



UNIVERSIDAD
NACIONAL
DE COLOMBIA

Aplicación del desarrollo dirigido por modelos para la generación de una familia de productos de software enmarcados en el proceso de inscripción del ICFES

Pedro Augusto Salazar Díaz

Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento de Ingeniería de Sistemas e Industrial
Bogotá, D. C., Colombia
2019

Aplicación del desarrollo dirigido por modelos para la generación de una familia de productos de software enmarcados en el proceso de inscripción del ICFES

Pedro Augusto Salazar Díaz

Trabajo de grado presentado como requisito parcial para optar al título de:
Magíster en Ingeniería - Ingeniería de Sistemas y Computación

Director:

M.Sc., Henry Roberto Umaña Acosta

Codirector:

M.Sc., Jeisson Andrés Vergara Vargas

Línea de Investigación:

Ingeniería de Software Dirigida por Modelos

Grupo de Investigación:

Colectivo de Investigación en Ingeniería de Software (ColSWE)

Universidad Nacional de Colombia

Facultad de Ingeniería, Departamento de Ingeniería de Sistemas e Industrial

Bogotá, D. C., Colombia

2019

A Dios, por ser mi fuerza y mi guía en la vida; a mi esposa y mi hija, por su amor, apoyo y paciencia; y a mis papás por todo su amor y apoyo incondicional en todo momento.

Agradecimientos

Agradezco a mi esposa, hija, papás y hermanas por su apoyo, colaboración, comprensión y consejos durante toda la maestría.

Doy gracias a los profesores Henry Roberto Umaña Acosta y Jeisson Andrés Vergara Vargas por orientarme en este camino, por su conocimiento, apoyo y dedicación en la elaboración de este trabajo.

Adicionalmente, agradezco a las personas que han hecho parte del grupo de investigación Col-SWE de la Universidad Nacional de Colombia, en la línea de investigación de Ingeniería de Software Dirigida por Modelos, porque sus aportes han sido importantes en el desarrollo de este trabajo.

Finalmente, expreso mi agradecimiento al ICFES, al Director de Tecnología e Información Felipe Guzmán Ramírez, a la Subdirectora de Desarrollo de Aplicaciones Marcela Cañon Vargas, a Natalia Benavides y a Vivian Aranda por su apoyo en el desarrollo de este trabajo.

Resumen

El proceso de inscripción del Instituto Colombiano para la Evaluación de la Educación - ICFES tiene una variedad de formularios. Este trabajo usa el desarrollo de software dirigido por modelos para generar una familia de productos de software enmarcados en el proceso de inscripción del ICFES a los exámenes Saber 11 y Saber Pro. Se adapta la metodología usada en el grupo de investigación ColSWE de la Universidad Nacional de Colombia, en la línea de investigación de Ingeniería de Software Dirigida por Modelos, para obtener como resultado un editor que soporta la gramática de un lenguaje de dominio específico orientado al modelado de un formulario de inscripción. Finalmente, se genera la familia de productos de software usando el editor obtenido, reduciendo los costos de desarrollo y el tiempo de construcción de cada formulario.

Palabras clave: desarrollo de software dirigido por modelos, lenguajes de dominio específico, aplicación web, familia de productos de software.

Abstract

The registration process for assessments of the Colombian Institute for the Assessment of Education - ICFES has many forms. This work uses model-driven software development to generate a family of software products for the ICFES registration process for the Saber 11 and Saber Pro exams. The methodology used in the ColSWE research group of the Universidad Nacional de Colombia, in the research line of Model-driven Software Engineering, was adapted to obtain as a result an editor that supports the grammar of a domain-specific language oriented to model a registration form. Finally, the family of software products is generated by the obtained editor, reducing the development costs and construction time of each form.

Keywords: Model-Driven Development, Domain-Specific Language, Web Application, Software Product Family.

Contenido

Agradecimientos	vii
Resumen	ix
Lista de Figuras	xiii
Lista de Tablas	xv
1. Introducción	1
1.1. Definición del problema	2
1.1.1. Línea de Productos de Software	3
1.1.2. Desarrollo de Software Dirigido por Modelos - MDD	3
1.1.3. Comparación entre Línea de Productos de Software y Desarrollo Dirigido por Modelos	4
1.2. Objetivos	4
1.2.1. Objetivo general	4
1.2.2. Objetivos específicos	5
1.3. Esquema del documento	5
2. Contexto y trabajo relacionado	6
2.1. Lenguajes de dominio específico	6
2.2. Desarrollo de software dirigido por modelos - MDD	6
2.3. Metodología	7
2.4. Trabajo relacionado	9
3. Implementación MDD para proceso de inscripción para Saber Pro - Individual	11
3.1. Implementación de referencia	13
3.2. Análisis de componentes	15
3.2.1. Presentación	15
3.2.2. Lógica de negocio	16
3.3. Metamodelo	17
3.4. Diseño del DSL	20
3.4.1. Dominio	21
3.4.2. Arquitectura	25

3.4.3. Tecnología	26
3.5. Desarrollo de transformaciones	27
3.6. Construcción del editor	30
3.7. Generación de artefactos de software	32
4. Construcción de una familia de productos de software	33
4.1. Formulario de inscripción a Saber Pro - Exterior	33
4.2. Formulario de inscripción a Saber 11 - Estudiante	35
4.3. Formulario de inscripción a Saber 11 - Individual	39
5. Evaluación	42
5.1. Análisis conceptual	42
5.2. Análisis práctico	43
6. Conclusiones y trabajo futuro	48
6.1. Conclusiones	48
6.2. Trabajo futuro	48
A. Anexo: Modelo definido en el DSL del formulario Saber Pro - Individual	50
A.1. Fragmento del modelo del formulario	50
A.2. Fragmento del modelo del entidades	52
A.3. Modelo de la arquitectura	52
A.4. Modelo de la tecnología	53
Bibliografía	54

Lista de Figuras

1-1. Principales exámenes ofertados por el ICFES, sus poblaciones y cobertura territorial.	1
1-2. Ilustración de una familia de productos.	2
2-1. Metodología iterativa para el desarrollo dirigido por modelos.	8
3-1. Diagrama de 3-capas de la implementación de referencia.	13
3-2. Diagrama de componentes de la implementación de referencia.	14
3-3. Página para la selección de la aplicación y del aspirante, obtenida en la implementación de referencia.	14
3-4. Página para diligenciar el formulario de inscripción, obtenida en la implementación de referencia.	15
3-5. Elementos principales del metamodelo.	17
3-6. Metamodelo del dominio.	18
3-7. Metamodelo de la arquitectura.	19
3-8. Metamodelo de la tecnología.	20
3-9. Árbol de sintaxis concreta general.	21
3-10. Árbol de sintaxis concreta del dominio.	21
3-11. Árbol de sintaxis concreta del dominio - Formulario.	23
3-12. Árbol de sintaxis concreta del dominio - Entidades.	24
3-13. Árbol de sintaxis concreta de la arquitectura.	26
3-14. Árbol de sintaxis concreta de la tecnología.	27
3-15. Elementos del DSL relacionados con las secciones y las preguntas.	28
3-16. Transformación de una sección y sus preguntas para generar un archivo HTML.	28
3-17. Generación del plug-in con Eclipse.	30
3-18. Instalación del plug-in en Eclipse.	31
3-19. Uso del plug-in mostrando sugerencias.	31
3-20. Resultado del formulario generado automáticamente para la inscripción de Saber Pro - Individual.	32
4-1. Página del formulario de inscripción para Saber Pro - Exterior, obtenida de la generación automática.	34

4-2.	Página del formulario de inscripción para Saber 11 - Estudiante, obtenida de la generación automática.	38
4-3.	Página del formulario de inscripción para Saber 11 - Individual, obtenida de la generación automática.	41
5-1.	Implementación de referencia del archivo HTML de la sección datos de autorización.	43
5-2.	Elementos del DSL relacionados con las secciones y las preguntas.	44
5-3.	Fragmento de la definición del formulario usando el DSL para la inscripción al examen Saber Pro población individual.	44
5-4.	Transformación de una sección y sus preguntas para generar un archivo HTML.	45
5-5.	Comparación de la sección de datos de autorización, entre la implementación de referencia (izquierda) y el código generado automáticamente (derecha).	45

Lista de Tablas

3-1. Estructura del capítulo de información personal del formulario de inscripción para Saber Pro - Individual.	11
3-2. Estructura del capítulo de información académica del formulario de inscripción para Saber Pro - Individual.	12
3-3. Estructura del capítulo de información de la citación del formulario de inscripción para Saber Pro - Individual.	12
3-4. Estructura del capítulo de información socioeconómica del formulario de inscripción para Saber Pro - Individual.	12
3-5. Componentes genéricos de la capa de presentación.	16
3-6. Componentes individuales de la capa de presentación.	16
3-7. Componentes individuales de la capa de lógica de negocio.	17
3-8. Descripción de las transformaciones.	29
4-1. Estructura del capítulo de información básica del formulario de inscripción para Saber Pro - Exterior.	33
4-2. Estructura del capítulo de información personal del formulario de inscripción para Saber 11 - Estudiante.	35
4-3. Estructura del capítulo de información académica y de citación del formulario de inscripción para Saber 11 - Estudiante.	36
4-4. Estructura del capítulo de información socioeconómica del formulario de inscripción para Saber 11 - Estudiante.	36
4-5. Estructura del capítulo de información de antecedentes escolares y de citación del formulario de inscripción para Saber 11 - Estudiante.	37
4-6. Estructura del capítulo de información personal del formulario de inscripción para Saber 11 - Individual.	39
4-7. Estructura del capítulo de información académica y de citación del formulario de inscripción para Saber 11 - Individual.	40
4-8. Estructura del capítulo de información socioeconómica del formulario de inscripción para Saber 11 - Individual.	40
5-1. Similitud entre el código generado a partir del modelo y la implementación de referencia.	46
5-2. Líneas de código (LC) generadas a partir de las instancias del DSL de los formularios.	47

1. Introducción

El Instituto Colombiano para la Evaluación de la Educación - ICFES es una entidad gubernamental colombiana encargada de evaluar la calidad de la educación en todos sus niveles y de investigar sobre factores que la impactan [1]. Esta evaluación se realiza a partir de exámenes periódicos presentados por la población educativa. El proceso de inscripción a los exámenes ofertados se realiza de acuerdo con la población: estudiante o individual [2]. Los estudiantes son los evaluados que pertenecen a una institución educativa y que son pre-inscritos por esta. Los individuales son los evaluados que no son pre-inscritos por una institución y quieren presentar alguno de los exámenes. Adicionalmente, los exámenes tienen una cobertura territorial a nivel nacional, aunque las pruebas Saber Pro de educación superior se ofertan a nivel nacional y en el exterior. La estructura de examen-población-territorio se muestra en la Figura 1-1.

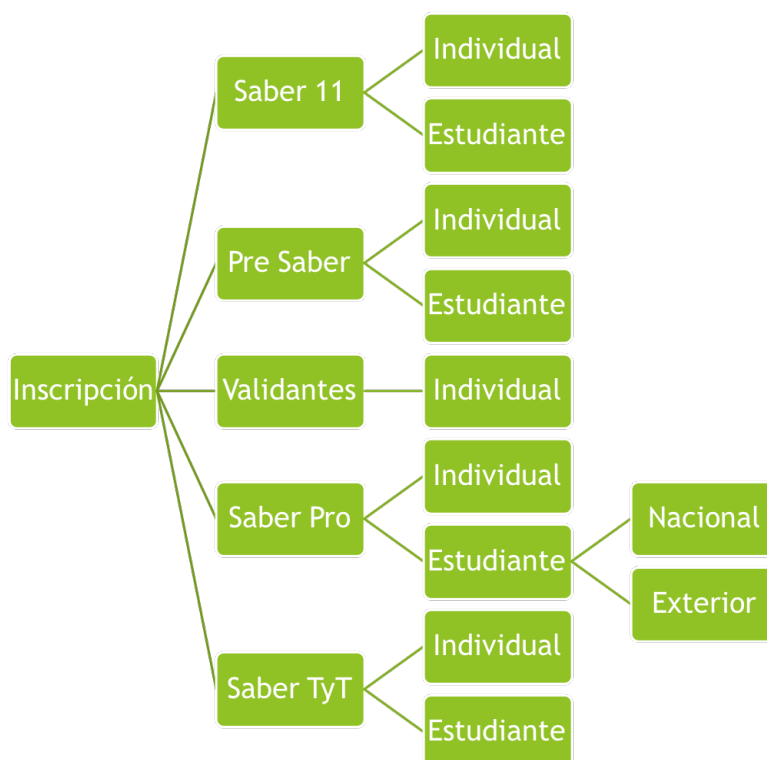


Figura 1-1.: Principales exámenes ofertados por el ICFES, sus poblaciones y cobertura territorial.

Actualmente, el ICFES tiene un formulario de inscripción por cada examen-población-territorio, que suman en total 10 formularios, de acuerdo con la Figura 1-1. Esta cantidad podría aumen-

tar porque estos servicios se pueden ofrecer a otras instituciones, como ya ha pasado con la Policía Nacional. Los formularios solicitan información común como nombres, documento de identidad, fecha de nacimiento, entre otros; así como información específica de acuerdo con el examen-población-territorio.

