PHP.
Plantilla Controlador	Es una Plantilla que transforma Controladores a Controladores PHP.
Plantilla Archivo Rutas	Es una plantilla que transforma un conjunto de Controladores a Archivo Rutas.

Tabla 2 Descripción de los elementos de desarrollo

- Elementos de PHP: los productos de la generación de código, proveniente de un elemento de ISML y pasada por un Generador. Son 5 elementos, descritos en la siguiente tabla:

Elemento	Descripción
Modelo PHP	Es un archivo con la sintaxis de Eloquent que refleja un objeto almacenado en la base de datos.
Migración PHP	Es un archivo que, después de ejecutarse un comando, actualiza la base de datos con creación o eliminación de tablas.
Página PHP	Una página web escrita en el lenguaje de programación PHP.
Controlador PHP	Un archivo PHP que permite ejecutar las peticiones que envía una o más páginas.
Archivo Rutas	Aquel que indica todas las posibles redirecciones de una aplicación web en PHP.

Tabla 3 Descripción de los elementos de PHP

- Elementos de ISML: son aquellos que modelan una aplicación web en el lenguaje antes mencionado. Cuenta con 14 elementos, descritos en la siguiente tabla:

Elemento	Descripción
Servicio	Se trata de una estructura que tiene normalmente métodos sin implementar y que el programador tiene que implementarlos en el lenguaje de destino de la generación de código. Administra varias Entidades y lo pueden múltiples Controladores.
Controlador	Es una estructura que contiene métodos que ejecutarán las Páginas ISML mediante las llamadas a acción que poseen. Un controlador solamente puede controlar una página a la vez.
Entidad	Representación de un objeto que se almacenará en la base de datos. La administra un Servicio.
Página ISML	Es una estructura que permite conocer la interfaz gráfica de una página web. Es manejada por un Controlador y posee múltiples Widgets.
Widget	Cualquier elemento que tenga una Página ISML dentro de su estructura. Para efectos de este sistema, hay nueve tipos de Widget.
Botón	Un Widget que al hacerle clic, ejecuta una acción que se le asigne.
Caja Chequeo	Un Widget que tiene dos valores posibles, marcado y no marcado.
Caja Texto	Un Widget que puede ser llenado con texto y tiene un tamaño determinado.
Caja Contraseña	Un Widget especialmente diseñado para diligenciar contraseñas. Es parecido a una Caja Texto pero oculta el texto que se le escriba.
Etiqueta	Un Widget que cumple la función de mostrar el texto que tenga.
Imagen	Un Widget que muestra un archivo de imagen que se enlaza desde alguna parte.
Panel	Es un tipo de widget que permite hacer divisiones dentro de una página web, y puede contener otros widgets.
Formulario	Es un tipo de widget que permite que los valores inscritos en los widgets allí insertados puedan ejecutar una petición HTML.
Tabla	Es un tipo de widget que muestra los widgets insertados allí en formato de filas y columnas.

Tabla 4 Descripción de los elementos de ISML

5. Requerimientos

5.1. Requerimientos funcionales

Los requerimientos funcionales de Anchurus-GEN tienen que ver, básicamente, sobre qué va a generar el sistema y cómo va a generarlo (es decir, el formato en el que tienen que salir los objetos generados en código). Son los siguientes:

# Requerimiento	***1***	***Tipo de Requerimiento***	***Funcional***
Descripción	***El Sistema debe generar el archivo de una página en PHP en formato Blade a partir de una página de ISML***		
Autor	***Frank Sebastián Franco Hernández***		

# Requerimiento	***2***	***Tipo de Requerimiento***	***Funcional***
Descripción	***El Sistema debe generar el archivo de un controlador en PHP a partir de un controlador de ISML***		
Autor	***Frank Sebastián Franco Hernández***		

# Requerimiento	***3***	***Tipo de Requerimiento***	***Funcional***
Descripción	***El Sistema debe generar el archivo de clase en PHP a partir de un servicio de ISML***		
Autor	***Frank Sebastián Franco Hernández***		

# Requerimiento	***4***	***Tipo de Requerimiento***	***Funcional***

Descripción	***El Sistema debe generar el archivo de migraciones en PHP a partir de una entidad de ISML***
Autor	***Frank Sebastián Franco Hernández***

# Requerimiento	***5***	***Tipo de Requerimiento***	***Funcional***
Descripción	***El Sistema debe generar el archivo de rutas en PHP a partir de un conjunto de controladores de ISML.***		
Autor	***Frank Sebastián Franco Hernández***		

# Requerimiento	***6***	***Tipo de Requerimiento***	***Funcional***
Descripción	***El Sistema debe generar el archivo de modelo en PHP a partir de una entidad de ISML***		
Autor	***Frank Sebastián Franco Hernández***		

# Requerimiento	***7***	***Tipo de Requerimiento***	***Funcional***
Descripción	***El Sistema debe generar el código para mostrar un botón en PHP en formato Blade a partir de un botón de ISML.***		
Autor	***Frank Sebastián Franco Hernández***		

# Requerimiento	***8***	***Tipo de Requerimiento***	***Funcional***
Descripción	***El Sistema debe generar el código para mostrar una caja de***		

	chequeo en PHP en formato Blade a partir de una caja de chequeo de ISML.
Autor	***Frank Sebastián Franco Hernández***

# Requerimiento	***9***	***Tipo de Requerimiento***	***Funcional***
Descripción	***El Sistema debe generar el código para mostrar una tabla en HTML a partir de una tabla de ISML.***		
Autor	***Frank Sebastián Franco Hernández***		

# Requerimiento	***10***	***Tipo de Requerimiento***	***Funcional***
Descripción	***El Sistema debe generar el código para mostrar un formulario en PHP en formato Blade a partir de un formulario de ISML.***		
Autor	***Frank Sebastián Franco Hernández***		

# Requerimiento	***11***	***Tipo de Requerimiento***	***Funcional***
Descripción	***El Sistema debe generar el código para mostrar una imagen en PHP en formato Blade a partir de una imagen de ISML.***		
Autor	***Frank Sebastián Franco Hernández***		

# Requerimiento	***12***	***Tipo de Requerimiento***	***Funcional***
Descripción	***El Sistema debe generar el código para mostrar una caja de texto en PHP en formato Blade a partir de una caja de texto de***		

	ISML.
Autor	***Frank Sebastián Franco Hernández***

# Requerimiento	***13***	***Tipo de Requerimiento***	***Funcional***
Descripción	***El Sistema debe generar el código para mostrar una etiqueta en PHP en formato Blade a partir de una etiqueta de ISML.***		
Autor	***Frank Sebastián Franco Hernández***		

# Requerimiento	***14***	***Tipo de Requerimiento***	***Funcional***
Descripción	***El Sistema debe generar el código para mostrar una división en HTML a partir de un panel de ISML.***		
Autor	***Frank Sebastián Franco Hernández***		

# Requerimiento	***15***	***Tipo de Requerimiento***	***Funcional***
Descripción	***El Sistema debe generar el código para mostrar un campo de contraseña en PHP en formato Blade a partir de un campo de contraseña de ISML.***		
Autor	***Frank Sebastián Franco Hernández***		

5.2. Requerimientos no funcionales

Los requerimientos no funcionales de Anchurus-GEN son de dos clases: de mantenibilidad y de portabilidad. Los primeros tienen que ver con cómo tiene que ser la estructura de las clases del sistema y los segundos, sobre en qué sistemas operativos puede operar Anchurus-GEN. Son los siguientes:

# Requerimiento	***16***	***Tipo de Requerimiento***	***Mantenibilidad***
Descripción	***El sistema debe estar dividido por paquetes***		
Autor	***Frank Sebastián Franco Hernández***		

# Requerimiento	***17***	***Tipo de Requerimiento***	***Mantenibilidad***
Descripción	***El sistema debe tener cada plantilla como una clase separada***		
Autor	***Frank Sebastián Franco Hernández***		

# Requerimiento	***18***	***Tipo de Requerimiento***	***Mantenibilidad***
Descripción	***El sistema debe tener cada generador como una clase separada***		
Autor	***Frank Sebastián Franco Hernández***		

# Requerimiento	***19***	***Tipo de Requerimiento***	***Mantenibilidad***
Descripción	***El sistema debe tener sus funcionalidades comunes en una o más clases utilitarias***		
Autor	***Frank Sebastián Franco Hernández***		

# Requerimiento	***20***	***Tipo de Requerimiento***	***Mantenibilidad***
Descripción	***El sistema debe tener los generadores referenciados desde la clase maestra.***		
Autor	***Frank Sebastián Franco Hernández***		

# Requerimiento	***21***	***Tipo de Requerimiento***	***Portabilidad***
Descripción	***El sistema debe ser compatible con Linux, con Windows y con MacOS/X.***		
Autor	***Frank Sebastián Franco Hernández***		

# Requerimiento	***22***	***Tipo de Requerimiento***	***Portabilidad***
Descripción	***El código generado debe ser compatible con las plataformas soportadas por Laravel.***		
Autor	***Frank Sebastián Franco Hernández***		

IV – DISEÑO

En esta sección se explica cómo fue el diseño de Anchurus-GEN. Se empezará mostrando el despliegue de la aplicación en el sistema (vista física), en segundo lugar se mostrará la vista de componentes (relación de la aplicación con su entorno de software, vista lógica) y finalmente se dará lugar al diseño detallado del sistema, paquetes y clases con sus relaciones y los métodos y atributos que tengan dentro.

1. Vista física del sistema

La siguiente ilustración es el diagrama de despliegue de Anchurus-GEN apenas se encuentre en producción:

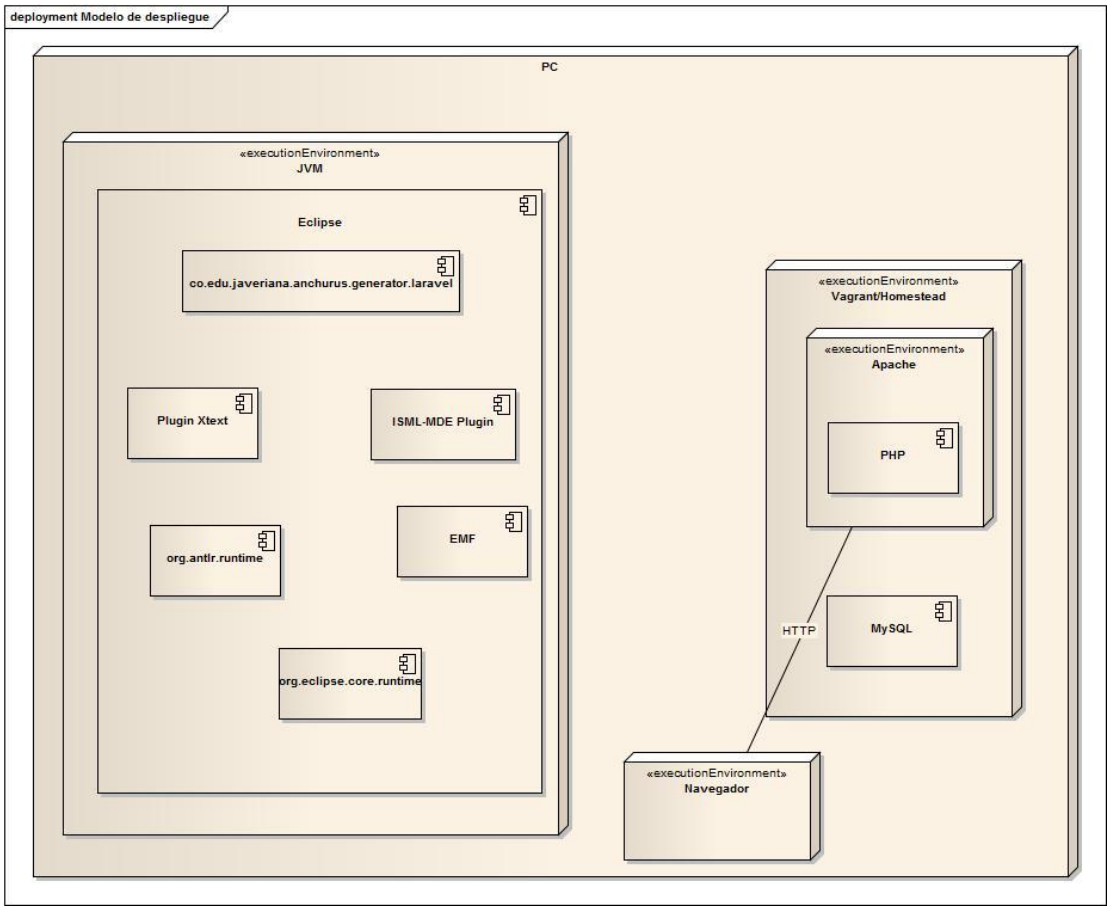


Ilustración 4 Diagrama de despliegue de Anchurus-GEN

El nodo PC es una máquina que contiene todos los componentes necesarios para trabajar con el sistema instalado. Contiene el nodo de la máquina virtual de Java (JVM), el nodo de la máquina virtual Vagrant configurada como Homestead (Vagrant/Homestead) y el nodo del navegador.

Eclipse (ejecutándose en JVM) contiene los elementos necesarios para que Anchurus-GEN funcione, que son el plugin de Xtext (Plugin Xtext), el plugin ISML-MDE (ISML-MDE Plugin), el componente EMF (EMF a secas dentro del diagrama), el núcleo de ejecución de Eclipse (org.eclipse.core.runtime) y el componente ANTLR (org.antlr.runtime), aparte del sistema propiamente dicho (co.edu.javeriana.anchurus.generator.laravel). Aquí se aprovecha el principal valor de Eclipse como plataforma que es su extensibilidad, puesto que se pueden insertar tantos plugins como el proyecto lo demande y la máquina lo soporte [62].