Estos formularios conforman una familia de productos de software, que se definen como un conjunto de productos que tienen propiedades comunes entre sí y propiedades especiales de cada uno de los miembros [3], este concepto se ilustra en la Figura 1-2. Estos formularios están desarrollados de manera independiente, de manera que, ante la solicitud de un nuevo formulario para un nuevo examen o población, o ante una solicitud de cambio, es necesario realizar un análisis de impacto, desarrollo de software y pruebas para certificar el buen funcionamiento de estos cambios. En este trabajo, se propone crear una familia de productos con el fin de reducir los tiempos y costos de estos nuevos formularios o cambios en los que actualmente se ofertan.

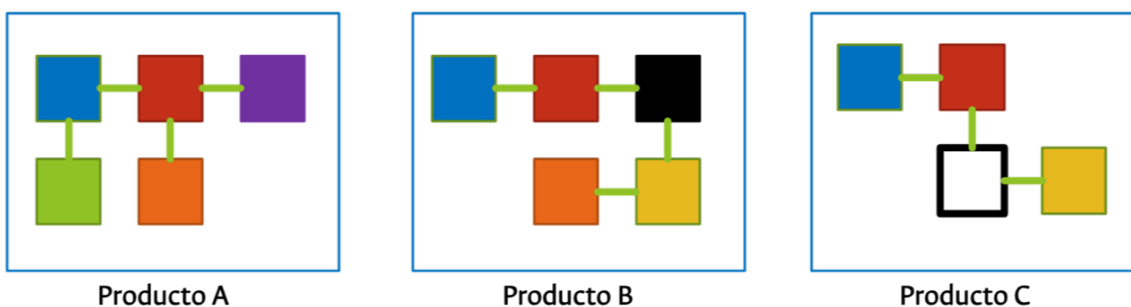


Figura 1-2.: Ilustración de una familia de productos.

1.1. Definición del problema

Como se mencionó anteriormente, el ICES tiene 10 formularios de inscripción para cada una de las combinaciones examen-población-territorio, mostradas en la Figura 1-1. Estos formularios están desarrollados de manera independiente, de manera que, ante la solicitud de un nuevo formulario por un nuevo examen o población, o ante una solicitud de cambio, es necesario realizar un análisis del impacto, desarrollo de software y pruebas para certificar el buen funcionamiento de estos cambios. La característica de familia de productos puede ser aprovechada para reducir los tiempos y costos de estos nuevos formularios o cambios en los que actualmente se ofertan. Las líneas de producto de software y el desarrollo dirigido por modelos son paradigmas para la construcción de software en general.

1.1.1. Línea de Productos de Software

Una línea de productos de software (en inglés *Software Product Line*, SPL) es un conjunto de sistemas de software que comparten características comunes, cumplen con las necesidades específicas de un dominio de negocio y se desarrollan a partir de activos centrales definidos, que son artefactos o recursos reutilizables y que forman la base para una SPL. Cada producto se construye a partir de los componentes disponibles de la base de activos comunes, adaptándose según la necesidad del cliente mediante mecanismos de variabilidad previamente definidos como parametrización o herencia. Por esto, construir un nuevo sistema es más una tarea de integración que de programación. Para cada SPL hay una guía que define la creación de productos [4].

El Software Engineering Institute de Carnegie Mellon define tres actividades principales en el desarrollo de SPL: desarrollo de activos centrales, desarrollo de producto y administración [4]. En el desarrollo de activos, se definen y desarrollan los componentes reutilizables que constituirán la línea del producto. El desarrollo de producto incluye el análisis de requerimientos, configuración e integración de los activos en la creación de productos. Por último, la administración tanto técnica como organizacional es importante para la construcción de la SPL, ya que la gestión de actividades y recursos debe ser coordinada y supervisada durante todo el proceso.

1.1.2. Desarrollo de Software Dirigido por Modelos - MDD

El desarrollo de software dirigido por modelos (en inglés *Model-Driven Development*, MDD) es un paradigma que mejora el proceso de construcción de software a través de la definición de modelos y transformaciones para generar otros artefactos (modelos, código, script, entre otros) usando lenguajes de un dominio específico [5].

Los elementos esenciales en MDD son el metamodelo, el lenguaje de dominio específico y las transformaciones que generan artefactos de software [6]. Primero, el metamodelado describe conceptos y relaciones de la estructura de un modelo en términos del dominio de negocio que define un campo de interés o conocimiento acotado. Segundo, el lenguaje de dominio específico usa los elementos identificados en el metamodelo con el fin de permitir expresar los modelos de software, es decir, las instancias del metamodelo. Por último, las transformaciones generan los artefactos de software usando la estructura del lenguaje de dominio específico y el modelo definido.

En este caso de estudio, el conjunto de todos los productos que pueden ser creados a partir del lenguaje de dominio específico es conocido como una familia de productos de software [6].

1.1.3. Comparación entre Línea de Productos de Software y Desarrollo Dirigido por Modelos

SPL genera familias de productos a partir de artefactos comunes llamados activos centrales, los cuales son coordinados y configurados para generar productos. Este paradigma está orientado a productos de gran escala y no debe ser aplicado en reutilizaciones pequeñas o de un solo sistema. Por lo general, SPL es usado en sistemas de gran escala con varias funcionalidades y comercializados a varios clientes, ofreciendo personalización del producto y aumentando la satisfacción del cliente [4].

MDD puede generar familias de productos a partir de metamodelos que permiten definir modelos para cada implementación particular. Este paradigma no condiciona el tamaño del software a construir, se adapta a los cambios tecnológicos y a los cambios en los requerimientos, mejora la comunicación con los usuarios y construye modelos como productos de larga duración [5].

Ambos paradigmas tienen beneficios en común como: aumento en la calidad del producto, uso más eficiente de los recursos humanos, incremento en la productividad y reutilización de artefactos [4, 5]. Teniendo en cuenta la revisión de los dos principales paradigmas para la construcción de familias de productos de software, se considera apropiado usar MDD para el proceso de inscripción del ICFES, ya que este producto no se va a comercializar a otros clientes, no es de gran tamaño y el metamodelo del proceso de inscripción será el eje central del desarrollo tanto para los exámenes actualmente ofertados como para los futuros.

El desarrollo dirigido por modelos (en inglés *Model-Driven Development*, MDD) es una solución a este problema, ya que se enfoca en la construcción de modelos a partir de definiciones de negocio y transformaciones para la generación semiautomática de estos formularios. Los cambios posteriormente solicitados se realizan sobre el modelo y no sobre el código, lo que reduce tiempo, costo y errores generados en la implementación de los formularios, tanto en los actuales como en los nuevos procesos de inscripción. La pregunta que orienta este trabajo es: ¿cómo implementar una familia de productos para el proceso de inscripción del ICFES usando MDD?

1.2. Objetivos

1.2.1. Objetivo general

Aplicar desarrollo de software dirigido por modelos para la generación de una familia de productos de software enmarcados en el proceso de inscripción del ICFES a los exámenes Saber 11 y Saber Pro.

1.2.2. Objetivos específicos

- Realizar una implementación de referencia para el proceso de inscripción del ICFES al examen Saber Pro.
- Diseñar un lenguaje de dominio específico para la familia de productos de software para la inscripción del ICFES, a partir del análisis de la implementación de referencia.
- Definir los cuatro modelos de inscripción para los exámenes Saber 11 (estudiante e individual) y Saber Pro (individual y estudiante exterior) como instancias del lenguaje de dominio específico diseñado.
- Diseñar un conjunto de transformaciones modelo a texto entre los modelos generados y un conjunto de lenguajes de programación.
- Implementar una herramienta que permita la generación semiautomática de código fuente a partir de las transformaciones definidas.
- Validar la consistencia entre el código fuente generado por la herramienta y el código fuente de la implementación de referencia.

1.3. Esquema del documento

El presente trabajo está estructurado de la siguiente forma. El capítulo 2 describe de manera general los conceptos involucrados en el desarrollo dirigido por modelos, la metodología usada y el trabajo relacionado en el uso de MDD para la construcción de familia de productos. El capítulo 3 evidencia la implementación de la metodología usando MDD para el proceso de inscripción de Saber Pro para la población individual. El capítulo 4 especifica los modelos de tres procesos de inscripción que conforman una familia de productos de software. El capítulo 5 evalúa los resultados obtenidos de los capítulos 3 y 4. Finalmente, en el capítulo 6 se presentan las conclusiones del trabajo y algunas ideas para trabajos futuros.

2. Contexto y trabajo relacionado

El objetivo de este capítulo es describir de manera general los conceptos involucrados en este trabajo. Este capítulo tiene el siguiente orden: lenguajes de dominio específico, desarrollo de software dirigido por modelos, metodología y trabajo relacionado.

2.1. Lenguajes de dominio específico

Los lenguajes de propósito general (en inglés *General-Purpose Languages*, GPL) son una herramienta para dar instrucciones a las computadoras. Java, C o Python son lenguajes de propósito general. Un lenguaje de dominio específico (en inglés *Domain-Specific Language*, DSL) es un lenguaje optimizado para resolver un problema en particular [7]. SQL y CSS son DSL frecuentemente usados.

Los DSL son una parte esencial en MDD puesto que permiten la definición de un lenguaje usando conceptos del dominio del negocio. Son una especificación de alto nivel, independiente de la plataforma de implementación. Los DSL tienen varios beneficios como [7]:

- Reducen la cantidad de código escrito, manteniendo el mismo contenido semántico y aumentando la productividad en comparación con los de propósito general.
- Aumentan la calidad porque tienen menos errores y una gran capacidad de mantenimiento.
- No se enfocan en los detalles de la implementación, por lo que el análisis de negocio es más fácil de implementar. Los mensajes de error pueden usar conceptos de dominio, de esta manera son más ordenados y permiten que el experto de dominio pueda participar directamente en correcciones.
- La comunicación es más clara debido a que no aborda los detalles de implementación. De esta manera, se enfoca en la complejidad esencial del negocio y no en la de implementación.

2.2. Desarrollo de software dirigido por modelos - MDD

El desarrollo de software dirigido por modelos (en inglés *Model-Driven Software Development*, MDSD) o simplemente desarrollo dirigido por modelos (en inglés *Model-Driven Development*, MDD) es una

metodología que mejora el proceso de construcción de software a través de la definición de modelos y transformaciones para generar otros artefactos (modelos, código, script, entre otros) usando lenguajes de un dominio específico [5].

Los elementos esenciales en MDD son el metamodelo, el DSL y las transformaciones que generan artefactos de software [6]. Primero, el metamodelado describe conceptos y relaciones de la estructura de un modelo en términos del dominio de negocio que define un campo de interés o conocimiento acotado. Segundo, el DSL usa los elementos identificados en el metamodelo con el fin de permitir expresar los modelos de software, es decir, las instancias del metamodelo. Por último, las transformaciones generan los artefactos de software usando la estructura del DSL y el modelo definido.

El conjunto de todos los productos que pueden ser creados a partir del DSL es conocido como una familia de productos de software [6].

2.3. Metodología

La metodología propuesta tiene como base las metodologías de Sthal [6], Cruz [8] y Vergara [9]. Esta consiste en realizar un análisis sobre una aplicación con el fin de identificar elementos comunes, definir el metamodelo y luego usarlo para generar artefactos de software usando transformaciones. Las siguientes secciones describen cada etapa de la metodología propuesta en la Figura 2-1.

1. **Implementación de referencia:** esta implementación es útil para identificar la transición del modelo a la implementación en una plataforma específica. Es creada manualmente luego de la definición de la arquitectura, tecnologías y herramientas para cumplir con los requerimientos del software.
2. **Análisis:** consiste en analizar e identificar tres clases de componentes en la implementación de referencia.
 - a) **Componentes genéricos:** son los componentes específicos en la plataforma; estos son elementos imprescindibles para el buen funcionamiento de la aplicación.
 - b) **Componentes frecuentes:** son los componentes que tienen la misma estructura de acuerdo con el dominio o que son exactamente iguales en cada aplicación.
 - c) **Componentes individuales:** son los componentes que no pueden ser abstraídos y generados automáticamente porque son particularidades de la aplicación, estos deben ser incluidos al finalizar la generación de componentes.

El análisis se realiza considerando los siguientes aspectos:



Figura 2-1.: Metodología iterativa para el desarrollo dirigido por modelos.

- **Dominio de aplicación:** identificar las relaciones y entidades en términos de los conceptos que son usados por el negocio.
 - **Arquitectura:** definir las relaciones de los elementos de software de acuerdo con los componentes de la aplicación.
 - **Plataforma:** describir los elementos propios del lenguaje de programación o framework utilizado, por ejemplo, archivos de configuración, elementos estáticos, entre otros.
3. **Metamodelado y diseño del DSL:** a partir de los componentes identificados en la etapa anterior se define el metamodelo. Este contiene los elementos comunes que se abstraen en elementos usando conceptos del dominio de la aplicación, arquitectura y plataforma. El metamodelo permite realizar una primera validación del modelo que está en proceso de definición. La abstracción realizada en esta etapa y expresada en el metamodelo es el corazón de la ingeniería dirigida por modelos. En esta etapa no es necesario definir relaciones entre el dominio de aplicación, arquitectura y plataforma, sino poder definir el metamodelo de cada uno con el fin de identificar las entidades y relaciones de cada aspecto.