Cada uno de los componentes insertos en Eclipse (y por lo tanto en JVM) cumple una labor muy específica. El plugin de Xtext es un marco de trabajo para desarrollar lenguajes de programación y lenguajes de dominio específico [63], el plugin de ISML es un conjunto de plugins que permiten manejar modelos de dicho lenguaje y generar código a partir de los modelos, el componente EMF es un framework que permite hacer tratamiento de modelos en general en Eclipse [64], el núcleo de ejecución de Eclipse es lo que permite que Eclipse mismo funcione [65] y el componente ANTLR brinda la capacidad de los análisis léxico y sintáctico para interpretar los modelos ISML, que son la entrada de la generación de código [66]. Sobre el sistema como tal, su mayor especificación se encuentra en el diseño detallado.

Eclipse con Anchurus-GEN generará código en PHP, que para ser probado y ejecutado se desplegará en un servidor Apache con PHP y MySQL instalados. Este servidor Apache (y MySQL) están en la máquina virtual Vagrant configurada como Homestead.

Vagrant (configurado como Homestead) es una máquina virtual que permite crear y configurar entornos de desarrollo portables, de bajo peso y reproducibles [67]. En este caso Vagrant tiene dentro Apache, un servidor web de código abierto gratuito [68] y MySQL, un gestor de bases de datos, también de código abierto [69]. Dentro de Apache está PHP [70], un lenguaje de programación del lado de servidor para sitios web.

El navegador, que se conecta vía HTTP al servidor Apache dentro de Vagrant/Homestead, es cualquier cliente ligero con capacidad de mostrar sitios Web (especialmente aquellos escritos en PHP).

2. Vista lógica del sistema

La siguiente es la ilustración del diagrama de componentes del sistema Anchurus-GEN:

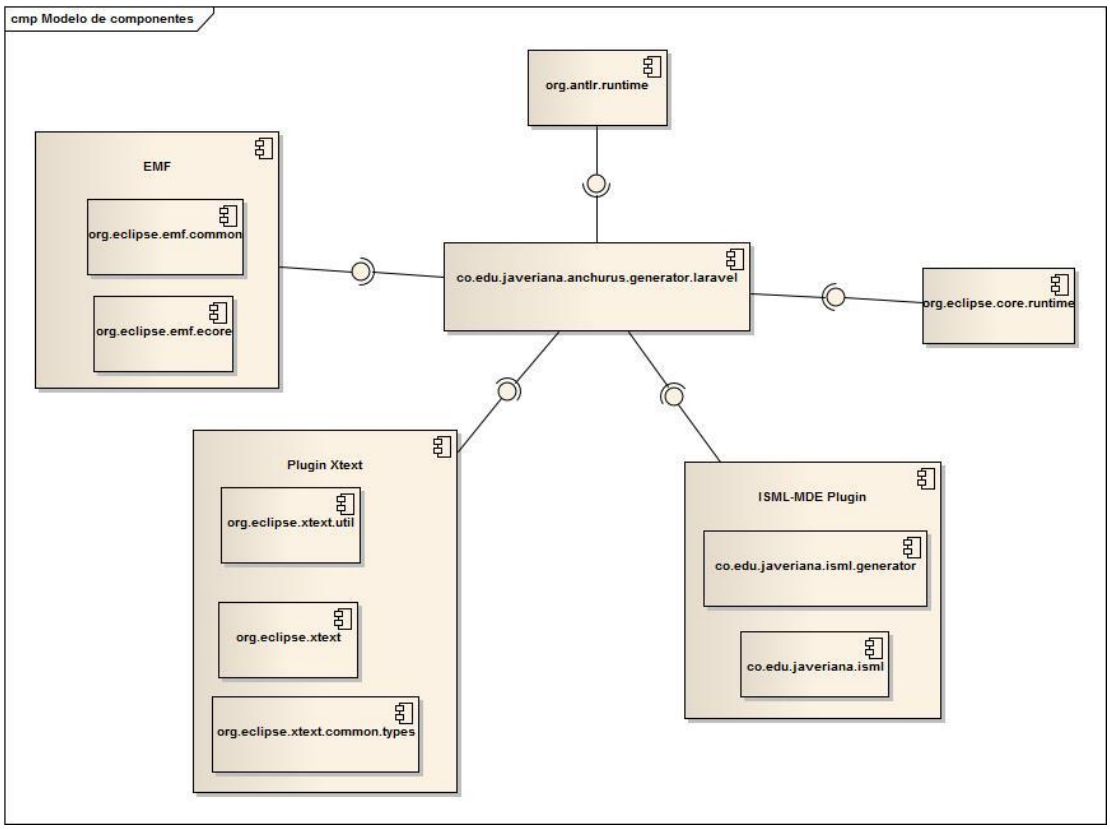


Ilustración 5 Diagrama de componentes de Anchurus-GEN

Anchurus-GEN consta de seis componentes, cinco de ellos proveen interfaz al sistema central.

1. **co.edu.javeriana.anchurus.generator.laravel**: Es donde se encuentra el sistema propiamente dicho, cuya mayor especificación se encuentra en el diagrama de clases del diseño detallado.
2. **org.eclipse.core.runtime:** Son los cimientos de Eclipse. Como proyecto por aparte dentro de Eclipse, abre otra instancia del IDE en tiempo de ejecución. [65]
3. **org.eclipse.antlr.runtime:** Es un componente que brinda análisis léxico y sintáctico para efectos de la generación de código mediante la herramienta ANTLR. [66]
4. **Plugin Xtext:** Marco de trabajo para el desarrollo de lenguajes de programación y lenguajes de dominio específico. [63]
   4.1. **org.eclipse.xtext:** Se trata del componente raíz de Xtext, que colabora con el procesamiento del lenguaje textual. [63]

4.2. **org.eclipse.xtext.util:** Trabaja en el procesamiento del texto y en la construcción del árbol de sintaxis del lenguaje textual. [71]
4.3. **org.eclipse.xtext.common.types:** Este componente trabaja directamente en la máquina virtual de Java y trabaja sobre el tipado de Xtext. [72]

5. **EMF:** Se trata del framework para el tratamiento de modelos y metamodelos en Eclipse. [5]

5.1. **org.eclipse.emf.common:** Funciones comunes para el manejo de modelos y metamodelos en Eclipse. [64]
5.2. **org.eclipse.emf.ecore:** Es una API para el metamodelo conocido como Ecore, del EMF. [73]

6. **ISML-MDE Plugin:** Conjunto de todos los plugins que permiten manipular modelos en ISML y hacer generación de código a partir de ellos.
6.1. **co.edu.javeriana.isml:** Se trata del paquete raíz de ISML, donde se encuentran todos los constructos del lenguaje.
6.2. **co.edu.javeriana.isml.generator:** Es la API de ISML para generación de código hacia otros lenguajes, como Java EE o PHP.

3. Diseño detallado del sistema

En este punto se explicará cómo se estructura por dentro Anchurus-GEN, desde lo más general (paquetes) hasta lo más particular (clases y métodos), para dar una idea sobre el funcionamiento del sistema.

La siguiente es una ilustración del diagrama de paquetes del sistema Anchurus-GEN:

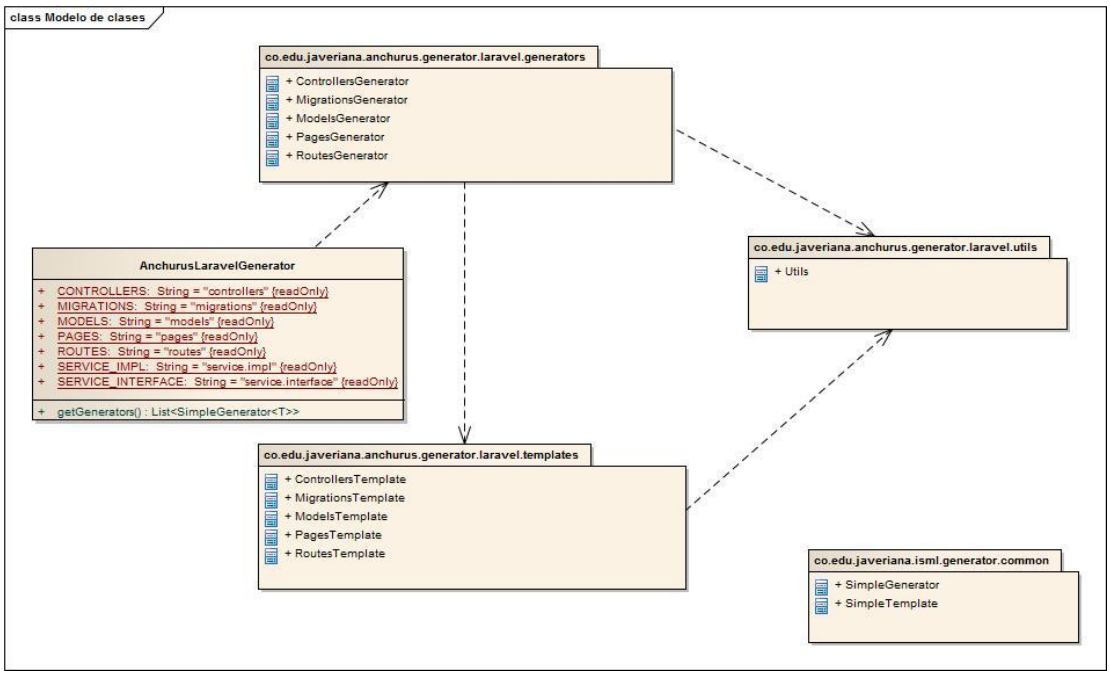


Ilustración 6 Diagrama de paquetes de Anchurus-GEN

Cuenta con una clase maestra (`AnchurusLaravelGenerator`) y cuatro paquetes, el paquete `co.edu.javeriana.anchurus.generator.laravel.generators`, el paquete `co.edu.javeriana.anchurus.generator.laravel.templates`, el paquete `co.edu.javeriana.anchurus.generator.laravel.utils` y el paquete `co.edu.javeriana.isml.generator.common`. Los dos primeros paquetes utilizan métodos del tercer paquete y el primer método pide instancias de clases del segundo paquete. El cuarto paquete, aparentemente no relacionado, tiene relaciones internas con las clases del paquete de generadores y del paquete de plantillas, que se detallarán más adelante.

3.1 Clase `AnchurusLaravelGenerator`

Es la clase maestra del proyecto y es donde parte toda la tarea de generación de código del sistema. Tiene 7 atributos estáticos y finales y un método. La función de cada uno de los valores de los atributos es servir de conexión entre el sistema y el archivo de configuración JSON en el proyecto donde se generará el código, con el fin de crear los directorios donde se emplazarán los archivos de código generado. Dicho así, el atributo `CONTROLLERS` es la cone-

xión con los archivos de controladores, `MIGRATIONS` con los archivos de migraciones, MODELS con los archivos de modelos, `PAGES` con los archivos de las páginas, `ROUTES` con el archivo de rutas, `SERVICE_IMPL` debe hacerlo con la implementación de los servicios y `SERVICE_INTERFACE` con la declaración de los servicios. La función del método `getGenerators()` es crear la instancia de cada generador de código del paquete `laravel.generators`.

3.2 Paquete `co.edu.javeriana.isml.generator.common`

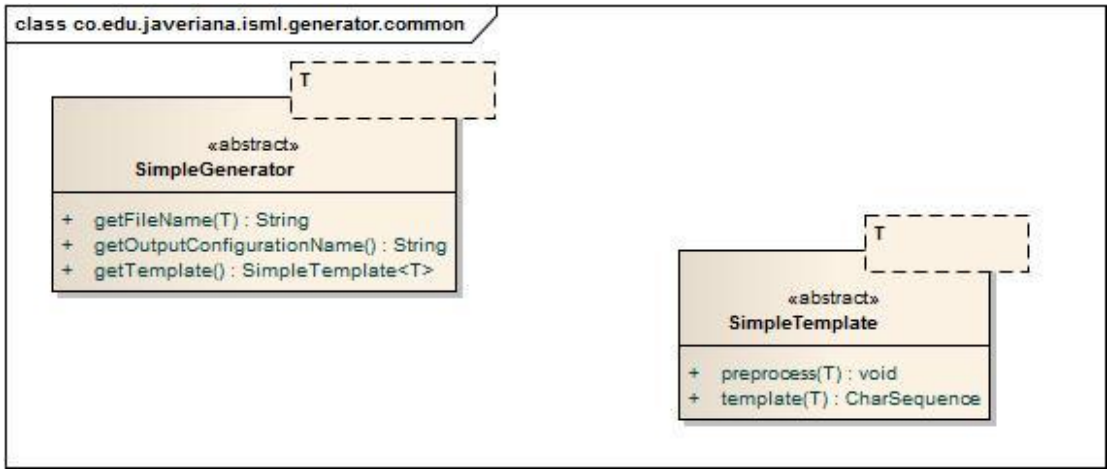


Ilustración 7 Paquete de funciones comunes de ISML

Estas son dos de las clases del paquete `co.edu.javeriana.isml.generator.common`. `SimpleGenerator` es la superclase de aquellas situadas en `laravel.generators` y `SimpleTemplate` lo es con las clases situadas en `laravel.templates`. En las dos siguientes secciones se explicará qué clases heredan de estas dos clases. El desglose del funcionamiento de estas clases se explica en las siguientes tablas:

Clase	`SimpleGenerator`
Descripción	Es la clase padre de los generadores del paquete generators.
Método	**Descripción**
`String getFileName(T)`	Da nombre al archivo que se generará.
`String getOutputConfigurationName()`	Recupera un nombre identificador único para el generador. Dicho identificado permite localizar la configuración de dicho generador en el archivo

	`generation.conf.json`.
`SimpleTemplate<T> getTemplate()`	Obtiene la plantilla de la cual se generará el archivo.

Tabla 5 Clase `SimpleGenerator`

Clase	`SimpleTemplate`
Descripción	Es la superclase de las plantillas del paquete templates.
Método	**Descripción**
`void preprocess(T)`	Provee todo lo necesario antes de procesar la plantilla que va a derivar en el archivo que se va a generar.
`CharSequence template(T)`	La plantilla que dará lugar al archivo a generar.

Tabla 6 Clase `SimpleTemplate`

3.3 Paquete `co.edu.javeriana.anchurus.generator.laravel.generators`

La siguiente es la estructura interna del paquete `laravel.generators`:

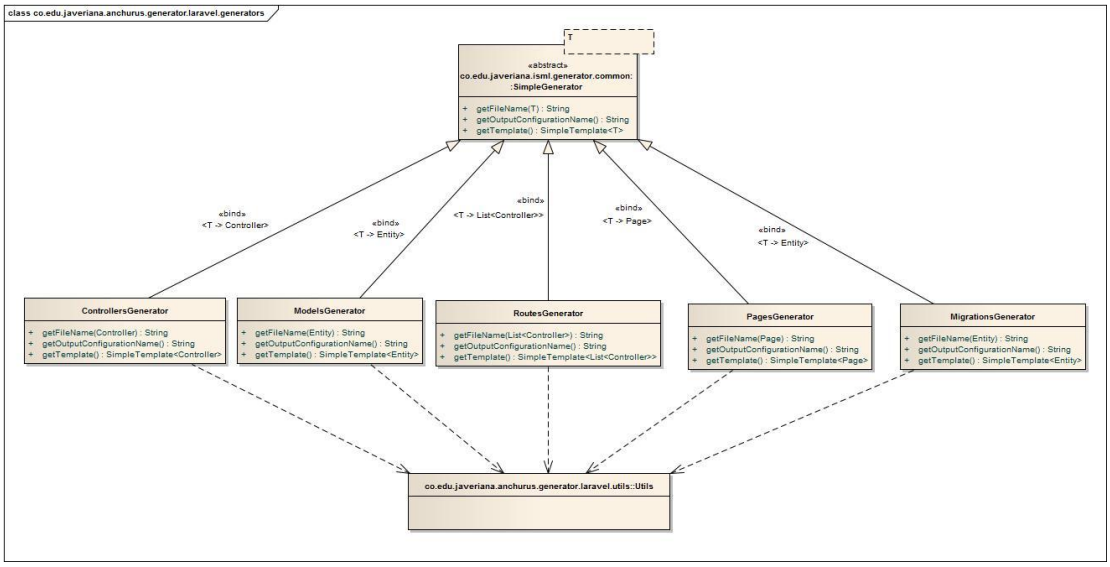


Ilustración 8 Paquete de generadores

Este paquete contiene cinco clases que heredan de la clase *abstracta* `SimpleGenerator` del paquete `co.edu.javeriana.isml.common`, que se explicó más atrás. Las clases de este paquete son las generadoras de los objetos en PHP, que se apoyan en los templates a los que llaman en sus métodos `getTemplate()`. Todas las clases tienen tres métodos que hacen exactamente lo mismo: obtener el nombre de configuración de salida (hacia el archivo json) de la clase específica (métodos `getOutputConfigurationName()`), `getTemplate()` que crea una instancia de cada template del paquete `co.edu.javeriana.anchurus.generator.laravel.templates` y `getFileName(T)` (reemplazando T por cada clase de entrada), obtiene el nombre del archivo, pero de una manera distinta según el caso.

Cada clase le da nombre a su función: `ControllersGenerator` es la generadora de controladores, `ModelsGenerator` es la generadora de modelos, `RoutesGenerator` es la generadora del archivo de rutas, `PagesGenerator` es la generadora de páginas y `MigrationsGenerator` es la generadora del archivo de migraciones. Explicación sobre el funcionamiento de cada método según cada clase se encuentra en las siguientes tablas:

Método / Clase	`ControllersGenerator`
`getTemplate()`	Crea un nuevo `ControllersTemplate`

`getOutputConfigurationName()`	Recupera el atributo `CONTROLLERS` de la clase `AnchurusLaravelGenerator`
`getFileName(T)`	T es un controlador. Da nombre al archivo del controlador generado, con el nombre del controlador de ISML.

Tabla 7 Clase `ControllersGenerator`

Método / Clase	`ModelsGenerator`
`getTemplate()`	Crea un nuevo `ModelsTemplate`
`getOutputConfigurationName()`	Recupera el atributo `MODELS` de la clase `AnchurusLaravelGenerator`
`getFileName(T)`	T es una entidad. Da nombre al archivo del modelo generado, con el nombre de la entidad de ISML.

Tabla 8 Clase `ModelsGenerator`

Método / Clase	`RoutesGenerator`
`getTemplate()`	Crea un nuevo `ModelsTemplate`
`getOutputConfigurationName()`	Recupera el atributo `ROUTES` de la clase `AnchurusLaravelGenerator`
`getFileName(T)`	T es una lista de controladores. Da nombre al archivo del modelo generado bajo el estándar de PHP.

Tabla 9 Clase `RoutesGenerator`

Método / Clase	`PagesGenerator`
getTemplate()	Crea un nuevo `PagesTemplate`
`getOutputConfigurationName()`	Recupera el atributo `PAGES` de la clase `AnchurusLaravelGenerator`

`getFileName(T)`	T es una página. Da nombre al archivo de la página generada, con el nombre de la página de ISML.

Tabla 10 Clase `PagesGenerator`

Método / Clase	`MigrationsGenerator`
`getTemplate()`	Crea un nuevo `MigrationsTemplate`
`getOutputConfigurationName()`	Recupera el atributo `MIGRATIONS` de la clase `AnchurusLaravelGenerator`
`getFileName(T)`	T es una entidad. Da nombre al archivo de migraciones, con una fecha y una hora, y el nombre de la entidad de ISML.

Tabla 11 Clase `MigrationsGenerator`

3.4 Paquete `co.edu.javeriana.anchurus.generator.laravel.templates`

La siguiente es la estructura general del paquete `laravel.templates`:

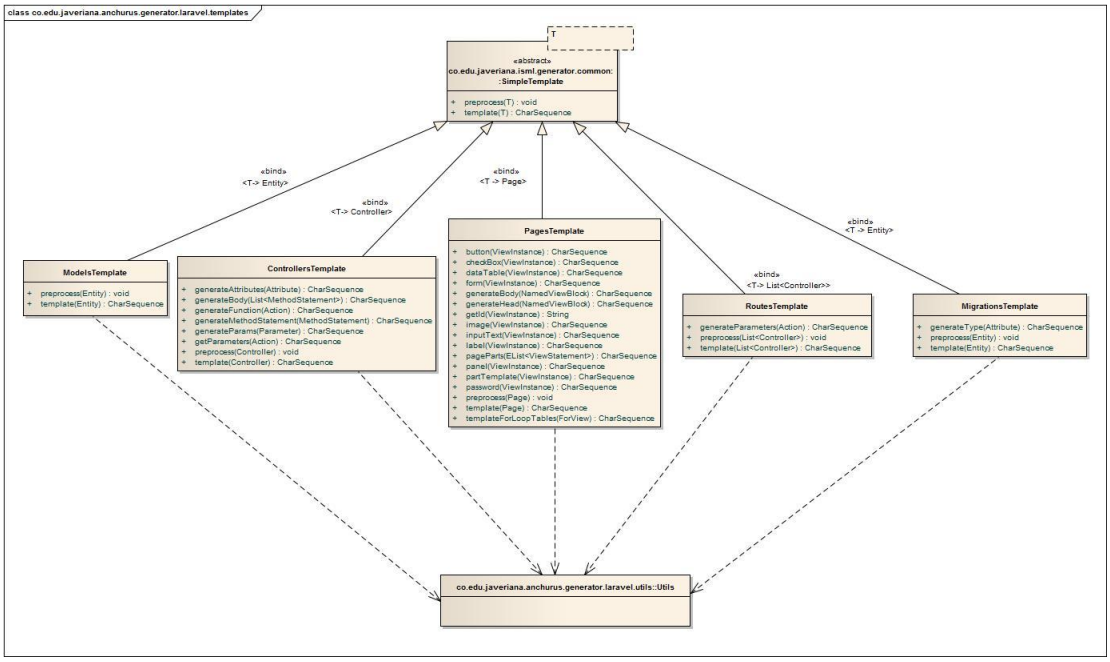


Ilustración 9 Paquete de templates

Este paquete tiene cinco clases que heredan de la clase *abstracta* `SimpleTemplate` del paquete `co.edu.javeriana.isml.common`, que se explicó más atrás. Cada una de estas clases retorna en su método `template()` una secuencia de caracteres (`CharSequence`) que es el cuerpo del objeto en PHP que está generando. Reciben cosas diferentes según la clase: entidades en el caso de `ModelsTemplate` y `MigrationsTemplate`, páginas en el caso de `PagesTemplate`, controladores en el caso de `ControllersTemplate` y una lista de controladores en el caso de `RoutesTemplate`.

Las clases contienen también en común un método `preprocess()` que recibe lo mismo que `template()`, cuya función es ejecutar acciones previas a la generación de código que tendrá lugar en el segundo método. Este método únicamente es útil donde se involucran controladores, en `ControllersTemplate` y en `RoutesTemplate`, donde ayuda llenando listas extra que se utilizarán a posteriori en el código.

Cada clase da cuenta de su función en el nombre, ser una plantilla del código que se va a generar. Explicación sobre cada clase se encuentra en las siguientes tablas:

Clase	**ModelsTemplate**

Descripción	La clase que tiene la capacidad de traducir las entidades ISML a modelos en PHP.
Método	**Descripción**
`void preprocess(Entity)`	No tiene ninguna utilidad este método. Se declara porque hereda de la clase abstracta.
`CharSequence template(Entity)`	Este método genera el archivo de modelo PHP para la entidad que se le pasa por parámetro.

Tabla 12 Clase `ModelsTemplate`

Clase	**ControllersTemplate**
Descripción	Esta clase tiene la capacidad de traducir los controladores en ISML a sus contrapartes en PHP.
Método	**Descripción**
`void preprocess(Controller)`	Este método llena dos listas al mismo tiempo que permiten, a posteriori, ser utilizadas en el método `template()` para llenar las sentencias `use` de la cabecera del controlador.
`CharSequence template(Controller)`	Este método genera el archivo de controlador PHP para el controlador que se le pasa por parámetro.
`CharSequence generateAttributes(Attribute)`	Este método convierte la declaración de atributo de ISML pasado por parámetro a su equivalente en formato PHP.
`CharSequence generateFunction(Action)`	Este método genera el código PHP de una función, desde la acción de ISML pasada por parámetro.

`CharSequence generateBody(List<MethodStatement>)`	Este método genera el cuerpo de la función de PHP, desde una lista de sentencias de método de ISML.
`CharSequence generateMethodStatement(MethodStatement)`	En realidad no es un solo método sino doce métodos distintos con la marca `dispatch`, que generan las distintas partes de una función en PHP, a saber: sentencias `if`, asignaciones, retornos, ciclos (`for` (en PHP serían `foreach`) y `while`), llamado a otros métodos, tipos, referencia a variables y retorno a páginas (vistas).
`CharSequence generateParams(Parameter)`	Este método genera formato para el parámetro dado, distinguiendo entre parámetros que sean Entidades de ISML y otro tipo de parámetros.
`CharSequence getParameters(Action)`	Este método genera una cadena separada por comas, que son los parámetros formales del método en PHP, desde una acción de ISML pasada por parámetro.

Tabla 13 Clase ControllersTemplate

Clase	**PagesTemplate**
Descripción	Esta clase tiene la capacidad de traducir una página de ISML a su equivalente en PHP.
Método	**Descripción**
`void preprocess(Page)`	No tiene ninguna utilidad este método. Se declara porque hereda de la clase abstracta.
`CharSequence template(Page)`	Este método genera el archivo de la página PHP desde la página ISML pasada por parámetro.

`CharSequence pageParts(EList<ViewStatement>)`	Este método delega a otro la generación de cada elemento pasado en la lista de partes pasada por parámetro.
`CharSequence plantillaParte(ViewInstance)`	Este método despacha a 12 métodos diferentes que son los que generan cada widget de la página, el que recibe una `IfView` retorna una sentencia `if` con formato de Blade, el que recibe una `ForView` hace lo propio con los `for` con formato de Blade e igualmente con las `Reference`.
`CharSequence label(ViewInstance)`	Este método convierte la `ViewInstance (label)` pasada como parámetro en un `Label` con formato de Blade.
`CharSequence getId(ViewInstance)`	Este método genera un identificador único para la `ViewInstance` pasada como parámetro.
`CharSequence inputText(ViewInstance)`	Este método convierte la `ViewInstance (Text)` pasada como parámetro en un `InputText` con formato de Blade.
`CharSequence button(ViewInstance)`	Este método convierte la `ViewInstance (Button)` pasada como parámetro en un Button con formato de Blade.
`CharSequence form(ViewInstance)`	Este método convierte la `ViewInstance (form)` pasada como parámetro en un `Form` con formato de Blade.
`CharSequence panel(ViewInstance)`	Este método convierte la `ViewInstance (Panel)` pasada como parámetro en un `div` con formato HTML.
`CharSequence dataTable(ViewInstance)`	Este método convierte la `ViewInstance (DataTable)` pasada como parámetro en una tabla con formato HTML.
`CharSequence genera-`	Este método genera la primera fila de la

`teHead(NamedViewBlock)`	tabla.
`CharSequence generateBody(NamedViewBlock)`	Este método delega a otro método la generación del cuerpo de la tabla.
`CharSequence templateForLoopTables(ForView)`	Este método genera las celdas que no son de primera fila de la tabla (el cuerpo de la tabla).
`CharSequence password(ViewInstance)`	Este método convierte la `ViewInstance (Password)` pasada como parámetro en un `Password` con formato de Blade.
`CharSequence checkBox(ViewInstance)`	Este método convierte la `ViewInstance (CheckBox)` pasada como parámetro en una `CheckBox` con formato de Blade.
`CharSequence image(ViewInstance)`	Este método convierte la `ViewInstance (Image)` pasada como parámetro en una `Image` con formato de Blade.