Posteriormente, se define el DSL, en el que se expresa la gramática del lenguaje de dominio de la aplicación, de la arquitectura, de la plataforma y sus relaciones. Es importante que el DSL tenga una correspondencia con el metamodelo definido pues el metamodelo es la base de abstracción de elementos que son usado en el DSL con el fin de poder definir instancias del metamodelo llamadas modelos.

Los árboles sintácticos permiten estructurar la gramática y poder realizar validaciones en casos concretos para validar la definición del DSL.

La gramática del DSL es importante porque será el medio por el cual se definirán las aplicaciones que se generan automáticamente.

4. **Diseño de transformaciones:** en esta etapa se identifican las relaciones entre la definición formal del modelo y la plataforma seleccionada para generar automáticamente los artefactos de software para esta última. Los modelos son construidos usando las definiciones del metamodelo y DSL obtenidas en la etapa anterior.
5. **Construcción de la herramienta:** al tener la definición del metamodelo, DSL y las transformaciones, se construye un editor para que los desarrolladores puedan crear instancias del DSL. Este editor valida, sugiere y autocompleta fragmentos de código del DSL.
6. **Validación:** consiste en evaluar si el software cumple con los requerimientos. El principal objetivo es asegurar que el software generado cumple con el funcionamiento que fue planteado en la implementación de referencia.

2.4. Trabajo relacionado

El trabajo relacionado identifica las aplicaciones que han usado MDD en su construcción y entre ellas, las que lo han usado en la construcción de familias de productos de software. A continuación, se listan las aplicaciones.

- MoDev [8] simplifica el proceso de desarrollo web, mejora la productividad y la calidad. MoDev abstrae la arquitectura del software para que pueda ser modelada y generada con el fin de reducir esfuerzos e invertir el tiempo de desarrollo en la lógica requerida del negocio.
- Una herramienta para la generación automática del código fuente para aplicaciones con arquitectura Modelo Vista Controlador (MVC) usando MDD [10]. A partir del modelado textual usando Xtext, se genera código fuente para varias plataformas (Java, PHP, Jasper, HTML).
- Desarrollo de un sitio web de comercio electrónico usando MDD [11] y el patrón de arquitectura MVC. Dentro del proceso de MDD, se usó UML para la definición del modelo

independiente de la plataforma (en inglés *Platform-Independent Model*, PIM) y el modelo específico de la plataforma (en inglés *Platform-Specific Model*, PSM) se definió para Java Server Faces (JSF).

- Car License Internet Payment (CLIP) [12] permite a los propietarios de carros en Chile pagar los costos de las licencias de los carros. MDA fue usado para definir modelos abstractos que permiten modelar aplicaciones web basadas en MVC.
- Desarrollo de organizaciones virtuales como mecanismo para mostrar la cooperación y colaboración entre ellas [13]. Se definió el metamodelo, pero no se realizó la etapa de implementación.
- Desarrollo de prototipos de aplicaciones de internet enriquecidas (en inglés *Rich Internet Application*, RIA) usando MDD [14] con el propósito de fabricarlos rápidamente. MDD se implementó hasta generar automáticamente el código de un prototipo listo para ser ejecutado.
- Generación de interfaces de usuario para dispositivos móviles usando MDD [15]. Desde el nivel de modelamiento PIM, definen la interfaz y luego, usando transformaciones, se generan interfaces para dispositivos móviles con iOS y Android.
- Despliegue dirigido por modelos para el escalamiento de arquitecturas de software distribuidas en la nube [16].

Algunas aplicaciones que han usado MDD para la construcción de una familia de productos de software son:

- MD-SPL Cupi2¹ [17, 18] que genera proyectos completos de programación como ejercicios para la plataforma de enseñanza de programación.
- Generación de servicios SOA a partir de una familia de procesos de negocio [19].
- Desarrollo de software para nodos de un sistema de inteligencia ambiental (en inglés *ambient intelligence*, AmI) [20]. El uso de MDD reduce la complejidad de esta tarea debido a la variedad de sensores y dispositivos que se usarán como nodos dentro del sistema.

¹Universidad de los Andes - Cupi2. <https://cupi2.virtual.uniandes.edu.co/>

3. Implementación MDD para proceso de inscripción para Saber Pro - Individual

La estructura de un formulario de inscripción está conformada por capítulos, los cuales agrupan secciones, que a su vez agrupan preguntas. El formulario de inscripción para Saber Pro - población individual esta conformado por cuatro capítulos, siete secciones, y cuarenta y dos preguntas. El detalle del capítulo de información personal se muestra en la Tabla 3-1, el de información académica en la Tabla 3-2, el de información de la citación en la Tabla 3-3 y el de información socioeconómica en la Tabla 3-4.

Tabla 3-1.: Estructura del capítulo de información personal del formulario de inscripción para Saber Pro - Individual.

Capítulo	Sección	Pregunta
Información personal	Datos de autorización	¿Autoriza la entrega de información a terceros?
	Datos personales	Primer nombre
		Segundo nombre
		Primer apellido
		Segundo apellido
		Tipo de documento
		Número de documento
		Nacionalidad
		Género
		Fecha de nacimiento
	Datos de contacto	Departamento
		Municipio
		Dirección
		Teléfono
		Área de residencia
Correo electrónico		

Tabla 3-2.: Estructura del capítulo de información académica del formulario de inscripción para Saber Pro - Individual.

Capítulo	Sección	Pregunta
Información académica	Estudios de pregrado	Universidad
		Programa
		Semestre actual
		Valor matrícula
		¿Cómo se preparó para el examen?
	Estudios de educación media	¿Es validante ICFES?
		Nombre del Colegio
		Título obtenido
		Tipo de documento
		Número de documento

Tabla 3-3.: Estructura del capítulo de información de la citación del formulario de inscripción para Saber Pro - Individual.

Capítulo	Sección	Pregunta
Información de citación	Ciudad de presentación del examen	Ciudad de preferencia
		Zona de preferencia

Tabla 3-4.: Estructura del capítulo de información socioeconómica del formulario de inscripción para Saber Pro - Individual.

Capítulo	Sección	Pregunta
Información socioeconómica	Datos familiares	Nivel educativo de su padre
		Nivel educativo de su madre
		Horno microondas u horno eléctrico o a gas
		Moto
		Servicio o conexión a internet
		Servicio cerrado de televisión (cable satelital o parabólica)
		Computador
		Máquina lavadora de ropa
		Automóvil particular
		¿Cuántos libros físicos o electrónicos hay en su hogar?
		¿Cuántas personas conforman el hogar donde vive actualmente, incluido usted?
		En total, ¿en cuántos cuartos duermen las personas de su hogar?
		¿Es usted jefe de hogar o cabeza de familia?
		¿Cuántas personas dependen económicamente de usted?

3.1. Implementación de referencia

La implementación de referencia del formulario fue construida usando una arquitectura de 3-capas (3-tier): la capa de datos, la capa de lógica de negocio y la capa de presentación, como se muestra en la Figura 3-1. Estas se encuentran físicamente separadas.

La capa de presentación tiene tres capas lógicas: GUI contiene el HTML y estilos CSS; la lógica de presentación está asociada a cada página y ofrece acceso a los métodos que usan los eventos de la página; y los servicios de la capa de presentación consumen los de la capa de lógica de negocio.

La capa de lógica de negocio tiene también tres capas lógicas: los controladores exponen los servicios REST, los servicios contienen la lógica de negocio, y los repositorios se encargan de administrar la persistencia con la base de datos.

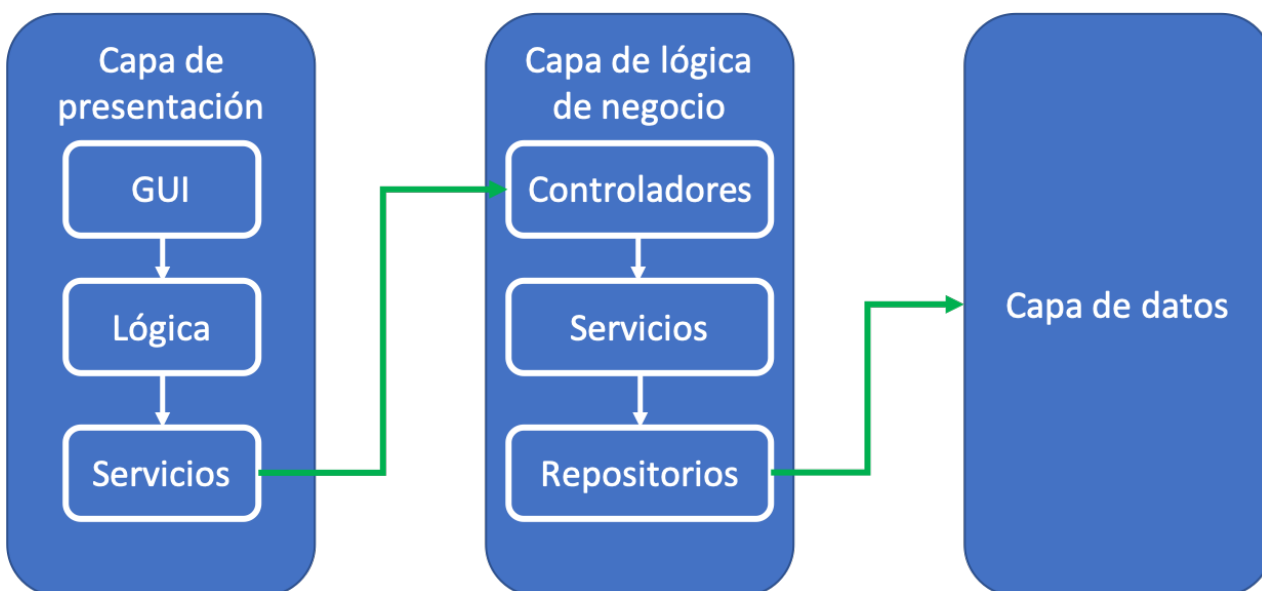


Figura 3-1.: Diagrama de 3-capas de la implementación de referencia.

El sistema está compuesto por tres componentes mostrados en la Figura 3-2. Las tecnologías usadas fueron seleccionadas de acuerdo con lo que el ICFES usa actualmente. El componente de presentación se construyó con Angular 7, el de servicios de negocio con Spring 5 y Java 8, y el de base de datos con Oracle. La conexión entre los servicios de negocio y la base de datos usa JDBC. El componente de servicios de negocio expone servicios REST que son consumidos por el de presentación.

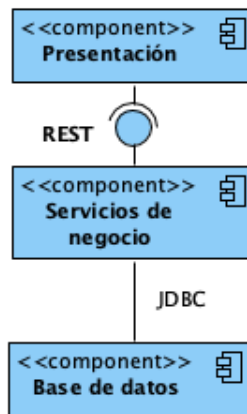


Figura 3-2.: Diagrama de componentes de la implementación de referencia.

La implementación de referencia sólo se hizo para el formulario de inscripción a Saber Pro - Individual. El resultado de la implementación de referencia es un aplicación con dos páginas. Una para seleccionar la aplicación y el aspirante para redirigirlo al formulario, y otra con el formulario de inscripción para diligenciarlo. Las páginas son mostradas en las Figura 3-3 y Figura 3-4.

icfes mejor saber MINEDUCACIÓN

Facebook Twitter YouTube

Listado de aspirantes

Convocatoria: * Saber Pro

Mostrar 10 registros

Documento	Nombre	Estado	Acciones
CC 1122334455	JUAN SEBASTIAN ROMERO RAMIREZ	PPI	Formulario
CC 28476927	MARIA JOSE HERNANDEZ ARIAS	PPI	Formulario
CC 28476927	MARGARITA EXTERIOR RODRIGUEZ PAEZ	PPI	Formulario
CC 67092747	ROSA CAÑON MEDINA	PPI	Formulario
CC 67092747	ROSA EXTERIOR HERNANDEZ ARIAS	PPI	Formulario
CC 76394285	MARGARITA DIAZ PEREZ	PPI	Formulario
CC 98277194	CARLOS RODRIGUEZ PAEZ	PPI	Formulario
CC 98277194	CARLOS EXTERIOR CAÑON MEDINA	PPI	Formulario

Mostrando registros del 1 al 8 de un total de 8 registros Anterior Siguiete

© 2019 Instituto Colombiano para la Evaluación de la Educación ICFES | Atención al Ciudadano: Calle 26 No.69-76, Torre 2, Piso 15, Edificio Elemento, Bogotá |
 Horario de Atención: Lunes a Viernes 8 am a 4 pm Línea Local (57+1) 484-1460 | PBX: (57+1) 484-1410
 Línea Nacional Gratuita 018000-519535

Figura 3-3.: Página para la selección de la aplicación y del aspirante, obtenida en la implementación de referencia.