Tabla 14 Clase `PagesTemplate`

Clase	**RoutesTemplate**
Descripción	Esta clase tiene la capacidad de traducir una lista de controladores al archivo de rutas de PHP.
Método	**Descripción**
`void preprocess(List<Controller>)`	Este método llena dos listas al mismo tiempo que permiten, a posteriori, ser utilizadas en el método `template()`.
`CharSequence template(List<Controller>)`	Este método genera el archivo de rutas PHP para la lista de controladores que se le pasa por parámetro.
`CharSequence generateParame-`	Este método formatea la lista de parámetros de una acción de tal manera que puedan

`ters(Action)`	ponerse en la URL relativa para esa acción.

Tabla 15 Clase `RoutesTemplate`

Clase	**MigrationsTemplate**
Descripción	Esta clase tiene la capacidad de traducir una entidad a una migración de PHP.
Método	**Descripción**
`void preprocess(Entity)`	No tiene ninguna utilidad este método. Se declara porque hereda de la clase abstracta.
`CharSequence template(Entity)`	Este método genera el archivo de la migración de PHP desde la entidad ISML pasada por parámetro.
`CharSequence generateType(Attribute)`	Este método genera cada atributo para la tabla que será hecha en la migración.

Tabla 16 Clase `MigrationsTemplate`

3.5 Paquete `co.edu.javeriana.anchurus.generator.laravel.utils`

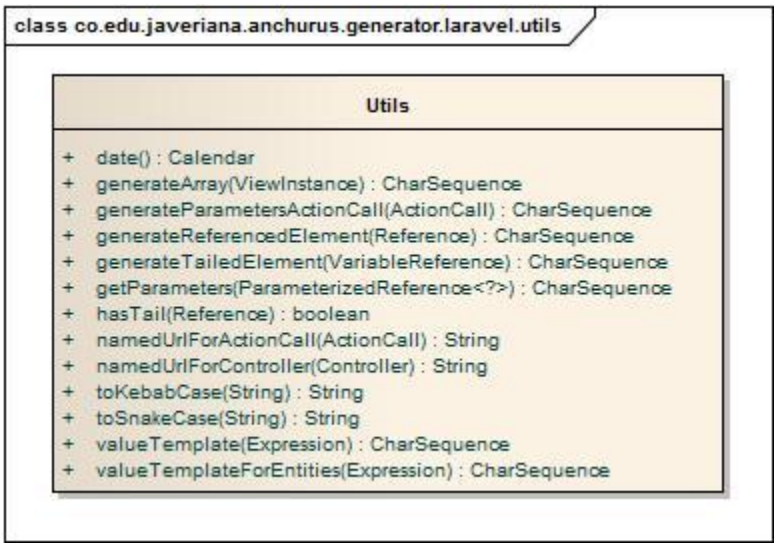


Ilustración 10 Clase `Utils`

El paquete `laravel.utils` posee únicamente esta clase, que se divide en trece métodos, que sirven a cualquiera de las clases de los dos paquetes anteriores.

Método	Descripción
`Calendar date()`	Devuelve la instancia actual del calendario (en otras palabras, obtiene la fecha y la hora actual).
`CharSequence generateArray(ViewInstance)`	Este método recibe una instancia de vista de ISML y devuelve un arreglo construido con el formato de PHP, con el fin de ser utilizado en el método `valueTemplate` en el caso donde la expresión ISML sea una `ViewInstance`.
`CharSequence generateParametersActionCall(ActionCall)`	Este método genera los parámetros reales del llamado a acción pasado por parámetro.
`CharSequence generateReferencedElement(Reference)`	Este método genera los elementos referenciados de la referencia pa-

	sada por parámetro.
`CharSequence generateTailedElement(VariableReference)`	Este método genera la referencia pasada por parámetro con su cola (de manera recursiva, si la cola también tiene cola, se va a generar también).
`CharSequence getParameters(ParameterizedReference<?>)`	Este método genera también parámetros reales, pero esta vez es de referencias parametrizadas que pueden ser de cualquier clase.
`boolean hasTail(Reference)`	Pregunta si una referencia dada tiene cola o no (devuelve verdadero o falso en uno u otro caso).
`String namedUrlForActionCall(ActionCall)`	Este método genera la URL relativa de un llamado a acción pasado por parámetro.
`String namedUrlForController(Controller)`	Este método genera un nombre para el controlador con tal de que sea compatible con URLs del sistema de rutas de la aplicación (convirtiendo el nombre del controlador a Kebab Case desde Camel Case mayúscula).
`String toKebabCase(String)`	Convierte la cadena en formato Camel Case minúscula a Kebab Case minúscula, ayudándose de un método estático de Google Case Format.
`String toSnakeCase(String)`	Convierte la cadena en formato Camel Case minúscula a Snake Case minúscula, ayudándose de un método estático de Google Case Format.
`CharSequence valueTemplate(Expression)`	Convierte la expresión ISML pasada por parámetro a una u otra

	cadena, según su tipo.
`CharSequence valueTemplateForEntities(Expression)`	Versión mejorada del método `valueTemplate` con un formato diferente de retorno si la expresión es un elemento referenciado y encima este elemento es una entidad.

Tabla 17 Métodos de la clase `Utils`

V – DESARROLLO DE LA SOLUCIÓN

Esta sección se dividirá en tres partes, en la primera se explicará cómo se aplicó la metodología DAD (de la sección Metodología), en la segunda se mencionará qué estándares se utilizaron para el correcto desarrollo de este proyecto y en la tercera se mostrará lo logrado con la aplicación, con entradas y salidas.

1. Aplicación de la metodología DAD

DAD en el proyecto se aplicó de la siguiente manera: en la fase de concepción, se delimitó determinada cantidad de ítems de trabajo que fueron definidos mediante Kanban [74], un método de trabajo proveniente de Japón, cuyo centro de atención es un tablero y una cantidad de tarjetas (conocidas como tarjetas Kanban). Dicho tablero se puede dividir en secciones para visualizar en qué punto se encuentra el progreso de una o más tareas indicadas en las tarjetas (por ejemplo, si no se ha hecho, si se está haciendo o si ya se entregó). En el caso de este proyecto, se utilizó una herramienta conocida como Trello [75] que utiliza Kanban.

Las reuniones durante el proyecto (fase de construcción) fueron semanales, donde se priorizaban los trabajos a realizar durante los 7 días siguientes a la reunión, para que, a la hora de la siguiente reunión, se revisaba el progreso realizado en el trabajo. Concluidas esas tareas, se asignaban otras para la siguiente semana, y así sucesivamente hasta que finalizó el total de las tareas asignadas inicialmente.

2. Estándares

Se utilizó un total de tres estándares para la ejecución del proyecto, dos para efectos de documentación y otro para el desarrollo. Para la construcción del SRS, el proyecto se acogió al estándar ISO/IEC/IEEE 29148:2011 [76] y para la construcción del SDD, el proyecto se acogió al estándar IEEE 1016:2009 [77], esto para los estándares de documentación. El estándar de desarrollo utilizado fue el MOF de OMG [26], para efectos de la especificación del lenguaje EMF para los metamodelos, concretamente el de ISML.

3. Funcionalidades de la aplicación

Lo que se mostrará a continuación son capturas de pantalla de la aplicación. Se muestra el modelo ISML en la aplicación y el código PHP generado luego de limpiar el proyecto o guardar el archivo, y en el caso de la página se muestra qué fue lo que generó. Nótese que las capturas de pantalla de Eclipse son del editor de ISML y se muestran para ilustrar la funcionalidad del generador.

3.1. Generación de código desde página ISML a página PHP

En la Ilustración 11 se muestra la página ISML `ViewDieta`, la cual se tradujo a la página view_dieta.php en la Ilustración 12 y se puede ver funcionando en la Ilustración 13. El bloque `Form` se mapeó al método de PHP `Form::open()`, el bloque Panel se tradujo a `<div>` de HTML, los métodos `Label()` se tradujeron a los métodos de PHP `Form::label()` y el método `Button()` se tradujo a un `<input type="submit">` de HTML.

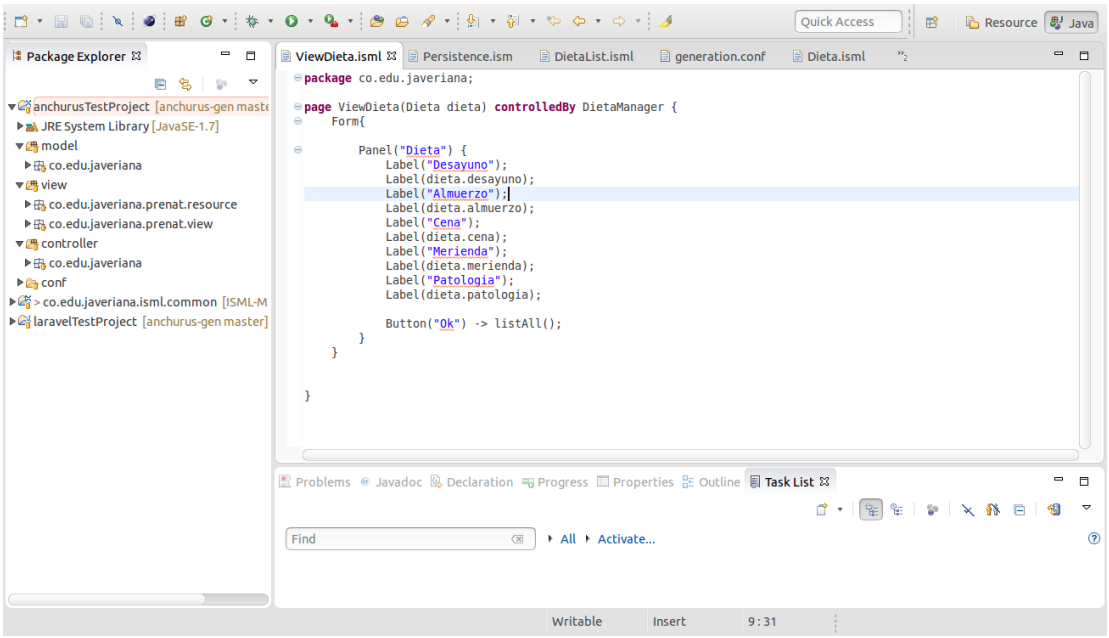


Ilustración 11 Página ISML ViewDieta dentro de Eclipse

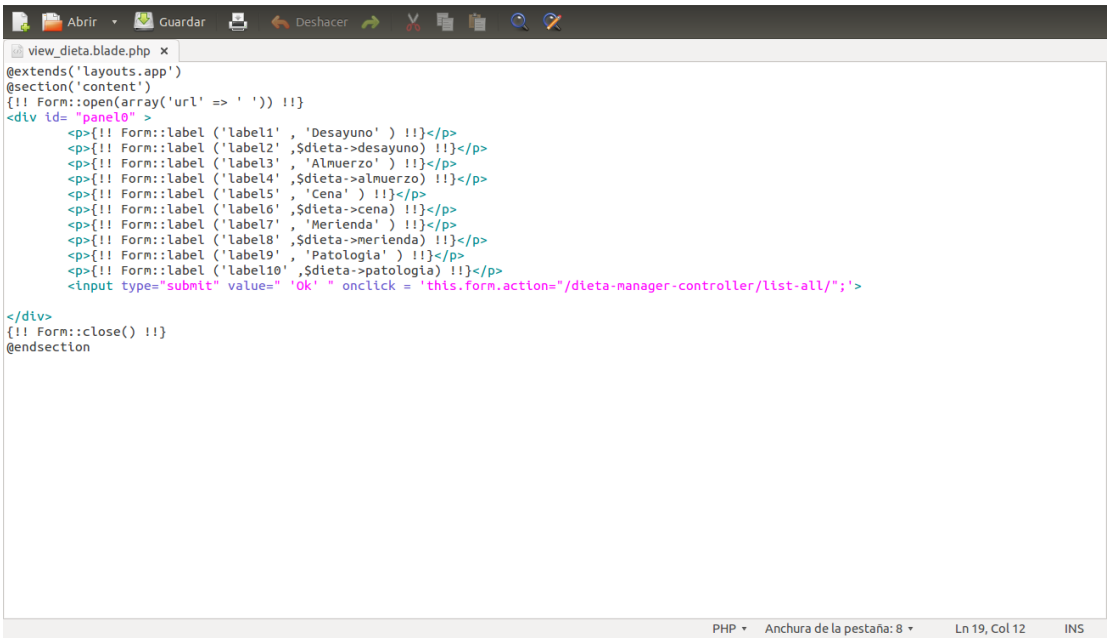


Ilustración 12 Archivo view_dieta.blade.php, resultado de la generación de código

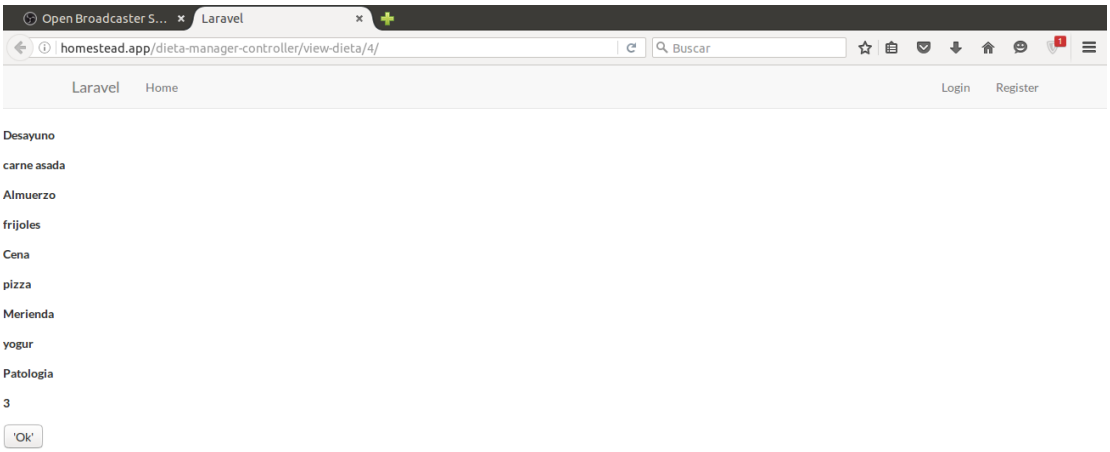


Ilustración 13 Probando el archivo anteriormente generado en un navegador web, luego de haberse montado Vagrant.