Formulario inscripción - Saber Pro Individual

Información Personal | Información Académica | Información de Citación | Información Socioeconómica

Datos Autorización

¿Autoriza la entrega de su información a terceros (Personas naturales o jurídicas, entidades públicas o privadas) que otorgan estímulos o incentivos a los estudiantes con mejores resultados?:*

Datos personales

Primer Nombre:* Ingrese primer nombre Segundo Nombre:* Ingrese segundo nombre

Primer Apellido:* Ingrese primer apellido Segundo Apellido:* Ingrese segundo apellido

Tipo de Documento:* Número de Documento de Identidad:* Ingrese número de documento

Nacionalidad:* Ingrese nacionalidad Género:*

Fecha de Nacimiento:* mm/dd/yyyy

Datos de contacto

Departamento:* Ingrese departamento Municipio:* Ingrese municipio

Dirección:* Ingrese dirección Teléfono:* Ingrese teléfono

Celular:* Ingrese celular Área de residencia:*

Correo electrónico:* Ingrese correo electrónico

Siguiete

© 2019 Instituto Colombiano para la Evaluación de la Educación ICFES | Atención al Ciudadano: Calle 26 No.69-76, Torre 2, Piso 15, Edificio Elemento, Bogotá | Horario de Atención: Lunes a Viernes 8 am a 4 pm Línea Local (57+1) 484-1460 | PBX: (57+1) 484-1410

Figura 3-4.: Página para diligenciar el formulario de inscripción, obtenida en la implementación de referencia.

3.2. Análisis de componentes

A continuación, se identifican los componentes genéricos, frecuentes e individuales de la implementación de referencia.

3.2.1. Presentación

3.2.1.1. Componentes genéricos

Los archivos de configuración de parámetros, rutas, plantilla base y registro de componentes son clasificados como componentes genéricos. La Tabla 3-5 describe la función de cada archivo.

Tabla 3-5.: Componentes genéricos de la capa de presentación.

Archivo	Función
app.component.html	Plantilla base para generar los componentes de Angular.
app-routing.module.ts	Descripción de la correspondencia de URLs con los componentes de Angular.
app.module.ts	Registro de los componentes que se usan en la aplicación.
environment.ts	Parámetros generales usados en la aplicación.

3.2.1.2. Componentes frecuentes

Estos componentes incluyen los componentes Angular de capítulos y secciones, y las entidades.

3.2.1.3. Componentes individuales

Estos comprenden el consumo de los servicios REST y algunos componentes por las particularidades en la implementación. La Tabla 3-6 describe la función de cada elemento.

Tabla 3-6.: Componentes individuales de la capa de presentación.

Tipo	Identificador	Función
Componente	Formulario	Gestiona el formulario.
Componente	Capítulo	Describe el comportamiento de un capítulo.
Componente	Sección	Describe el comportamiento de una sección.
Componente	Listado de aspirantes	Lista los aspirantes de un examen.
Componente	Selector de respuesta	Componente para gestionar las preguntas con un listado de respuestas posibles.
Servicio	Formulario	Consume los servicios de formulario de consulta y persistencia.
Servicio	Inscripción	Consume los servicios de inscripción para la consulta de exámenes y de aspirantes.

3.2.2. Lógica de negocio

3.2.2.1. Componentes genéricos

Incluye el archivo de configuración de parámetros para Spring (*application.properties*), donde se configura la conexión a base de datos.

3.2.2.2. Componentes frecuentes

Estos componentes comprenden la estructura de los controladores, servicios, repositorios y entidades.

3.2.2.3. Componentes individuales

Incluye la exposición y definición de servicios por las particularidades en la lógica de negocio. La Tabla 3-7 describe la función de cada archivo.

Tabla 3-7.: Componentes individuales de la capa de lógica de negocio.

Tipo	Identificador	Función
Controlador	Formulario	Expone los servicios relacionados con el formulario.
Controlador	Inscripción	Expone los servicios relacionados con el listado de aspirantes.
Servicio	Formulario	Contiene la lógica de negocio de los servicios de formularios.
Servicio	Inscripción	Contiene la lógica de negocio de los servicios de inscripción.

3.3. Metamodelo

El metamodelo tiene tres elementos principales: el dominio, la arquitectura y la tecnología, como se muestra en la Figura 3-5.

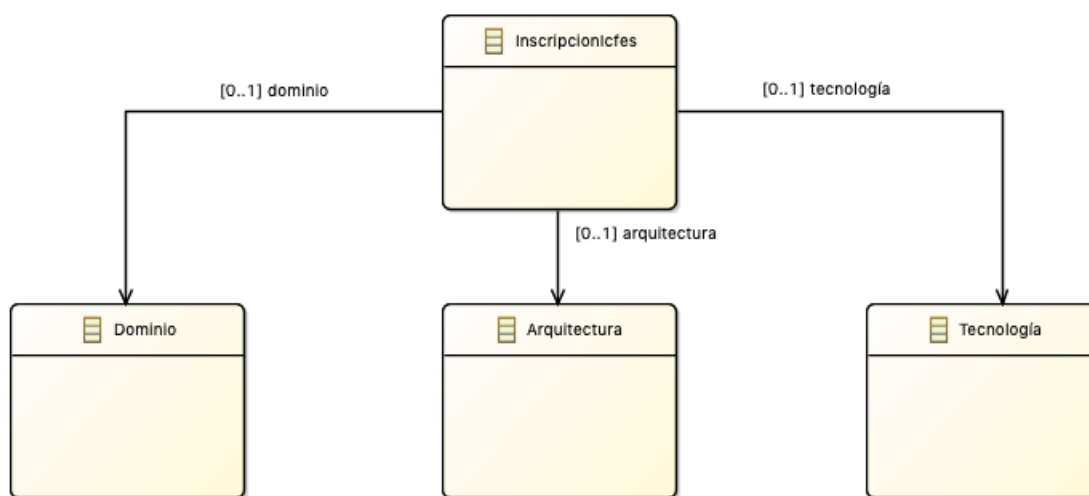


Figura 3-5.: Elementos principales del metamodelo.

El matamodelo del dominio (Figura 3-6) contiene la definición del dominio del negocio: el formulario de inscripción y las entidades relacionadas con la inscripción que representan el modelo

de datos de la aplicación. El formulario esta conformado por un conjunto ordenado de capítulos, que agrupan secciones ordenadas y estas a su vez contienen preguntas definiendo el tipo y las respuestas disponibles cuando se trata de una pregunta de selección.

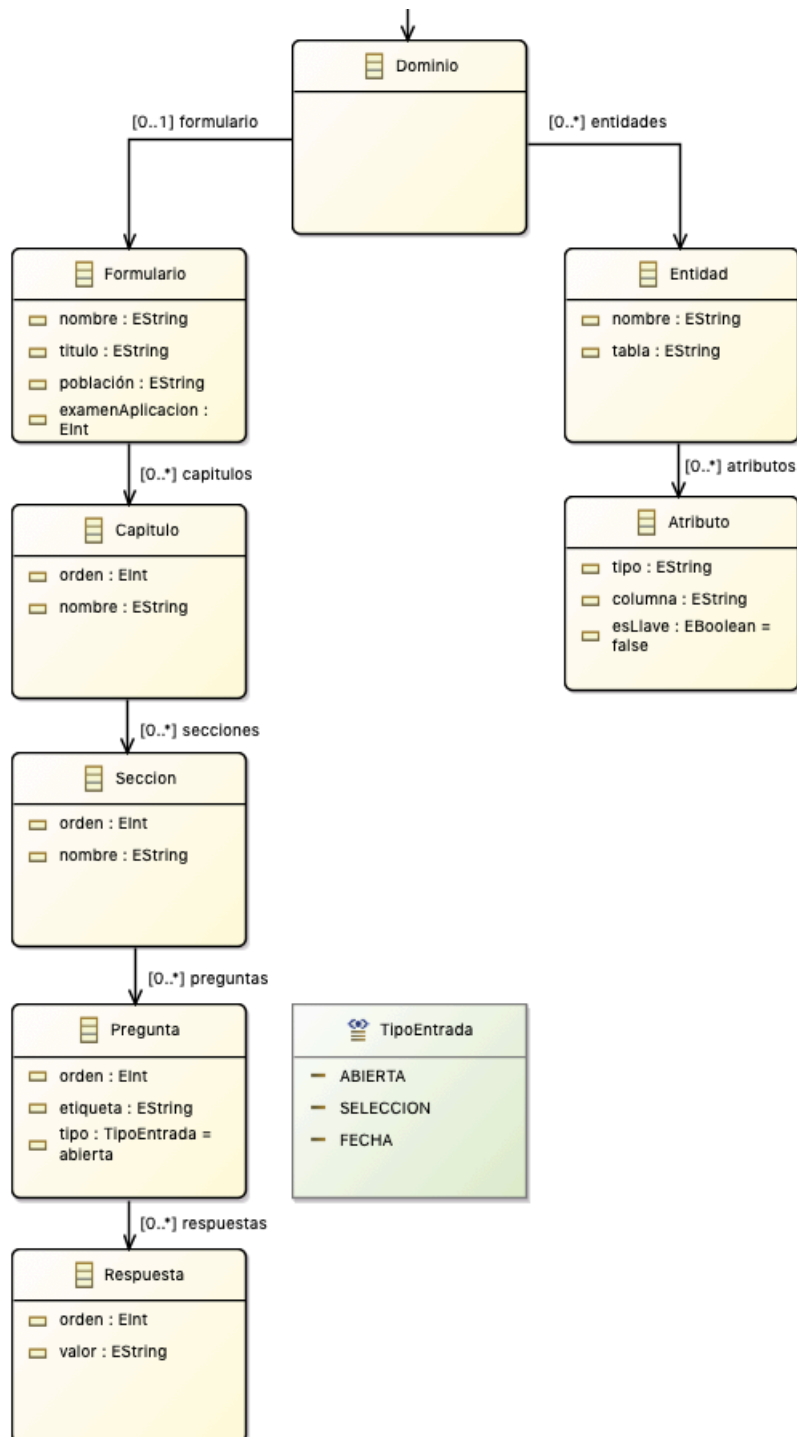


Figura 3-6.: Metamodelo del dominio.

El metamodelo de la arquitectura (Figura 3-7) muestra la arquitectura de capas usada en la capa de presentación (*front*) y la lógica de negocio (*back*). El back contiene los controladores que exponen los servicios REST, usan servicios de negocio que consultan y persisten datos a través de los repositorios asociados a una entidad. El front está compuesto por componentes de Angular donde se definen los elementos de las páginas web y su comportamiento. El flujo de datos de consulta y la persistencia se realiza mediante los servicios que consume el front.

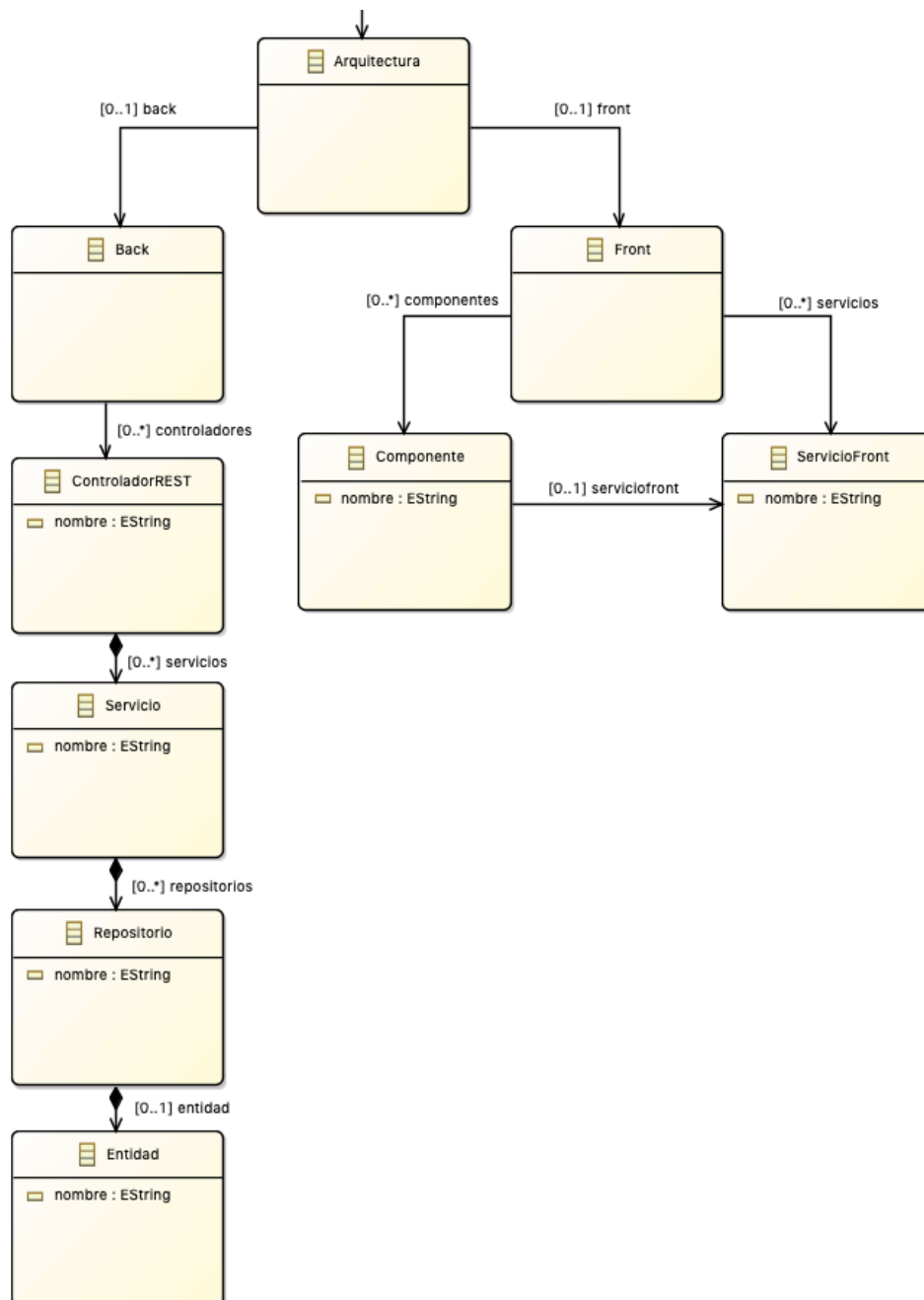


Figura 3-7.: Metamodelo de la arquitectura.

El metamodelo de tecnología (Figura 3-8) contiene elementos particulares usados en el framework de trabajo en cada capa. En este metamodelo, se definieron los archivos de configuración del front y del back y se adicionó el elemento BaseDatosPropiedad con el fin de definir los parámetros de conexión a la base de datos.

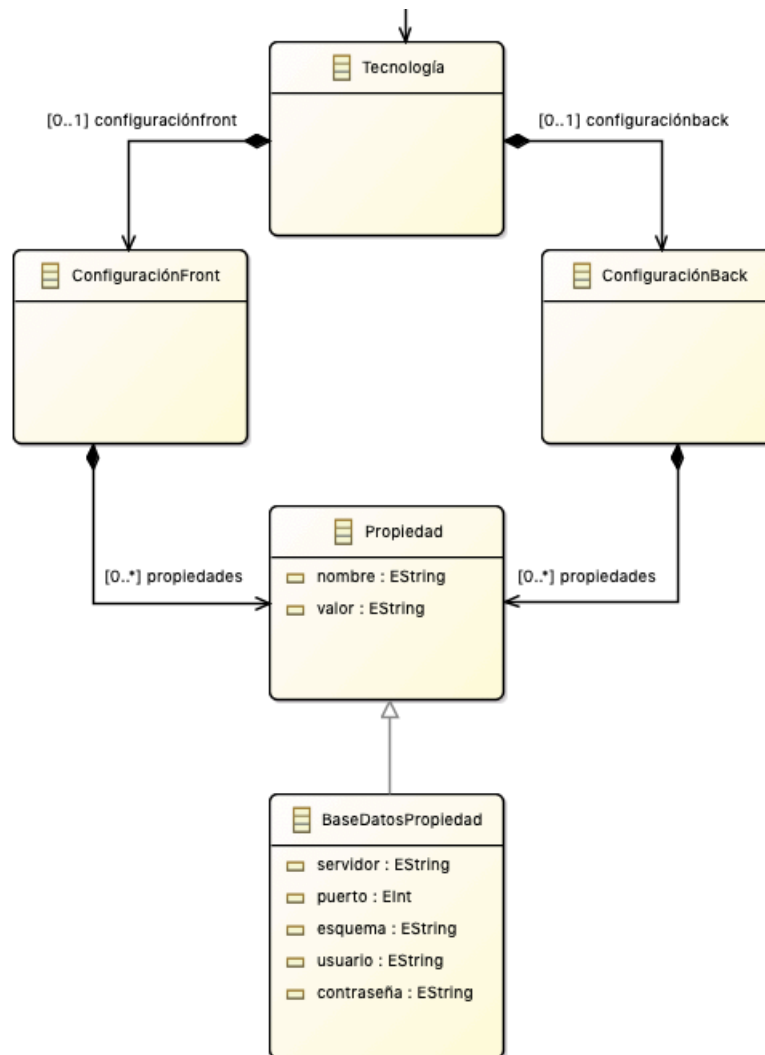


Figura 3-8.: Metamodelo de la tecnología.

3.4. Diseño del DSL

En esta etapa, se definió la gramática del lenguaje de dominio específico en el cual se modela el sistema para el formulario de inscripción. La gramática se presenta en la definición realizada con Xtext¹ y en árboles de sintaxis concreta (en inglés *Concrete Syntax Tree*, CST), que representan

¹Xtext, <https://www.eclipse.org/Xtext/>

la gramática de un lenguaje de computación usando un árbol como estructura.

La gramática se definió en términos del dominio, la arquitectura y la tecnología como se muestra a continuación y en la Figura 3-9.

grammar inscripcionFormulario with org.eclipse.xtext.common.Terminals

generate inscripcionFormulario “<http://www.gov.co/icfes/dsl/inscripcion/InscripcionFormulario>”

InscripcionICFES:

`‘inscripción-icfes’ { dominio=Dominio arquitectura=Arquitectura tecnologia=Tecnologia };`

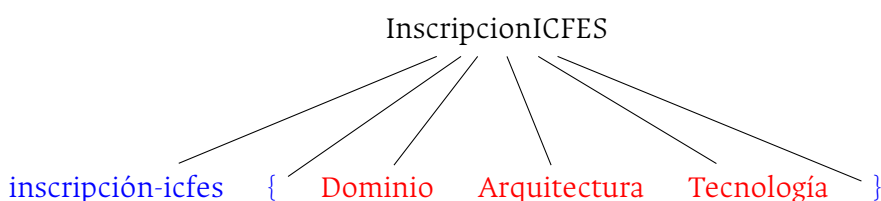


Figura 3-9.: Árbol de sintaxis concreta general.

3.4.1. Dominio

El dominio tiene dos elementos: el formulario y las entidades. Cada uno de estos elementos permite definir en el sistema el dominio del negocio para la inscripción. La gramática del dominio en general se muestra a continuación y en la Figura 3-10.

Dominio:

`‘dominio’ { formulario=Formulario entidades=Entidades };`

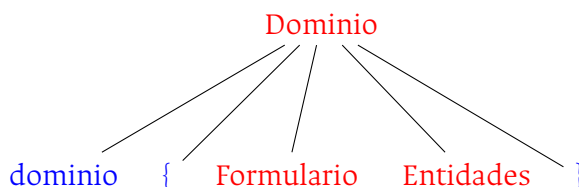


Figura 3-10.: Árbol de sintaxis concreta del dominio.

3.4.1.1. Formulario

La gramática del formulario permite definir un identificador, el título, los capítulos (debe tener al menos uno), el componente de presentación, el examen-aplicación con que se relaciona y la

población a la que esta orientado. El capítulo tiene un identificador, el título, las secciones (debe tener como mínimo una), el orden y el componente de presentación. La sección tiene un identificador, orden, título, preguntas (debe tener al menos una) y el componente de presentación. La pregunta tiene un identificador, tipo, orden, etiqueta y preguntas en caso de contar con opciones de respuesta predefinidas. Por último, la respuesta está conformada por un identificador, orden y valor.

El componente de presentación se relaciona con los componentes descritos en la arquitectura del front, con el fin de definir el comportamiento que tendrá cada elemento del formulario.

La gramática del formulario se muestra a continuación y en la Figura 3-11.

Formulario:

```
'formulario-inscripción' name=ID '{
'título' ':' titulo=STRING ','
examenAplicacion=ExamenAplicacion ','
poblacion=Poblacion ','
'componente' ':' componente=[Componente] ','
'capítulos' '{ (capitulos+=Capitulo)+ }'
'};
```

ExamenAplicacion:

```
'examen-aplicación' ':' name=INT;
```

Poblacion:

```
'población' '[' poblacion=('individual' | 'estudiante') '];
```

Capitulo:

```
name=ID '{ 'orden-capítulo' ':' orden=INT ','
'título' ':' nombre=STRING ','
'componente' ':' componente=[Componente] ','
'secciones' '{ (secciones+=Seccion)+ }'
'};
```

Seccion:

```
name=ID '{ 'orden-sección' ':' orden=INT ','
'título' ':' nombre=STRING ','
'componente' ':' componente=[Componente] ','
'preguntas' '{ (preguntas+=Pregunta)+ }'
'};
```

Pregunta:

```

name=ID '{ 'orden-pregunta' ':' orden=INT '{
'etiqueta' ':' etiqueta=STRING '{
'tipo-pregunta' ':' tipo=TipoEntrada
(',' 'respuestas' '{ (respuestas+=Respuesta)+ ')}'?
}';

```

Respuesta:

```

name=STRING '{ 'orden-respuesta' ':' orden=INT '{
'valor-respuesta' ':' valor=STRING
}';

```

enum TipoEntrada:

```

ABIERTA='abierta' | SELECCION='selección' | FECHA='fecha';

```

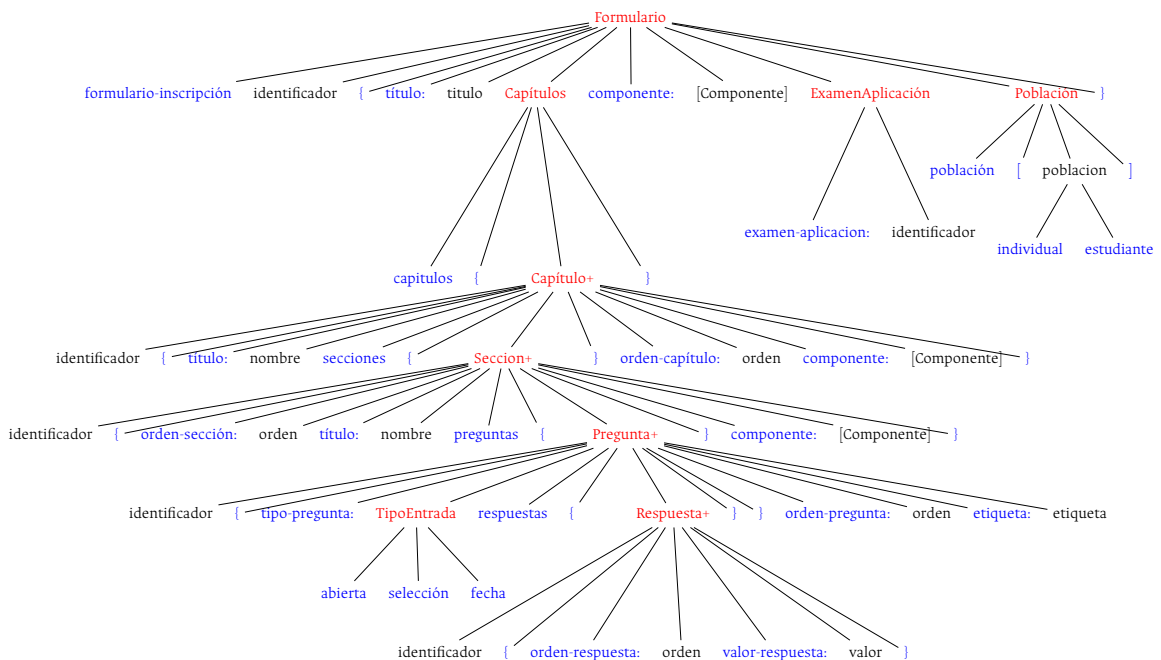


Figura 3-11.: Árbol de sintaxis concreta del dominio - Formulario.

3.4.1.2. Entidades

La gramática de las entidades permite definir varias entidades relacionadas con el sistema de inscripción. Esta tiene un identificador, la tabla en base de datos que representa y varios atributos, que tienen un tipo (número, cadena, fecha u otra Entidad), un identificador, el tipo de relación que no es obligatorio y otras características como si es llave o no, la columna que representa, si es un tipo de dato grande, si es necesario en la aplicación pero no en el modelo de datos

(fueraModelo), si es usado en el front o si es un atributo para ordenar datos por ese criterio. La gramática de las entidades se muestra a continuación y en la Figura 3-12.