3.2. Generación de código desde controlador ISML a controlador PHP y archivo de rutas

En la Ilustración 14 se muestra parte del controlador ISML `DietaManager`, en la Ilustración 15 se muestra el controlador DietaManagerController.php y en la Ilustración 16 se muestra el archivo de rutas routes.php.

Cuando se convirtió el controlador ISML al controlador PHP, cada método del controlador ISML se mapeó a una función pública en el controlador PHP y los atributos recibidos por parámetro se convirtieron en variables locales dentro de la función, a las cuales se les asignó el valor `NULL`. Las sentencias `show` se convirtieron, en el archivo generado, en sentencias `return` que devolvían métodos `view()`, que a su misma vez, contenían la página a donde se debían remitir dichos métodos.

Cuando se convierte un conjunto de controladores ISML hacia el archivo de rutas, cada método se mapea hacia una posible ruta de la página, y cada parámetro recibido se convierte en parámetro de la URL relativa generada.

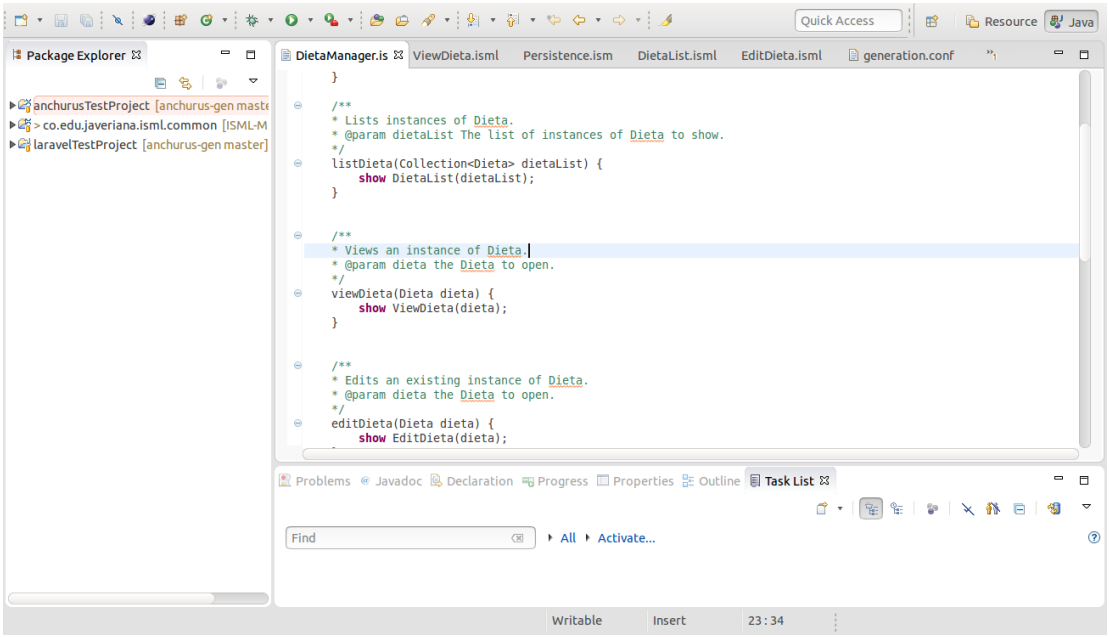
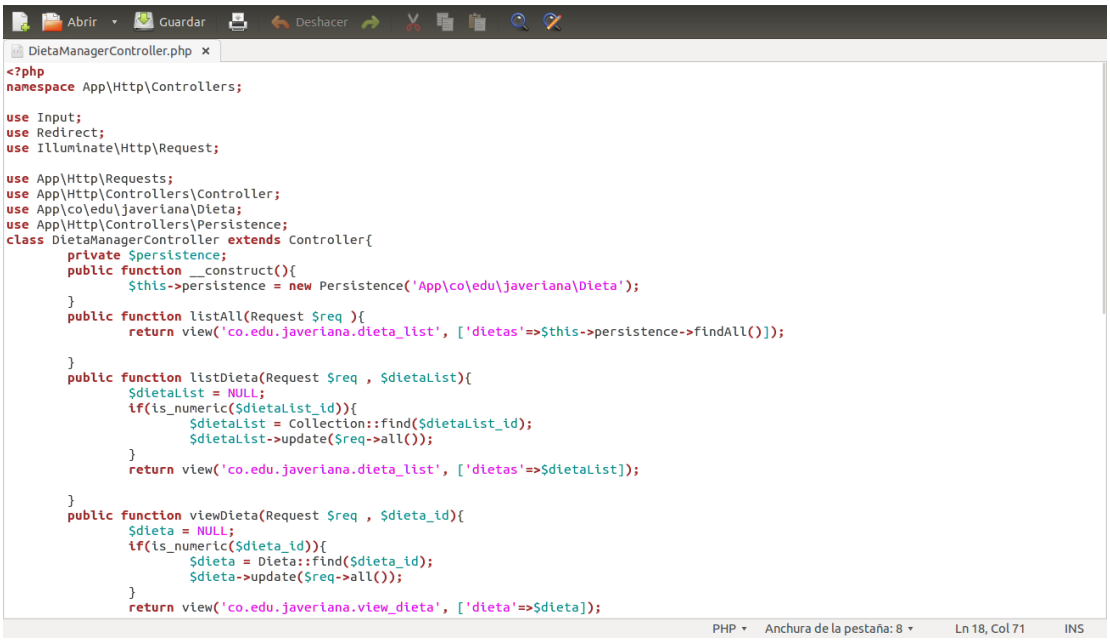
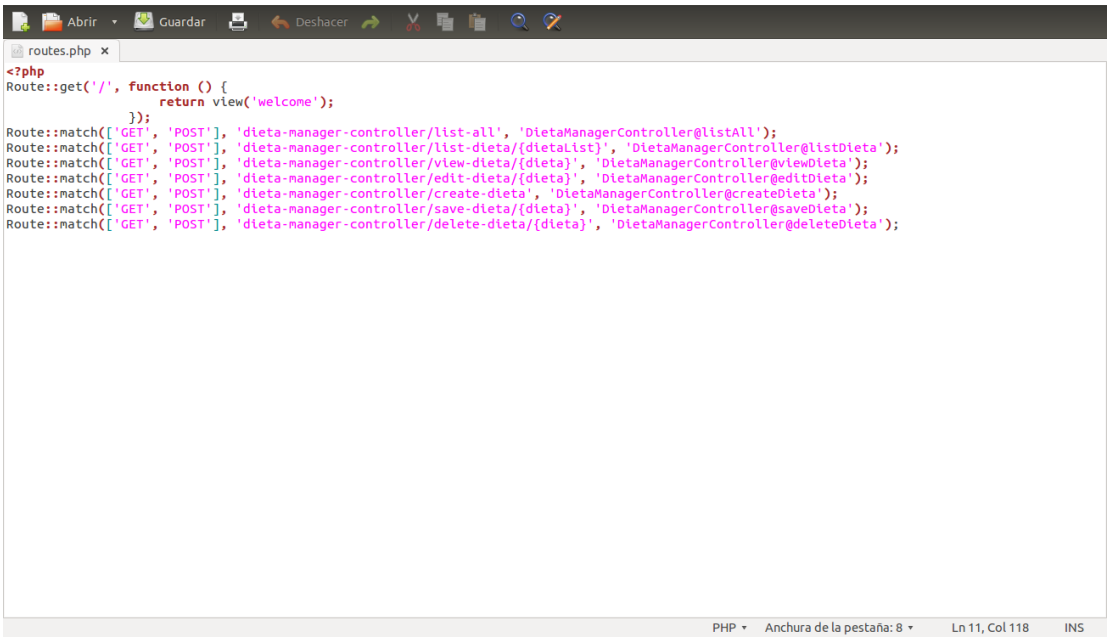


Ilustración 14 Parte del controlador ISML DietaManager dentro de Eclipse




```php
<?php
namespace App\Http\Controllers;

use Input;
use Redirect;
use Illuminate\Http\Request;

use App\Http\Requests;
use App\Http\Controllers\Controller;
use App\co\edu\javeriana\Dieta;
use App\Http\Controllers\Persistence;
class DietaManagerController extends Controller{
        private $persistence;
        public function __construct(){
                $this->persistence = new Persistence('App\co\edu\javeriana\Dieta');
        }
        public function listAll(Request $req ){
                return view('co.edu.javeriana.dieta_list', ['dietas'=>$this->persistence->findAll()]);

        }
        public function listDieta(Request $req , $dietaList){
                $dietaList = NULL;
                if(is_numeric($dietaList_id)){
                        $dietaList = Collection::find($dietaList_id);
                        $dietaList->update($req->all());
                }
                return view('co.edu.javeriana.dieta_list', ['dietas'=>$dietaList]);

        }
        public function viewDieta(Request $req , $dieta_id){
                $dieta = NULL;
                if(is_numeric($dieta_id)){
                        $dieta = Dieta::find($dieta_id);
                        $dieta->update($req->all());
                }
                return view('co.edu.javeriana.view_dieta', ['dieta'=>$dieta]);
```

Ilustración 15 Parte del controlador DietaManagerController.php después de ser generado




```php
<?php
Route::get('/', function () {
                return view('welcome');
                });
Route::match(['GET', 'POST'], 'dieta-manager-controller/list-all', 'DietaManagerController@listAll');
Route::match(['GET', 'POST'], 'dieta-manager-controller/list-dieta/{dietaList}', 'DietaManagerController@listDieta');
Route::match(['GET', 'POST'], 'dieta-manager-controller/view-dieta/{dieta}', 'DietaManagerController@viewDieta');
Route::match(['GET', 'POST'], 'dieta-manager-controller/edit-dieta/{dieta}', 'DietaManagerController@editDieta');
Route::match(['GET', 'POST'], 'dieta-manager-controller/create-dieta', 'DietaManagerController@createDieta');
Route::match(['GET', 'POST'], 'dieta-manager-controller/save-dieta/{dieta}', 'DietaManagerController@saveDieta');
Route::match(['GET', 'POST'], 'dieta-manager-controller/delete-dieta/{dieta}', 'DietaManagerController@deleteDieta');
```

Ilustración 16 Archivo de rutas luego de haber sido generado

3.3. Generación de código de entidad ISML a modelo y migración PHP

En la Ilustración 17 se muestra la entidad ISML `Dieta`, en la Ilustración 18 se muestra el modelo Dieta.php y en la Ilustración 19 se muestra el archivo de migración create_dietas_table.php. Cuando se genera un modelo a partir de la entidad, solo se crea un archivo de clase PHP que extiende de la clase `Model`, cuyo nombre es el mismo nombre de la entidad ISML de origen. Cuando se crea una migración a partir de la entidad, al nombre del archivo generado se le antecede la fecha y la hora en la cual fue generado dicho archivo. Dentro del archivo generado, cada atributo de la entidad desde donde se genera se mapea hacia métodos de formato `$table->tipo_atributo('nombre_atributo')->default(valor_por_defecto)` que se encuentran dentro de la función `Schema::create()`.