Entidades:

```
{Entidades} 'entidades' '{ (entidades+=Entidad)* }';
```

Entidad:

```
'entidad' name=ID ([ 'tabla' '=' tabla=STRING '']?) ('{ (atributos+=Atributo)* }')?;
```

Atributo:

```
(tipo=TipoDato | entidad=[Entidad]) name=ID
(relacion=TipoRelacion ([ 'atributo' ':' atributoRelacion=STRING '']?)?
('{ ('llave' ':' llave='verdadero')?
('columna' ':' columna=STRING)?
('esGrande' ':' esGrande='verdadero')?
('fueraModelo' ':' fueraModelo='verdadero')?
('incluirFront' ':' incluirFront='falso')?
('ordenar' ':' columnaOrden=STRING orden=('asc' | 'desc'))?}'))?;
```

enum TipoDato:

```
NUMERO='número' | CADENA='cadena' | FECHA='fecha';
```

enum TipoRelacion:

```
SIN_ESPECIFICAR='-' | UNO_UNO='UnoAUno' | MUCHOS_UNO='MuchosAUno'
| UNO_MUCHOS='UnoAMuchos' | MUCHOS_MUCHOS='MuchosAMuchos';
```

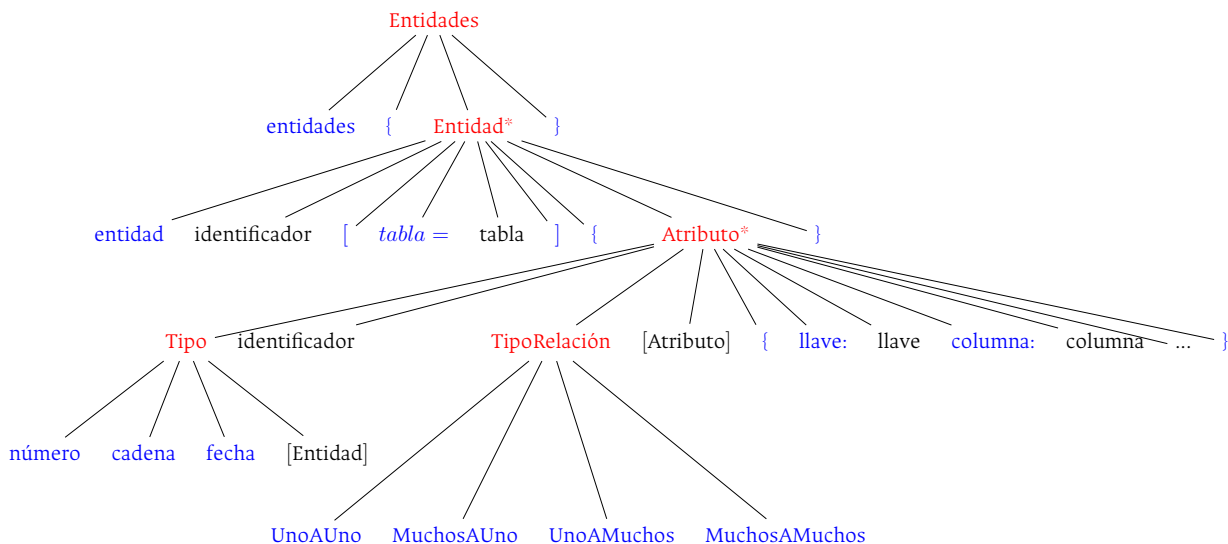


Figura 3-12.: Árbol de sintaxis concreta del dominio - Entidades.

3.4.2. Arquitectura

La gramática de la arquitectura tiene dos elementos la arquitectura del back y la del front. La arquitectura del back permite la definición de controladores, los servicios asociados a estos, los repositorios usados en esos servicios y la entidad que es gestionada por el repositorio. La arquitectura del front permite definir componentes y servicios, y relacionarlos entre si. La gramática de la arquitectura se muestra a continuación y en la Figura 3-13.

Arquitectura:

```
'arquitectura' '{ back=ArquitecturaBack front=ArquitecturaFront }';
```

ArquitecturaBack:

```
'back' '{ controladores=Controladores }';
```

ArquitecturaFront:

```
'front' '{ componentes=Componentes servicios=ServiciosFront }';
```

Controladores:

```
'controladores' '{ (controladores+=Controlador)+ }';
```

Controlador:

```
'controlador' name=ID '{ servicios=Servicios }';
```

Servicios:

```
'servicios' '{ (servicios+=Servicio)+ }';
```

Servicio:

```
'servicio' name=ID '{ repositorios=Repositorios }';
```

Repositorios:

```
{Repositorios} 'repositorios' '{ (repositorios+=Repositorio)* }';
```

Repositorio:

```
'repositorio' name=ID '[' entidad=[Entidad] ']';
```

Componentes:

```
'componentes' '{ (componentes+=Componente)+ }';
```

Componente:

```
'componente' name=ID ('{
  'servicios' '{ servicios+=[ServicioFront] (',' servicios+=[ServicioFront])* }'
}')*;
```

ServiciosFront:

```
{ServiciosFront} 'serviciosFront' '{ (servicios+=ServicioFront)* }';
```

ServicioFront:

```
'servicioFront' name=ID;
```

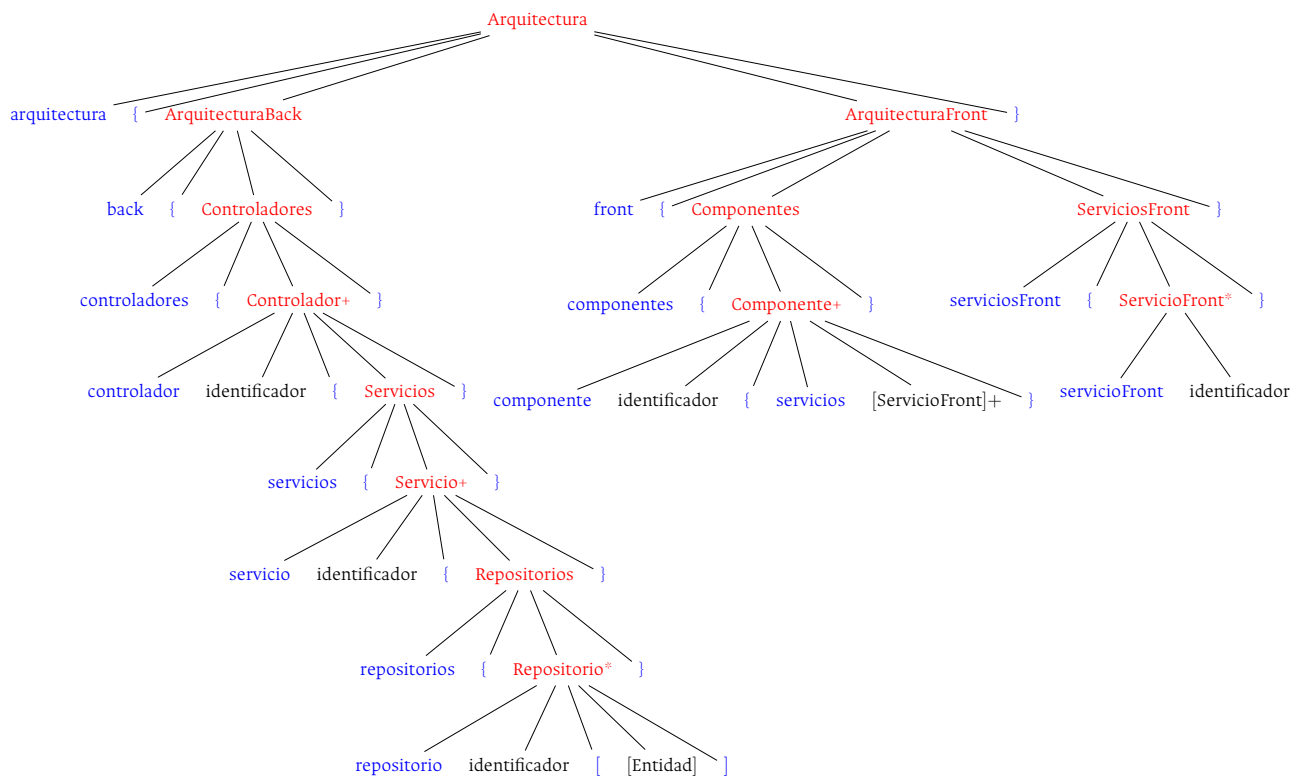


Figura 3-13.: Árbol de sintaxis concreta de la arquitectura.

3.4.3. Tecnología

La gramática de la tecnología permite definir parámetros de configuración del back y del front, se muestra a continuación y en la Figura 3-14.

Tecnología:

```
{Tecnología} 'tecnología' '{ (configuracion=Configuracion)? }';
```

Configuracion:

```
{Configuracion} 'configuración' '{
  'back' '{ (propiedadesBack=Propiedades)? }'
  'front' '{ (propiedadesFront=Propiedades)? }'
}';
```



```

Seccion:
  name=ID '{ 'orden-sección' ':' orden=INT ','
  'titulo' ':' nombre=STRING ','
  'componente' ':' componente=[Componente] ','
  'preguntas' '{ (preguntas+=Pregunta)+ '}'
  '}'

Pregunta:
  name=ID '{ 'orden-pregunta' ':' orden=INT ','
  'etiqueta' ':' etiqueta=STRING ','
  'tipo-pregunta' ':' tipo=TipoEntrada
  (, 'respuestas' '{ (respuestas+=Respuesta)+ '}')?
  '}'

Respuesta:
  name=STRING '{ 'orden-respuesta' ':' orden=INT ','
  'valor-respuesta' ':' valor=STRING
  '}'

enum TipoEntrada:
  ABIERTA='abierta' | SELECCION='selección' | FECHA='fecha';

```

Figura 3-15.: Elementos del DSL relacionados con las secciones y las preguntas.

```

def static _generarHTML(Seccion seccion) '''
<fieldset id="fs<seccion.name.toFirstUpper>">
  <legend><seccion.nombre></legend>
  <var indice = 0>
  <FOR preg : seccion.preguntas>
    <div class="form-group required col-sm-6">
      <label class="control-label col-sm-4"
        for="<preg.name.toLowerCase>"><preg.etiqueta>:
      </label>
      <div class="col-sm-8">
        <_obtenerEntrada(preg, indice)>
      </div>
    </div>
    <{indice++;null}>
  <ENDFOR>
</fieldset>
...

def static _obtenerEntrada(Pregunta pregunta, int indice) {
switch (pregunta.tipo) {
  case ABIERTA: {
    '''<input type="text" class="form-control" id="<pregunta.name.toLowerCase>"
      placeholder="Ingrese <pregunta.etiqueta.toLowerCase>"
      [(ngModel)]="seccion.preguntaaplicacions[<indice>].respuesta">'''
  }
  case SELECCION: {
    '''<app-select-respuesta [id]="<pregunta.name.toLowerCase>"
      [pregunta]="seccion.preguntaaplicacions[<indice>]">
      </app-select-respuesta>'''
  }
  case FECHA: {
    '''<input type="date" class="form-control" id="<pregunta.name.toLowerCase>"
      placeholder="Ingrese <pregunta.etiqueta.toLowerCase>"
      [(ngModel)]="seccion.preguntaaplicacions[<indice>].respuesta">'''
  }
  default: {
    '''Tipo de entrada no implementada.'''
  }
}
}
}

```

Figura 3-16.: Transformación de una sección y sus preguntas para generar un archivo HTML.

La descripción de cada transformación se encuentra en la Tabla 3-8.

Tabla 3-8.: Descripción de las transformaciones.

Paquete	Nombre	Descripción
Raíz	InscripcionFormulario	Coordina las transformaciones de arquitectura, dominio y tecnología.
	Arquitectura	Usa las transformaciones ControladorBack, ComponenteFront y ServicioFront.
	Dominio	Usa las transformaciones Formulario, Entidad y SQL.
	Tecnología	usa las transformaciones tecnologíaBack y TecnologíaFront.
Dominio	Entidad	Genera las clases TypeScript para el modelo del front y las clases Java para el modelo del back.
	SQL	Genera un script para la creación y asociación de registros de base de datos de formulario, capítulos, secciones, preguntas y respuestas.
Back	ControladorBack	Genera la estructura de los controladores del back, y los controladores FormularioController e IncripcionController con la exposición de los servicios REST.
	RepositorioBack	Genera la estructura de los repositorios.
	ServicioBack	Genera la estructura de los servicios del back, y los servicios FormularioServices e IncripcionService con la implementación de los servicios.
Front	ComponenteFront	Genera la estructura de un componente de Angular, y genera los componentes base de Formulario, Capitulo y Seccion.
	ServicioFront	Genera la estructura del consumo de servicios, y los archivos para consumir los servicios de inscripción y formulario.
	Formulario	Genera el componente Angular (HTML y TypeScript) con la definición del formulario.
	Capítulo	Genera el componente Angular (HTML y TypeScript) con la definición de un capítulo.
	Sección	Genera el componente Angular (HTML y TypeScript) con la definición de una sección.
Tecnología	TecnologíaBack	Genera el archivo de configuración <i>application.properties</i> .
	TecnologíaFront	Genera los archivos de configuración <i>app.component.html</i> , <i>app-routing.module.ts</i> , <i>app.module.ts</i> y <i>environment.ts</i> .

3.6. Construcción del editor

La generación del editor se realizó usando Eclipse³ para obtener un plug-in que pueda ser usado en este mismo programa. Esto es posible porque los proyectos de Xtext pueden ser exportados como plug-in de Eclipse, como se muestra en la Figura 3-17. En la Figura 3-18 se muestra la instalación del plug-in generado.