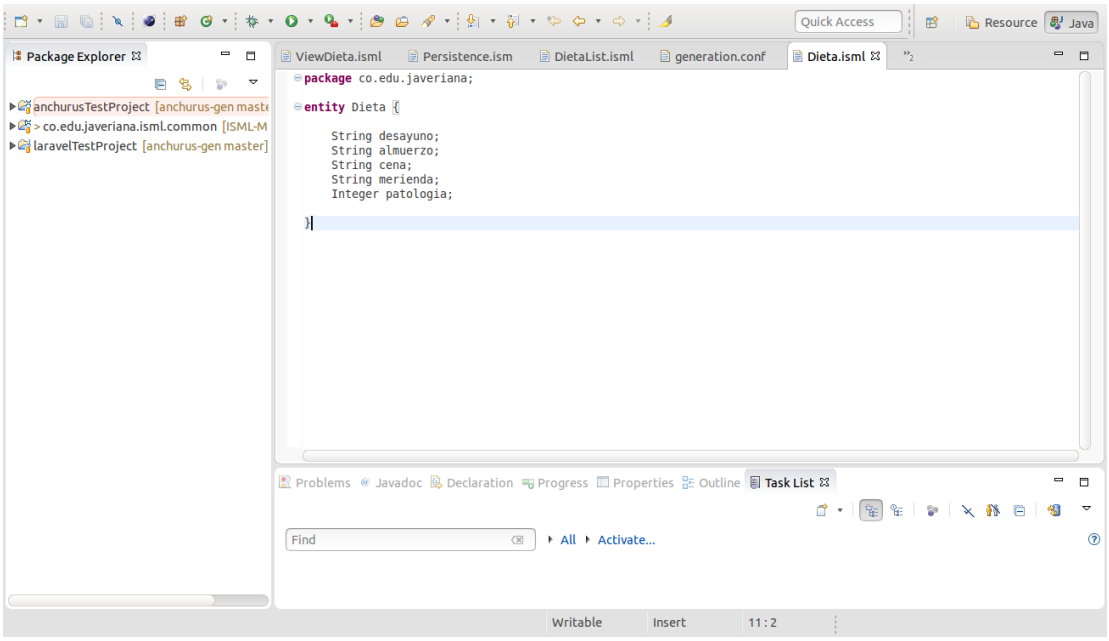


Ilustración 17 Entidad ISML Dieta en Eclipse

Abrir Guardar Deshacer

Dieta.php

```php
<?php
namespace App\co\edu\javeriana;
use Illuminate\Database\Eloquent\Model;

class Dieta extends Model{
        protected $guarded = [];

}
```

PHP Anchura de la pestaña: 8 Ln 10, Col 1 INS

Ilustración 18 Modelo Dieta.php luego de haber sido generado

Abrir Guardar Deshacer

2016_05_16_203106_c...ate_dietas_table.php

```php
<?php
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateDietasTable extends Migration{

        public function up(){
                Schema::create('dietas', function(Blueprint $table){
                        $table->increments('id');
                        $table->string('desayuno')->default('');
                        $table->string('almuerzo')->default('');
                        $table->string('cena')->default('');
                        $table->string('merienda')->default('');
                        $table->bigInteger('patologia')->default(0);
                        $table->timestamps();
                });

        }

        public function down(){
                DB::statement('SET FOREIGN_KEY_CHECKS = 0');
                Schema::drop('dietas');
                DB::statement('SET FOREIGN_KEY_CHECKS = 1');
        }

}
```

PHP Anchura de la pestaña: 8 Ln 1, Col 1 INS

Ilustración 19 Archivo de migración create_dietas_table.php con la fecha y la hora de su generación

3.4. Archivo de configuración

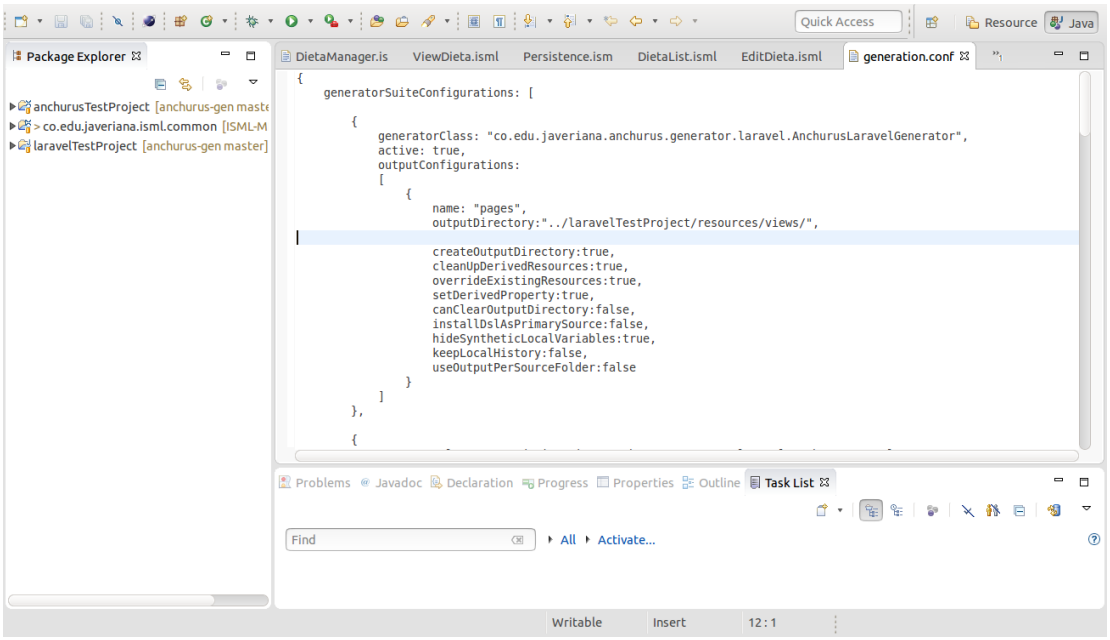


Ilustración 20 Archivo de configuración `generation.conf.json`

El anterior es un pantallazo de parte del archivo de configuración json que controla todas las generaciones de código. Aquí se pueden añadir o quitar bloques enteros de configuraciones (entiéndase como bloque lo que está entre paréntesis y coma al final) y estos bloques son los que permiten que cada elemento de la generación de código se ejecute de manera correcta.

VI – RESULTADOS

El plan de pruebas [78] contempló un total de cinco pruebas de sistema (esto es, que se puso todo el sistema en marcha para probar cada componente) que confirmarían o refutarían si el sistema Anchurus-GEN funcionaba correctamente en las funcionalidades especificadas en el documento de especificación de requerimientos [61]. Dichas pruebas las pasó el sistema a cabalidad, como se muestra a continuación en las siguientes tablas, que muestran con detalle cómo se ejecutó cada prueba:

Tabla 18 Caso de prueba 1, generación de páginas

Caso de prueba	1
Proyecto	Anchurus-GEN

Fecha de entrega	13/05/2016

Datos de entrada	Modelado de la página ViewDieta, que contiene un formulario, dentro de éste un panel y dentro de este panel, 10 etiquetas y un botón.
	`package co.edu.javeriana;`
	`page ViewDieta(Dieta dieta) controlledBy DietaManager {`
	`  Form{`
	`  Panel("Dieta") {`
	`  Label("Desayuno");`
	`  Label(dieta.desayuno);`
	`  Label("Almuerzo");`
	`  Label(dieta.almuerzo);`
	`  Label("Cena");`
	`  Label(dieta.cena);`
	`  Label("Merienda");`
	`  Label(dieta.merienda);`
	`  Label("Patologia");`
	`  Label(dieta.patologia);`
	`  Button("Ok") -> listAll();`
	`  }`
	`  }`
	`}`

Salida esperada	Página PHP view_dieta, con el formato Blade propio de Laravel.
	`@extends('layouts.app')`
	`@section('content')`
	`{!! Form::open(array('url' => ' ')) !!}`
	`<div id= "panel0" >`
	`<p>{!! Form::label ('label1' , 'Desayuno' ) !!}</p>`
	`<p>{!! Form::label ('label2' ,$dieta->desayuno) !!}</p>`
	`<p>{!! Form::label ('label3' , 'Almuerzo' ) !!}</p>`
	`<p>{!! Form::label ('label4' ,$dieta->almuerzo) !!}</p>`
	`<p>{!! Form::label ('label5' , 'Cena' ) !!}</p>`
	`<p>{!! Form::label ('label6' ,$dieta->cena) !!}</p>`
	`<p>{!! Form::label ('label7' , 'Merienda' ) !!}</p>`
	`<p>{!! Form::label ('label8' ,$dieta->merienda) !!}</p>`
	`<p>{!! Form::label ('label9' , 'Patologia' ) !!}</p>`
	`<p>{!! Form::label ('label10' ,$dieta->patologia) !!}</p>`
	`<input type="submit" value=" 'Ok' " onclick = 'this.form.action="/dieta-manager-controller/list-all/";'>`
	`</div>`
	`{!! Form::close() !!}`
	`@endsection`

Con-clusión	La prueba fue exitosa.

Tabla 19 Caso de prueba 2, generación de controladores

Caso de prueba	2
Proyecto	Anchurus-GEN
Fecha de entrega	13/05/2016
Datos de entrada	Modelado del controlador DietaManager, que contiene nueve métodos cuya función es hacer el CRUD de las dietas existentes sobre las páginas que contro-la.

```
package co.edu.javeriana;

controller DietaManager {
    has Persistence<Dieta> persistence;

    /**
    * Lists all instances of Dieta.
    */
    default listAll() {
        show DietaList(persistence.findAll());
    }

    /**
    * Lists instances of Dieta.
    * @param dietaList The list of instances of Dieta to show.
    */
    listDieta(Collection<Dieta> dietaList) {
        show DietaList(dietaList);
    }


    /**
    * Views an instance of Dieta.
    * @param dieta the Dieta to open.
    */
    viewDieta(Dieta dieta) {
        show ViewDieta(dieta);
    }


    /**
    * Edits an existing instance of Dieta.
    * @param dieta the Dieta to open.
    */
    editDieta(Dieta dieta) {
        show EditDieta(dieta);
    }

    /**
    * Creates a a new instance of Dieta.
    */
    createDieta() {
        show EditDieta(new Dieta);
    }


    /**
    * Saves an instance of Dieta.
    * @param dieta the Dieta to save.
    */
    saveDieta(Dieta dieta) {
    if(persistence.isPersistent(dieta)){
        persistence.save(dieta);
    } else {
        persistence.create(dieta);
    }
    -> listAll();
    }


    /**
    * Deletes an instance of Dieta.
    * @param dieta the Dieta to delete.
    */
    deleteDieta(Dieta dieta) {
        persistence.remove(dieta);
        -> listAll();
    }


}
```

Salida esperada	Controlador PHP DietaManagerController.

```php
<?php
namespace App\Http\Controllers;

use Input;
use Redirect;
use Illuminate\Http\Request;

use App\Http\Requests;
use App\Http\Controllers\Controller;
use App\co\edu\javeriana\Dieta;
use App\Http\Controllers\Persistence;
class DietaManagerController extends Controller{
        private $persistence;
        public function __construct(){
                $this->persistence = new Persistence('App\co\edu\javeriana\Dieta');
        }
        public function listAll(Request $req ){
                return view('co.edu.javeriana.dieta_list', ['dietas'=>$this->persistence->findAll()]);

        }
        public function listDieta(Request $req , $dietaList){
                $dietaList = NULL;
                if(is_numeric($dietaList_id)){
                        $dietaList = Collection::find($dietaList_id);
                        $dietaList->update($req->all());
                }
                return view('co.edu.javeriana.dieta_list', ['dietas'=>$dietaList]);

        }
        public function viewDieta(Request $req , $dieta_id){
                $dieta = NULL;
                if(is_numeric($dieta_id)){
                        $dieta = Dieta::find($dieta_id);
                        $dieta->update($req->all());
                }
                return view('co.edu.javeriana.view_dieta', ['dieta'=>$dieta]);

        }
```

```
public function editDieta(Request $req , $dieta_id){
        $dieta = NULL;
        if(is_numeric($dieta_id)){
                $dieta = Dieta::find($dieta_id);
                $dieta->update($req->all());
        }
        return view('co.edu.javeriana.edit_dieta', ['dieta'=>$dieta]);

}
public function createDieta(Request $req ){
        return view('co.edu.javeriana.edit_dieta', ['dieta'=>new Dieta]);

}
public function saveDieta(Request $req , $dieta_id){
        $dieta = NULL;
        if(is_numeric($dieta_id)){
                $dieta = Dieta::find($dieta_id);
                $dieta->update($req->all());
        }
        if($this->persistence->isPersistent($dieta)){
                $this->persistence->save($dieta);

        }
        else{
                $this->persistence->create($dieta);

        }
        return $this->listAll($req);

}
public function deleteDieta(Request $req , $dieta_id){
        $dieta = NULL;
        if(is_numeric($dieta_id)){
                $dieta = Dieta::find($dieta_id);
                $dieta->update($req->all());
        }
        $this->persistence->remove($dieta);

        return $this->listAll($req);

          }
 }
```

Conclusión	La prueba fue exitosa.

Tabla 20 Caso de prueba 3, generación del archivo de rutas

Caso de prueba	3
Proyecto	Anchurus-GEN
Fecha de entrega	13/05/2016
Datos de entrada	Modelado del controlador DietaManager, que contiene nueve métodos cuya función es hacer el CRUD de las dietas existentes sobre las páginas que controla.

```
package co.edu.javeriana;

controller DietaManager {
    has Persistence<Dieta> persistence;

    /**
    * Lists all instances of Dieta.
    */
    default listAll() {
        show DietaList(persistence.findAll());
    }

    /**
    * Lists instances of Dieta.
    * @param dietaList The list of instances of Dieta to show.
    */
    listDieta(Collection<Dieta> dietaList) {
        show DietaList(dietaList);
    }


    /**
    * Views an instance of Dieta.
    * @param dieta the Dieta to open.
    */
    viewDieta(Dieta dieta) {
        show ViewDieta(dieta);
    }


    /**
    * Edits an existing instance of Dieta.
    * @param dieta the Dieta to open.
    */
    editDieta(Dieta dieta) {
        show EditDieta(dieta);
    }

    /**
    * Creates a a new instance of Dieta.
    */
    createDieta() {
        show EditDieta(new Dieta);
    }


    /**
    * Saves an instance of Dieta.
    * @param dieta the Dieta to save.
    */
    saveDieta(Dieta dieta) {
    if(persistence.isPersistent(dieta)){
        persistence.save(dieta);
    } else {
        persistence.create(dieta);
    }
    -> listAll();
    }


    /**
    * Deletes an instance of Dieta.
    * @param dieta the Dieta to delete.
    */
    deleteDieta(Dieta dieta) {
        persistence.remove(dieta);
        -> listAll();
    }


}
```

Salida esperada

Archivo de rutas de la aplicación web.

```
<?php
Route::get('/', function () {
                    return view('welcome');
                });
Route::match(['GET', 'POST'], 'dieta-manager-controller/list-all', 'DietaManagerController@listAll');
Route::match(['GET', 'POST'], 'dieta-manager-controller/list-dieta/{dietaList}', 'DietaManagerController@listDieta');
Route::match(['GET', 'POST'], 'dieta-manager-controller/view-dieta/{dieta}', 'DietaManagerController@viewDieta');
Route::match(['GET', 'POST'], 'dieta-manager-controller/edit-dieta/{dieta}', 'DietaManagerController@editDieta');
Route::match(['GET', 'POST'], 'dieta-manager-controller/create-dieta', 'DietaManagerController@createDieta');
Route::match(['GET', 'POST'], 'dieta-manager-controller/save-dieta/{dieta}', 'DietaManagerController@saveDieta');
Route::match(['GET', 'POST'], 'dieta-manager-controller/delete-dieta/{dieta}', 'DietaManagerController@deleteDieta');
```

Conclusión

La prueba fue exitosa.