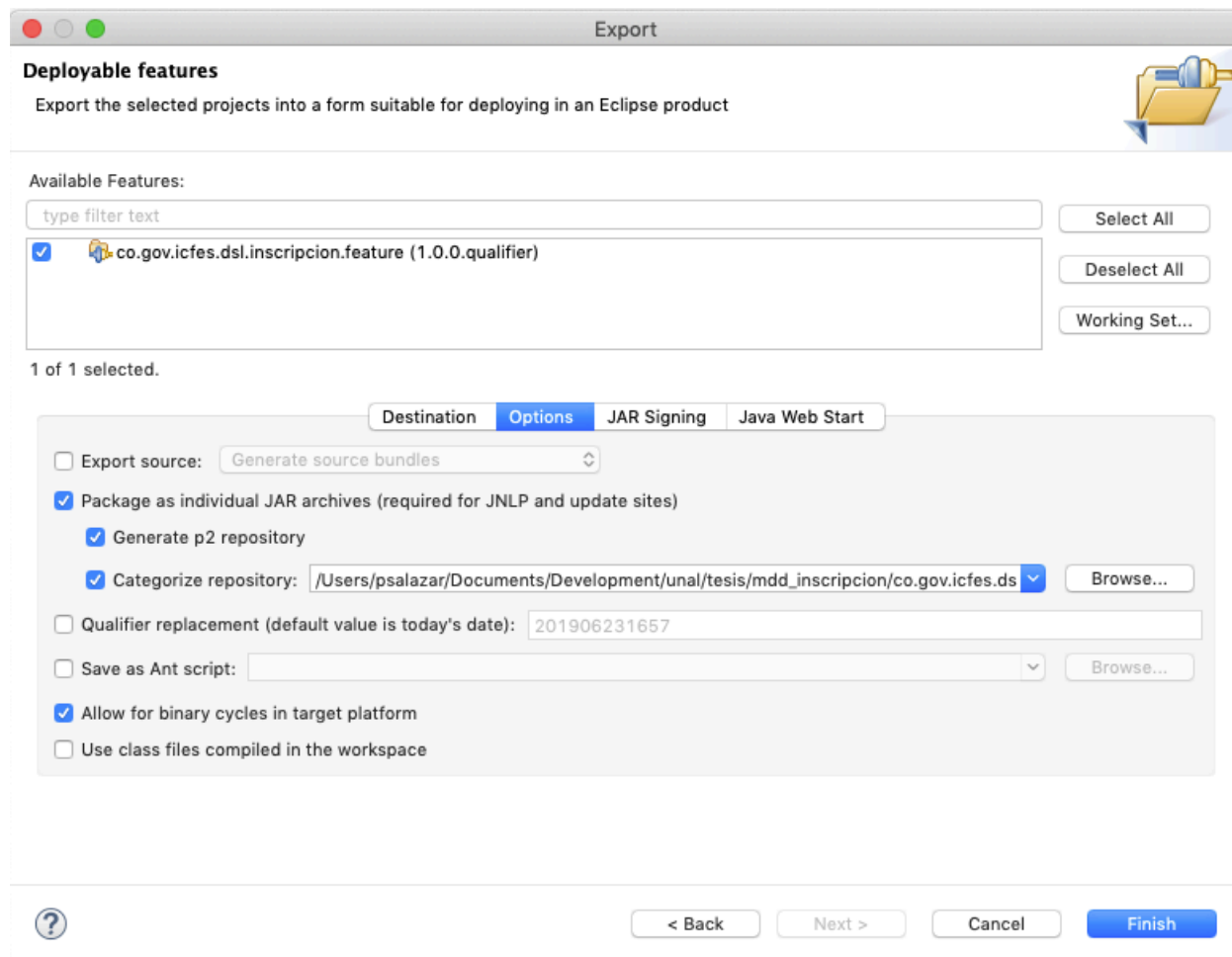


Figura 3-17.: Generación del plug-in con Eclipse.

³Eclipse Foundation. <https://www.eclipse.org/>

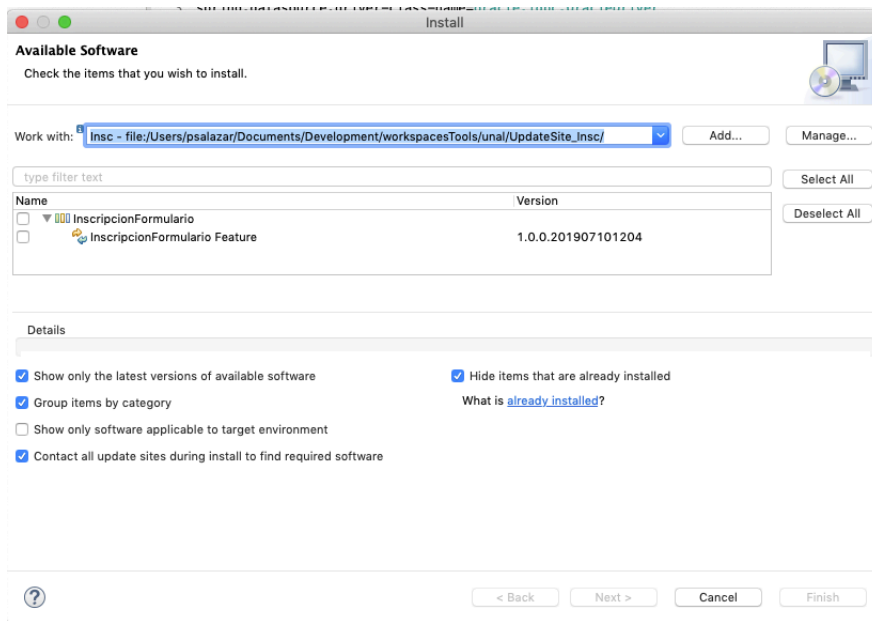


Figura 3-18.: Instalación del plug-in en Eclipse.

El plug-in identifica los archivos con la extensión *.inscripcionFormulario* para realizar validaciones, sugerencias y generar artefactos de software, como se muestra en la Figura 3-19.

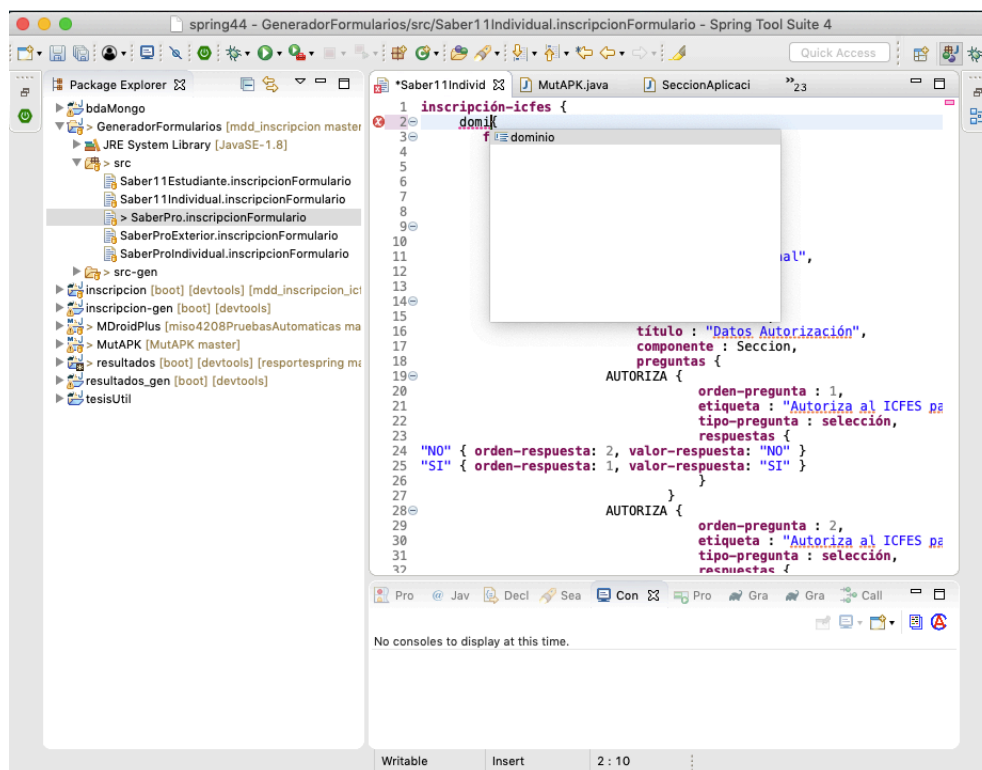


Figura 3-19.: Uso del plug-in mostrando sugerencias.

3.7. Generación de artefactos de software

La generación de artefactos de software crea los controladores, servicios, repositorios y la configuración de la capa de lógica de negocio. También genera el modelo, los componentes Angular, archivo de rutas, configuración de módulos y servicios de la capa de presentación. Por último, genera un script SQL para configurar el formulario, capítulos, secciones, preguntas y respuestas para el examen especificado. El modelo usado para la generación del formulario para Saber Pro - Individual se encuentra en el Anexo A. La evaluación de los resultados obtenidos en esta generación frente a la implementación de referencia se muestra en el capítulo 5. La página generada para diligenciar el formulario se muestra en la Figura 3-20.

Formulario inscripción - Saber Pro Individual

Información Personal | Información Académica | Información de Citación | Información Socioeconómica

Datos Autorización

¿Autoriza la entrega de su información a terceros (Personas naturales o jurídicas, entidades públicas o privadas) que otorgan estímulos o incentivos a los estudiantes con mejores resultados?:*

Datos personales

Primer Nombre:*	<input type="text" value="Ingrese primer nombre"/>	Segundo Nombre:*	<input type="text" value="Ingrese segundo nombre"/>
Primer Apellido:*	<input type="text" value="Ingrese primer apellido"/>	Segundo Apellido:*	<input type="text" value="Ingrese segundo apellido"/>
Tipo de Documento:*	<input type="text" value=""/>	Número de Documento de Identidad:*	<input type="text" value="Ingrese número de documento"/>
Nacionalidad:*	<input type="text" value="Ingrese nacionalidad"/>	Género:*	<input type="text" value=""/>
Fecha de Nacimiento:*	<input type="text" value="mm/dd/yyyy"/>		

Figura 3-20.: Resultado del formulario generado automáticamente para la inscripción de Saber Pro - Individual.

4. Construcción de una familia de productos de software

La construcción de una familia de productos de software se obtiene usando el DSL para definir un modelo por cada formulario de inscripción. A continuación, se presenta la estructura de cada formulario y el resultado obtenido en la generación.



4.1. Formulario de inscripción a Saber Pro - Exterior





Está conformado por un capítulo, cinco secciones, y dieciocho preguntas. El detalle del capítulo de información básica se muestra en la Tabla 4-1.

El formulario obtenido de la generación se muestra en la Figura 4-1.

Tabla 4-1.: Estructura del capítulo de información básica del formulario de inscripción para Saber Pro - Exterior.

Sección	Pregunta
Datos de autorización	¿Autoriza la entrega de información a terceros?
Datos personales	Primer nombre
	Segundo nombre
	Primer apellido
	Segundo apellido
	Tipo de documento
	Número de documento
	Nacionalidad
	Género
Datos de contacto	Fecha de nacimiento
	Departamento
	Municipio
	Dirección
	Correo electrónico
Datos académicos	Celular
	Universidad
Ciudad de presentación del examen	Programa
	Ciudad

icfes mejor saber  

Formulario inscripción - Saber Pro Exterior

Información básica

Datos autorización

¿Autoriza la entrega de su información a terceros (Personas naturales o jurídicas, entidades públicas o privadas) que otorgan estímulos o incentivos a los estudiantes con mejores resultados?:*

Datos personales

Primer Nombre:*	<input type="text" value="Ingrese primer nombre:"/>	Segundo Nombre:*	<input type="text" value="Ingrese segundo nombre:"/>
Primer Apellido:*	<input type="text" value="Ingrese primer apellido:"/>	Segundo Apellido:*	<input type="text" value="Ingrese segundo apellido:"/>
Tipo de Documento:*	<input type="text" value=""/>	Número de Documento de Identidad:*	<input type="text" value="Ingrese número de documento"/>
Nacionalidad:*	<input type="text" value="Ingrese nacionalidad:"/>	Género:*	<input type="text" value=""/>
Fecha de Nacimiento:*	<input type="text" value="Ingrese fecha de nacimiento:"/>		

Figura 4-1.: Página del formulario de inscripción para Saber Pro - Exterior, obtenida de la generación automática.

4.2. Formulario de inscripción a Saber 11 - Estudiante

Está conformado por cuatro capítulos, ocho secciones, y setenta y cinco preguntas. El detalle del capítulo de información personal se muestra en la Tabla 4-2, el de información académica y de citación en la Tabla 4-3, el de información socioeconómica en la Tabla 4-4 y el de antecedentes escolares en la Tabla 4-5.

El formulario obtenido de la generación se muestra en la Figura 4-2.

Tabla 4-2.: Estructura del capítulo de información personal del formulario de inscripción para Saber 11 - Estudiante.

Sección	Pregunta
Datos de autorización	¿Autoriza al ICFES para tratar sus datos personales con fines de estímulos o incentivos?
	¿Autoriza al ICFES para tratar sus datos personales con fines de investigación?
Datos personales	Primer nombre
	Segundo nombre
	Primer apellido
	Segundo apellido
	Tipo de documento
	Número de documento
	Nacionalidad
	Género
	Fecha de nacimiento
¿Pertenece usted a un grupo étnico minoritario?	
Discapacidades	¿Tiene alguna discapacidad?
Datos de contacto	Departamento
	Municipio
	Dirección
	Teléfono
	Celular
	Área de residencia
	Correo electrónico

Tabla 4-3.: Estructura del capítulo de información académica y de citación del formulario de inscripción para Saber 11 - Estudiante.