Tabla 21 Caso de prueba 4, generación del archivo de migraciones

Caso de prueba	4
Proyecto	Anchurus-GEN
Fecha de entrega	13/05/2016

Datos de entrada	Modelado de la entidad Dieta, que contiene cuatro atributos de tipo String (desayuno, almuerzo, cena y merienda) y un atributo de tipo Integer (patología).
	`package co.edu.javeriana;`
	`entity Dieta {`
	`String desayuno;`
	`String almuerzo;`
	`String cena;`
	`String merienda;`
	`Integer patologia;`
	`}`

Salida esperada	Archivo de migraciones de PHP, cuyo nombre incluye la fecha y hora actuales.

	<?php
	use Illuminate\Database\Schema\Blueprint;
	use Illuminate\Database\Migrations\Migration;
	
	class CreateDietasTable extends Migration{
	
	public function up(){
	Schema::create('dietas', function(Blueprint $table){
	$table->increments('id');
	$table->string('desayuno')->default('');
	$table->string('almuerzo')->default('');
	$table->string('cena')->default('');
	$table->string('merienda')->default('');
	$table->bigInteger('patologia')->default(0);
	$table->timestamps();
	});
	
	}
	
	public function down(){
	DB::statement('SET FOREIGN_KEY_CHECKS = 0');
	Schema::drop('dietas');
	DB::statement('SET FOREIGN_KEY_CHECKS = 1');
	}
	
	}

Conclusión	La prueba fue exitosa.

Tabla 22 Caso de prueba 5, generación del archivo de modelo

Caso de prueba	5
Proyecto	Anchurus-GEN
Fecha de entrega	13/05/2016
Datos de	Modelado de la entidad Dieta, que contiene cuatro atributos de tipo String

entrada	(desayuno, almuerzo, cena y merienda) y un atributo de tipo Integer (patología).
	`package co.edu.javeriana;`
	`entity Dieta {`
	`String desayuno;`
	`String almuerzo;`
	`String cena;`
	`String merienda;`
	`Integer patologia;`
	`}`

Salida esperada	Archivo de modelo de PHP.
	`<?php`
	`namespace App\co\edu\javeriana;`
	`use Illuminate\Database\Eloquent\Model;`
	`class Dieta extends Model{`
	`protected $guarded = [];`
	`}`

Conclusión	La prueba fue exitosa.

Comparaciones entre código generado y modelos ISML

La proporción en líneas de código entre el modelo ISML y el código generado en PHP es el siguiente, según los elementos generados:

Tabla 23 Proporciones de líneas de código escritas

Tipo de archivo ISML / Características	Elemento (s) de salida PHP	Proporciones
Controlador	Controlador, archivo de rutas	31.13%
Entidad	Modelo, migración	19.6%
Página	Página	90.53%

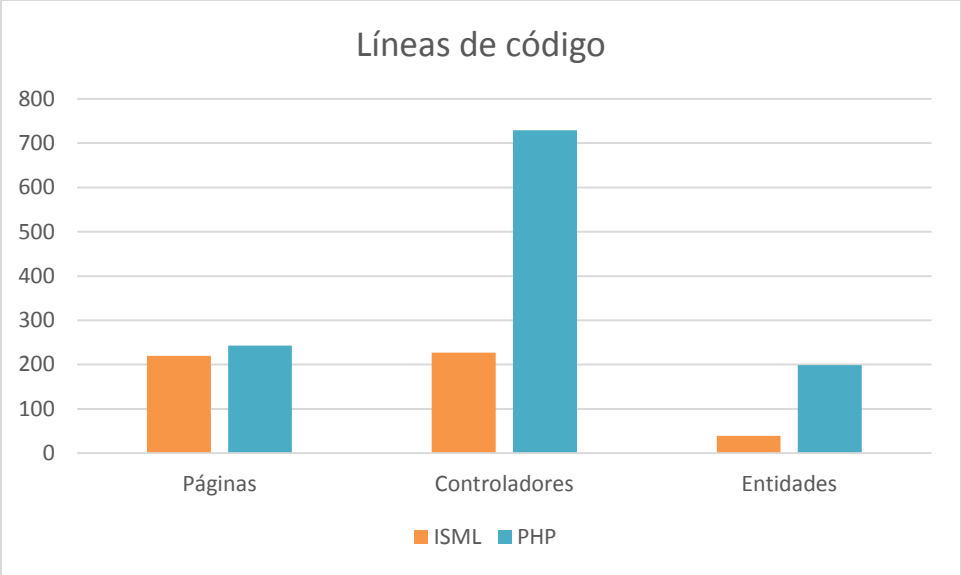


Ilustración 21 Gráfico de comparación entre las líneas de código generadas en PHP contra las líneas de código escritas en ISML.

Los porcentajes en la Tabla 23 indican el cociente entre las líneas de código escritas en los modelos contra las líneas de código generadas en PHP, pudiéndose interpretar como el porcentaje de tiempo utilizado cuando uno escribe modelos y después genera contra escribir la aplicación desde cero. En la Ilustración 21, se mostraron las mediciones concretas que derivaron en los porcentajes de la tabla. Se llegaron a estas medidas mediante los siguientes cálculos, utilizando una aplicación con 27 modelos ISML (5 controladores, 8 entidades y 14 páginas):

Ecuación 1 Proporción LDC para entidades

$$P_{ent} = \frac{\sum_{i=1}^{j} E_i}{\sum_{i=1}^{j} (Md_i + Mg_i)} \cdot 100\%$$

Para la Ecuación 1, P_{ent} es la proporción de líneas de código para entidades, j es la cantidad de entidades totales, E_i son las líneas de código que contiene una entidad ISML, Md_i las líneas de código que contiene un modelo de PHP y Mg_i, las líneas de código que contiene una migración.

Ecuación 2 Proporción LDC para controladores

$$P_{con} = \frac{\sum_{i=1}^{j} C_i}{\sum_{i=1}^{j} (R_i + G_i)} \cdot 100\%$$

Para la Ecuación 2, P_{con} es la proporción de líneas de código para controladores, j es la cantidad de controladores totales, C_i son las líneas de código que contiene un controlador ISML,

R_i las líneas de código generadas en el archivo de rutas por un solo controlador y G_i, las líneas de código que contiene un controlador generado.

Ecuación 3 Proporción LDC para páginas

$$P_{pag} = \frac{\sum_{i=1}^{j} P_i}{\sum_{i=1}^{j} G_i} \cdot 100\%$$

Para la Ecuación 3, P_{pag} es la proporción de líneas de código para páginas, j es la cantidad de páginas totales, P_i son las líneas de código que contiene una página ISML, y G_i, las líneas de código que contiene una página generada.

En el caso de las entidades, la proporción es del 19.6%, pudiendo ser incluso menor cuando se implementen las relaciones entre entidades dentro de Anchurus-GEN. Esta proporción indica que por cada línea de código de entidad en ISML, se generan cinco en PHP. En los controladores, esta proporción asciende al 31.13%, indicando que por cada línea de código de los controladores ISML se generan 3 en PHP. El caso de las páginas puede parecer desalentador, con una proporción del 90.53%, pero se espera que este porcentaje disminuya al implementarse los widgets faltantes. En el global (el cálculo se hace con la sumatoria de todas las LDC de ISML sobre todas las líneas de código generadas), la proporción es del 41.5%, que implica que por cada línea de código ISML se generan cerca de 2.4 en PHP, esto indica que la herramienta está haciendo bastante bien su trabajo, aun pudiendo ser mejorable.

VII – CONCLUSIONES

1 Análisis de Impacto del Desarrollo

Se espera que este proyecto impacte positivamente en la productividad de las empresas que tengan su base en el desarrollo de aplicaciones web, y también en eventuales reducciones de costos provenientes de las horas de desarrollo que se inviertan en la aplicación, que ya tendría parte del trabajo hecho con este generador funcionando.

En el orden disciplinar, se espera que la introducción de Anchurus-GEN permita que los ingenieros de software que desarrollen aplicaciones web cuenten con mayores opciones y de mejor calidad que lo existente para hacer su trabajo, y se espera que fuerce a los programadores a utilizar buenas prácticas de programación (como aplicar patrones tales como MVC).

Haciendo parte de una herramienta más grande, como lo es Eclipse, se espera que Anchurus-GEN sea utilizado en conjunto con otros plug-in que desee el ingeniero de software con la finalidad de contar con una gama más amplia de herramientas a la hora de acometer un proyecto de software que tenga que ver con web.

En el orden económico, se espera que las empresas que puedan utilizar este generador de código puedan reducir sus costos, producto de las menores horas de desarrollo que se puedan invertir cuando se acometa un proyecto de desarrollo que involucre una aplicación web.

2 Conclusiones y Trabajo Futuro

En esta memoria se especificó todo el proceso de desarrollo del sistema Anchurus-GEN, desde su especificación en cuanto a los requerimientos que debería tener para funcionar correctamente, hasta los casos de prueba que se ejecutaron sobre Anchurus-GEN. Se detalló el diseño del producto, se dio luz sobre el por qué elegir Anchurus-GEN sobre otras aplicaciones ya existentes y también se justificó con qué fin se creó el sistema.

Para el trabajo futuro se especificaron tres ítems, los cuales son: generar más widgets de las páginas, construir el generador de recursos y el generador de servicios.

Actualmente el generador de código soporta generación de etiquetas, cajas de texto, cajas de chequeo, cajas de contraseña, imágenes, tablas, formularios y paneles. ISML soporta estos widgets y otros más, como mapas de Google, listas de selección, enlaces, *spinners*, radiobotones, *comboboxes* y calendarios, por lo que se hace necesario implementar estos widgets faltantes, aunque en el generador de código están los esqueletos de los métodos generadores de estos widgets.

Actualmente el generador de código genera mensajes de texto únicamente en un idioma. Para tener soporte de múltiples idiomas, se necesita construir el generador de recursos. Un recurso es un archivo donde existe un conjunto de claves con su respectivo valor. En internacionalización, el valor es el texto que se mostrará en la página.

Actualmente al generador se le hizo una implementación para el servicio de persistencia, pero para mayor flexibilidad a la hora de crear otros, se necesita construir el generador de servicios para generar más lógica de negocio que pueda estar detrás de la página.

VIII- REFERENCIAS Y BIBLIOGRAFÍA

[1] ANIEL.es, «Programación web,» [En línea]. Available: http://www.aniel.es/desarrollo-web/programacion-web/. [Último acceso: 15 Noviembre 2015].

[2] Oracle, «Overview (Java (TM) EE 7 Specification APIs),» [En línea]. Available: https://docs.oracle.com/javaee/7/api/toc.htm. [Último acceso: 4 Noviembre 2015].

[3] Ç. Çivici, «PrimeFaces User Guide 5.3 First Edition,» [En línea]. Available: http://www.primefaces.org/docs/guide/primefaces_user_guide_5_3.pdf. [Último acceso: 4 Noviembre 2015].

[4] Université de Namur, «PReCISE - Model-driven Engineering,» [En línea]. Available: http://www.unamur.be/en/precise/research-fields/mde. [Último acceso: 4 Noviembre 2015].

[5] S. Kent, «Model Driven Engineering,» de *Proceedings of the Third International Conference on Integrated Formal Methods (IFM '02)*, M. J. Butler, L. Petre y K. Sere, Edits., Londres, Springer-Verlag, 2002, pp. 286-298.

[6] Real Academia Española, «Tándem,» 2014. [En línea]. Available: http://dle.rae.es/?w=t%C3%A1ndem&m=form&o=h. [Último acceso: 8 Noviembre 2015].

[7] L. M. C. Franky de Toro, J. A. Pavlich Mariscal, J. C. Olarte, M. C. Acero, A. Zambrano, J. L. Camargo y J. N. Pinzón, «ISML: Un Lenguaje y un Ambiente MDE para modelar y generar aplicaciones web con integración de componentes existentes,» *10° Congreso Colombiano de Computación,* 2015.

[8] E. Gamma, R. Helm, R. Johnson y J. Vlissides, «Design Patterns in Smalltalk MVC,» de *Design Patterns*, Mountain View, Addison-Wesley, 1994, pp. 14-16.

[9] J. C. Olarte Abello, ZOE-GEN: Un transformador para facilitar la generación de aplicaciones basado en modelos, Bogotá: Pontificia Universidad Javeriana, 2015.

[10 TIOBE Software, «TIOBE Index for October 2015,» Octubre 2015. [En línea]. Available: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html. [Último

] acceso: 4 Noviembre 2015].

[11] N. Diakopoulos y S. Cass, «Interactive: The Top Programming Languages 2015,» 20 Julio 2015. [En línea]. Available: http://spectrum.ieee.org/static/interactive-the-top-programming-languages-2015#index/2015/1/0/0/0/1/50/1/50/1/50/1/30/1/30/1/30/1/20/1/20/1/5/1/5/1/20/1/100/. [Último acceso: 8 Noviembre 2015].

[12] Django Software Foundation, «Django,» [En línea]. Available: https://www.djangoproject.com/. [Último acceso: 19 Noviembre 2015].

[13] Eclipse Foundation, «Eclipse,» Eclipse Foundation, 2015. [En línea]. Available: http://www.eclipse.org/home/. [Último acceso: 15 Noviembre 2015].

[14] S. W. Ambler, «Going Beyond Scrum,» 2013. [En línea]. Available: http://disciplinedagileconsortium.org/Resources/Documents/BeyondScrum.pdf. [Último acceso: 5 Noviembre 2015].

[15] F. Durán Muñoz, J. Troya Castilla y A. Vallecillo Moreno, «Desarrollo dirigido por modelos (MDA y MDE),» de *Desarrollo de software dirigido por modelos*, Barcelona, Fundació para l'Universitat Oberta de Catalunya, pp. 9-34.

[16] A. Rodrigues da Silva, «Model-driven engineering: A survey supported by the unified conceptual model,» *Computer Languages, Systems & Structures*, nº 43, pp. 139-155, 2015.

[17] Object Management Group, «About OMG (r),» 10 Agosto 2015. [En línea]. Available: http://www.omg.org/gettingstarted/gettingstartedindex.htm. [Último acceso: 19 Noviembre 2015].

[18] G. Génova, «What is a metamodel: the OMG's metamodeling infrastructure,» 14 Mayo 2009. [En línea]. Available: http://www.ie.inf.uc3m.es/ggenova/Warsaw/Part3.pdf. [Último acceso: 20 Noviembre 2015].

[19] Xtext, «Overview,» [En línea]. Available: https://eclipse.org/Xtext/documentation/index.html. [Último acceso: 9 Noviembre 2015].

[20] Object Management Group, «XML Metadata Interchange (XMI),» Junio 2015. [En línea]. Available: http://www.omg.org/spec/XMI/. [Último acceso: 15 Noviembre 2015].

[21] Eclipse, «Eclipse Modeling Framework (EMF),» [En línea]. Available: https://eclipse.org/modeling/emf/. [Último acceso: 9 Noviembre 2015].

[22] M. Rouse, «Integrated development environment (IDE) definition,» Febrero 2007. [En línea]. Available: http://searchsoftwarequality.techtarget.com/definition/integrated-development-environment. [Último acceso: 20 Noviembre 2015].

[23] M. Rouse, «OSGi (Open Service Gateway Initiative) definition,» Marzo 2011. [En línea]. Available: http://searchnetworking.techtarget.com/definition/OSGi. [Último acceso: 19 Noviembre 2015].

[24] Eclipse Foundation, «Graphical Modeling Framework FAQ,» 3 Abril 2013. [En línea]. Available: http://wiki.eclipse.org/Graphical_Modeling_Framework_FAQ. [Último acceso: 20 Noviembre 2015].

[25] Xtend, «Introduction,» [En línea]. Available: http://www.eclipse.org/xtend/documentation/index.html. [Último acceso: 9 Noviembre 2015].

[26] Object Management Group, «OMG's MetaObject Facility,» 8 Julio 2015. [En línea]. Available: http://www.omg.org/mof/. [Último acceso: 19 Noviembre 2015].

[27] Object Management Group, «MOF Model To Text Transformation,» Enero 2008. [En línea]. Available: http://www.omg.org/spec/MOFM2T/1.0/index.htm. [Último acceso: 19 Noviembre 2015].

[28] Obeo - Model Driven Company, «Acceleo,» [En línea]. Available: https://eclipse.org/acceleo/. [Último acceso: 19 Noviembre 2015].

[29] Apache Software Foundation, «Maven,» [En línea]. Available: https://maven.apache.org/. [Último acceso: 19 Noviembre 2015].

[30] Apache Software Foundation, «Apache Ant,» 2 Julio 2015. [En línea]. Available: http://ant.apache.org/. [Último acceso: 19 Noviembre 2015].

[31] PHP, «Prefacio - Manual de PHP,» [En línea]. Available: https://secure.php.net/manual/es/preface.php. [Último acceso: 10 Noviembre 2015].

[32] The GTK+ Team, «The GTK+ Project,» [En línea]. Available: http://www.gtk.org/. [Último acceso: 19 Noviembre 2015].

[33] Hewlett Packard, «HP-UX 11i v3: Mission-Critical UNIX Operating System,» [En línea]. Available: http://www8.hp.com/us/en/products/servers/hp-ux.html. [Último acceso: 19 Noviembre 2015].

[34] OpenBSD, «OpenBSD,» 18 Octubre 2015. [En línea]. Available: http://www.openbsd.org/. [Último acceso: 19 Noviembre 2015].

[35] W3Schools, «HTML and XHTML,» [En línea]. Available: http://www.w3schools.com/html/html_xhtml.asp. [Último acceso: 19 Noviembre 2015].

[36] W3Schools, «XML Tutorial,» [En línea]. Available: http://www.w3schools.com/xml/. [Último acceso: 19 Noviembre 2015].

[37] PHP, «Introducción a PDO,» [En línea]. Available: http://php.net/manual/es/intro.pdo.php. [Último acceso: 19 Noviembre 2015].

[38] Microsoft, «ODBC: Conectividad abierta de bases de datos,» [En línea]. Available: https://support.microsoft.com/es-es/kb/110093. [Último acceso: 19 Noviembre 2015].

[39] MIT, «Protocolo ligero de acceso a directorios,» [En línea]. Available: http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-es-4/ch-ldap.html. [Último acceso: 19 Noviembre 2015].

[40] M. Crispin, «Request for Comments 3501: Internet Message Access Protocol - Version 4rev1,» Marzo 2003. [En línea]. Available: http://www.ietf.org/rfc/rfc3501.txt. [Último acceso: 19 Noviembre 2015].

[41] J. Case, M. Fedor y M. L. Schoffstall, «A Simple Network Management Protocol,» Mayo 1990. [En línea]. Available: http://www.ietf.org/rfc/rfc1157.txt. [Último acceso: 19 Noviembre 2015].

[42] B. Skvorc, «The Best PHP Framework for 2015: SitePoint Survey Results,» 30 Marzo 2015. [En línea]. Available: http://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/. [Último acceso: 9 Noviembre 2015].

[43] Laravel, «Introduction,» [En línea]. Available: http://laravel.com/docs/4.2/introduction. [Último acceso: 10 Noviembre 2015].

[44] Ruby on Rails, «Ruby on Rails,» [En línea]. Available: http://rubyonrails.org/. [Último acceso: 19 Noviembre 2015].

[45] Microsoft, «ASP.NET MVC,» [En línea]. Available: http://www.asp.net/mvc. [Último acceso: 19 Noviembre 2015].

[46] Sinatra, «Sinatra,» [En línea]. Available: http://www.sinatrarb.com/intro-es.html. [Último acceso: 19 Noviembre 2015].

[47] O. Yanes Enríquez y H. Gracia del Busto, «Mapeo Objeto/Relacional (ORM),» *Telemática,* vol. 10, nº 3, pp. 1-7, 2011.

[48] M. Fowler, «Active Record,» de *Patterns of Enterprise Application Architecture*, Addison-Wesley, 2003, pp. 160-163.

[49] Symfony, «What is Symfony,» [En línea]. Available: http://symfony.com/what-is-symfony. [Último acceso: 16 Noviembre 2015].

[50] YAML, «YAML: YAML Ain't Markup Language,» 20 Noviembre 2011. [En línea]. Available: http://yaml.org/. [Último acceso: 17 Noviembre 2011].

[51] Nette, «Documentation,» [En línea]. Available: https://doc.nette.org/en/2.3/. [Último acceso: 17 Noviembre 2015].

[52] xlinesoft, «PHPrunner. The best PHP code generator in the world.,» 8 Septiembre 2015. [En línea]. Available: http://xlinesoft.com/phprunner/index.htm. [Último acceso: 18 Noviembre 2015].

[53] HKV Store, «PHPMaker,» 9 Noviembre 2015. [En línea]. Available: http://www.hkvstore.com/phpmaker/. [Último acceso: 18 Noviembre 2015].

[54] Bootstrap, «Bootstrap,» [En línea]. Available: http://getbootstrap.com/. [Último acceso: 18 Noviembre 2015].

[55] ScriptCase, «ScriptCase,» [En línea]. Available: http://www.scriptcase.net/. [Último acceso: 18 Noviembre 2015].

[56] SoftGalaxy, «The All New PHP MySQL Code Generator,» [En línea]. Available: http://softgalaxy.net/php-mysql/. [Último acceso: 19 Noviembre 2015].

[57] Yes Software, «CodeChargeStudio 5.1,» [En línea]. Available: https://www.yessoftware.com/products/product.php?product_id=1. [Último acceso: 19 Noviembre 2015].

[58] BigProf Software, «AppGini,» 4 Agosto 2015. [En línea]. Available: http://bigprof.com/appgini/. [Último acceso: 19 Noviembre 2015].

[59] SynApp2, «SynApp2,» [En línea]. Available: http://www.synapp2.org/main/. [Último acceso: 19 Noviembre 2015].

[60] Spring, «Spring Roo,» [En línea]. Available: http://projects.spring.io/spring-roo/. [Último acceso: 19 Noviembre 2015].

[61] F. S. Franco Hernández, *Especificación de requerimientos de software*, Bogotá: Pontificia Universidad Javeriana, 2016.

[62] The Eclipse Foundation, «Eclipse Documentation - Current Release,» 2015. [En línea]. Available: http://help.eclipse.org/mars/index.jsp?topic=%2Forg.eclipse.platform.doc.isv%2Fguide%2Fint_eclipse.htm. [Último acceso: 5 Mayo 2016].

[63] The Eclipse Foundation, «Xtext,» [En línea]. Available: https://eclipse.org/Xtext/. [Último acceso: 29 Abril 2016].

[64] The Eclipse Framework, «Eclipse Modeling Framework,» [En línea]. Available: https://eclipse.org/modeling/emf/. [Último acceso: 29 Abril 2016].

[65] The Eclipse Foundation, «Core,» [En línea]. Available: https://www.eclipse.org/eclipse/platform-core/. [Último acceso: 29 Abril 2016].

[66] ArchLinux User Repository, «Package Details: eclipse-user-runtime 3.0.0-3,» 2 Abril 2016. [En línea]. Available: https://aur.archlinux.org/packages/eclipse-antlr-runtime/.

[Último acceso: 29 Abril 2016].

[67] HashiCorp, «Vagrant,» [En línea]. Available: https://www.vagrantup.com/. [Último acceso: 5 Mayo 2016].

[68] The Apache Software Foundation, «About Apache,» [En línea]. Available: https://httpd.apache.org/ABOUT_APACHE.html. [Último acceso: 5 Mayo 2016].

[69] Oracle, «What is MySQL?,» [En línea]. Available: https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html. [Último acceso: 5 Mayo 2016].

[70] PHP Group, «Información general,» [En línea]. Available: https://secure.php.net/manual/es/faq.general.php. [Último acceso: 5 Mayo 2016].

[71] The Eclipse Foundation, «Package org.eclipse.xtext.util,» [En línea]. Available: http://download.eclipse.org/modeling/tmf/xtext/javadoc/2.3/org/eclipse/xtext/util/package-summary.html. [Último acceso: 29 Abril 2016].

[72] The Eclipse Foundation, «Package org.eclipse.xtext.common.types,» [En línea]. Available: http://download.eclipse.org/modeling/tmf/xtext/javadoc/2.3/org/eclipse/xtext/common/types/package-summary.html. [Último acceso: 29 Abril 2016].

[73] The Eclipse Foundation, «Package org.eclipse.emf.ecore,» [En línea]. Available: http://download.eclipse.org/modeling/emf/emf/javadoc/2.9.0/org/eclipse/emf/ecore/package-summary.html#details. [Último acceso: 29 Abril 2016].

[74] Kanban Tool, «¿Por qué utilizar la metodología Kanban?,» [En línea]. Available: http://kanbantool.com/es/metodologia-kanban. [Último acceso: 15 Mayo 2016].

[75] Trello Inc, «Trello,» Trello Inc, [En línea]. Available: https://trello.com/. [Último acceso: 15 Mayo 2016].

[76] IEEE, «IEEE Standard 29148-2011 - Systems and software engineering -- Life cycle processes --Requirements engineering,» IEEE Standards Association, 2011. [En línea]. Available: https://standards.ieee.org/findstds/standard/29148-2011.html. [Último acceso: 15 Mayo 2016].

[77] IEEE, «IEEE Standard 1016-2009 - IEEE Standard for Information Technology--Systems Design--Software Design Descriptions,» IEEE Standards Association, 2009. [En línea]. Available: https://standards.ieee.org/findstds/standard/1016-2009.html. [Último acceso: 15 Mayo 2016].

[78] F. S. Franco Hernández, *Documento de pruebas*, Bogotá: Pontificia Universidad Javeriana, 2016.

[79] F. S. Franco Hernández, *Documento de diseño de software*, Bogotá: Pontificia Universidad Javeriana, 2016.

IX – ANEXOS

Documento de requerimientos (SRS):

- Formato Word en: http://pegasus.javeriana.edu.co/~CIS1610AP01/docs/SRS.docx
- Formato PDF en: http://pegasus.javeriana.edu.co/~CIS1610AP01/docs/SRS.pdf

Documento de diseño (SDD):

- Formato Word en: http://pegasus.javeriana.edu.co/~CIS1610AP01/docs/SDD.docx
- Formato PDF en: http://pegasus.javeriana.edu.co/~CIS1610AP01/docs/SDD.pdf

Documento de pruebas:

- Formato Word en: http://pegasus.javeriana.edu.co/~CIS1610AP01/docs/Documento%20de%20pruebas.docx
- Formato PDF en: http://pegasus.javeriana.edu.co/~CIS1610AP01/docs/Documento%20de%20pruebas.pdf

Manual:

- Formato Word en: http://pegasus.javeriana.edu.co/~CIS1610AP01/docs/Manual.docx
- Formato PDF en: http://pegasus.javeriana.edu.co/~CIS1610AP01/docs/Manual.pdf