Sección	Pregunta
Datos académicos	¿Valor mensual de la pensión que paga actualmente?
	¿Cuántas veces ha presentado el examen Saber 11?
Datos para citación al examen	Primer nombre
	Departamento
	Ciudad de preferencia para la presentación del examen
	Zona de preferencia de presentación del examen

Tabla 4-4.: Estructura del capítulo de información socioeconómica del formulario de inscripción para Saber 11 - Estudiante.

Sección	Pregunta
Datos familiares	Nivel educativo de su padre
	Nivel educativo de su madre
	¿Cuántas hermanas y hermanos tiene usted en total?
	Estrato socioeconómico de su vivienda según recibo de energía eléctrica
	Nivel de SISBEN en el que está clasificada su familia
	¿Cuántas personas conforman el hogar donde vive actualmente, incluido usted?
	En total, ¿en cuántos cuartos duermen las personas de su hogar?
	¿Cuál es el material de los pisos que predomina en su vivienda?
	Teléfono fijo
	Servicio o conexión a internet
	Servicio cerrado de televisión (cable satelital o parabólica)
	Computador
	Reproductor de DVD
	Máquina lavadora de ropa
	Horno microondas
	Automóvil particular
	Horno eléctrico o a gas
	¿Cuántos libros hay en su casa o apartamento?
	¿Cuál es el total de ingresos mensuales de su hogar, en términos de salarios mínimos (SM)?
	¿Trabaja usted actualmente?
¿Usted recibe algún pago o salario por trabajar?	

Tabla 4-5.: Estructura del capítulo de información de antecedentes escolares y de citación del formulario de inscripción para Saber 11 - Estudiante.

Sección	Pregunta
Antecedentes escolares	¿Cuántos años de educación preescolar cursó?
	¿En qué año se matriculó por primera vez en el 1º grado de primaria?
	¿En qué año terminó y aprobó el 5º grado de primaria?
	¿En qué año se matriculó por primera vez en el 6º grado?
	¿Cuántos años lleva estudiando en el colegio actual?
	¿En cuántos colegios o establecimientos educativos diferentes estudió la primaria y secundaria?
	¿Alguna vez tuvo que retirarse del colegio (suspender estudio)?
	¿Cuál fue la razón principal por la que tuvo que retirarse o suspender sus estudios?
	¿Cuántos alumnos hay en total en su curso (salón)?
	En las últimas dos semanas de colegio, ¿cuántas veces faltó un día escolar completo?
	En su salón, usted considera que se encuentra
	¿Qué tipo de carrera desea estudiar cuando obtenga el título de bachiller?
	Municipio
	¿En cuál institución desea estudiar esa carrera?
	Por el prestigio de la institución
	El costo de la matrícula está a su alcance
	Por la ubicación y la cercanía
	¿Es la única que ofrece la carrera que desea?
	Brinda mejores/mayores oportunidades de empleo
	Amigos estudian en esa institución
	¿Cuál programa académico desea estudiar?
	La orientación vocacional/profesional recibida en el colegio
	La participación en uno de los talleres de encuentros juveniles del programa Buscando carrera del Ministerio de Educación Nacional
	La información o guía buscando carrera contenida en el portal www.colombiaprende.edu.co o en otros portales institucionales
	Interes personal
	Tradición familiar
	Posibilidad de mejorar la posición social y/o económica
Influencia de amigos, padres o familiares	

icfes mejorsaber **MINEDUCACIÓN**

f t RSS YouTube

Formulario inscripción - Saber 11 - Estudiante

Información Personal Información Académica y de Citación Información Socioeconómica Antecedentes Escolares

Datos Autorización

¿Autoriza al ICFES para tratar sus datos personales con fines de estímulos o incentivos?:*

¿Autoriza al ICFES para tratar sus datos personales con fines de investigación?:*

Datos personales

Primer Nombre::*

Segundo Nombre::*

Primer Apellido::*

Segundo Apellido::*

Tipo de Documento::*

Número de Documento de Identidad::*

Nacionalidad::*

Género::*

Fecha de Nacimiento::*

¿Pertenece usted a un grupo étnico minoritario?:*

Discapacidades

¿Tiene alguna discapacidad?:*

Figura 4-2.: Página del formulario de inscripción para Saber 11 - Estudiante, obtenida de la generación automática.

4.3. Formulario de inscripción a Saber 11 - Individual

Está conformado por tres capítulos, siete secciones, y cincuenta y dos preguntas. El detalle del capítulo de información personal se muestra en la Tabla 4-6, el de información académica y de citación en la Tabla 4-7 y el de información socioeconómica en la Tabla 4-8.

El formulario obtenido de la generación se muestra en la Figura 4-3.

Tabla 4-6.: Estructura del capítulo de información personal del formulario de inscripción para Saber 11 - Individual.

Sección	Pregunta
Datos de autorización	¿Autoriza al ICFES para tratar sus datos personales con fines de estímulos o incentivos?
	¿Autoriza al ICFES para tratar sus datos personales con fines de investigación?
Datos personales	Primer nombre
	Segundo nombre
	Primer apellido
	Segundo apellido
	Tipo de documento
	Número de documento
	Nacionalidad
	Género
	Fecha de nacimiento
	¿Pertenece usted a un grupo étnico minoritario?
Discapacidades	¿Tiene alguna discapacidad?
Datos de contacto	Departamento
	Municipio
	Dirección
	Teléfono
	Celular
	Área de residencia
	Correo electrónico

Tabla 4-7.: Estructura del capítulo de información académica y de citación del formulario de inscripción para Saber 11 - Individual.

Sección	Pregunta
Datos académicos	¿Está seguro de que el día del examen Usted ya habrá cumplido el requisito de ser graduado del bachillerato?
	¿Es usted validante ICFES?
	País
	Municipio
	Institución donde cursa o cursó sus estudios de bachiller
	¿Cuántas veces ha presentado el examen Saber 11?
	Año en que terminó el bachillerato
	Mes en que termino bachillerato
	Número acta de grado
	Fecha del acta de grado
Datos para citación al examen	¿Se encuentra usted privado de la libertad en un centro de reclusión?
	Ciudad de preferencia para la presentación del examen
	Zona de preferencia de presentación del examen

Tabla 4-8.: Estructura del capítulo de información socioeconómica del formulario de inscripción para Saber 11 - Individual.

Sección	Pregunta
Datos familiares	Nivel educativo de su padre
	Nivel educativo de su madre
	¿Cuántas hermanas y hermanos tiene usted en total?
	Estrato socioeconómico de su vivienda según recibo de energía eléctrica
	Nivel de SISBEN en el que está clasificada su familia
	¿Cuántas personas conforman el hogar donde vive actualmente, incluido usted?
	En total, ¿en cuántos cuartos duermen las personas de su hogar?
	¿Cuál es el material de los pisos que predomina en su vivienda?
	Teléfono fijo
	Servicio o conexión a internet
	Servicio cerrado de televisión (cable satelital o parabólica)
	Computador
	Reproductor de DVD
	Máquina lavadora de ropa
	Horno microondas
	Automóvil particular
	Horno eléctrico o a gas
	¿Cuántos libros hay en su casa o apartamento?
¿Cuál es el total de ingresos mensuales de su hogar, en términos de salarios mínimos (SM)?	

icfes mejor saber **MINEDUCACIÓN**

f t RSS YouTube

Formulario inscripción - Saber 11 Individual

Información Personal | Información Académica y de Citación | Información Socioeconómica

Datos Autorización

Autoriza al ICFES para tratar sus datos personales con fines de ESTIMULOS o INCENTIVOS:*

Autoriza al ICFES para tratar sus datos personales con fines de INVESTIGACIÓN:*

Datos personales

Primer Nombre:*

Segundo Nombre:*

Primer Apellido:*

Segundo Apellido:*

Tipo de Documento:*

Número de Documento de Identidad:*

Nacionalidad:*

Género:*

Fecha de Nacimiento:*

¿Pertenece usted a un grupo étnico minoritario?:*

Discapacidades

¿Tiene alguna discapacidad?:*

Figura 4-3.: Página del formulario de inscripción para Saber 11 - Individual, obtenida de la generación automática.

5. Evaluación

5.1. Análisis conceptual

Se generó una familia de productos de software a partir del modelo de cada formulario, lo que reduce el esfuerzo de construcción, parametrización y actualización de componentes de la aplicación. El esfuerzo de todo el proceso de desarrollo se transfiere en el uso de la metodología propuesta obteniendo como resultado un editor para usar el DSL que permite la generación de varios formularios. De los beneficios identificados por Pons [5] en MDD, en este trabajo se evidencian los siguientes:

- **Incremento en la productividad:** MDD disminuye los costos de desarrollo de software a través de la generación automática del código y otros artefactos de software a partir de los modelos. Una medida de productividad es el conteo de líneas de código de la aplicación [21, 22]. En la generación de una familia de productos de software usando MDD, se crean automáticamente varias líneas de código, en este caso, varios formularios de inscripción.
- **Adaptación a los cambios en los requisitos:** usando MDD, agregar o modificar una funcionalidad es una tarea sencilla porque el trabajo de automatización está hecho. Cuando se adiciona una nueva funcionalidad, solo se necesita desarrollar el modelo específico para esta. El resto de la información necesaria para generar los artefactos de software se implementan en las transformaciones y pueden ser reutilizados.
- **Consistencia:** MDD favorece la generación consistente de los artefactos. Esto reduce los errores al parametrizar los datos o en actualizaciones de componentes.
- **Reutilización:** MDD invierte en el desarrollo de modelos y transformaciones. La reutilización de artefactos ya probados incrementa la confianza en el desarrollo de nuevas funcionalidades.
- **Mejor comunicación con los usuarios:** los modelos omiten detalles técnicos de implementación que no son importantes para entender el comportamiento lógico del sistema. Por esta razón, los modelos son cercanos al dominio del problema, reduciendo la brecha semántica entre los conceptos que son comprendidos por los usuarios y el lenguaje en el cual se define la solución. Este factor influye positivamente en la producción de software alineado con los objetivos del negocio.

- **Mejor comunicación entre los desarrolladores:** los modelos favorecen la comprensión del sistema por parte de los desarrolladores. Esto permite discusiones más productivas y mejora los diseños. Adicionalmente, los modelos son parte del sistema y no solo documentación. De esta manera, siempre permanecen actualizados.
- **Los modelos son productos de larga duración:** los modelos son productos principales que consolidan lo que el sistema de la organización hace. Los modelos de alto nivel resisten los cambios a nivel de tecnología y solo cambian cuando se modifican los requisitos del negocio.

MDD presenta algunos desafíos como la integración de los componentes individuales obtenidos en la etapa de análisis de la metodología. Estos en general son fragmentos de código escritos manualmente que son incluidos después de la generación de automática. La integración y mantenimiento son un desafío al usar MDD que puede ser resuelto con patrones de software. Otro desafío es garantizar la calidad del código generado y las integraciones con los componentes individuales, puesto que al actualizar el modelo o las transformaciones se pueden inyectar errores.

5.2. Análisis práctico

La implementación de referencia del archivo HTML de la sección datos de autorización es mostrada en la Figura 5-1. La sección y las preguntas son elementos de la gramática definida en el DSL (Figura 5-2), que se abstraen de la implementación de referencia en las etapas de análisis y meta-modelado. La instancia del DSL que representa la sección de datos de autorización es mostrada en la Figura 5-3 y usando la transformación definida con Xtend (Figura 5-4), se obtiene el código generado mostrado en la Figura 5-5.

```

1  <fieldset id="fsDatosAutorizacion">
2    <legend>Datos Autorización</legend>
3    <div class="form-group required col-sm-6">
4      <label class="control-label col-sm-4" for="autoriza">¿Autoriza la entrega de su información a terceros (Pers
5      <div class="col-sm-8">
6        <app-select-respuesta [id]="autoriza" [pregunta]="seccion.preguntaaplicacions[0]"></app-select-respuesta>
7      </div>
8    </div>
9  </fieldset>
10

```

Figura 5-1.: Implementación de referencia del archivo HTML de la sección datos de autorización.

```

Seccion:
  name=ID '{ 'orden-sección' ':' orden=INT ','
  'título' ':' nombre=STRING ','
  'componente' ':' componente=[Componente] ','
  'preguntas' '{ (preguntas+=Pregunta)+ '}'
  '}'

Pregunta:
  name=ID '{ 'orden-pregunta' ':' orden=INT ','
  'etiqueta' ':' etiqueta=STRING ','
  'tipo-pregunta' ':' tipo=TipoEntrada
  (' ','respuestas' '{ (respuestas+=Respuesta)+ '}' )?
  '}'

Respuesta:
  name=STRING '{ 'orden-respuesta' ':' orden=INT ','
  'valor-respuesta' ':' valor=STRING
  '}'

enum TipoEntrada:
  ABIERTA='abierta' | SELECCION='selección' | FECHA='fecha';

```

Figura 5-2.: Elementos del DSL relacionados con las secciones y las preguntas.

```

inscripción-icfes {
  dominio{
    formulario-inscripción Inscripcion{
      título : "Saber Pro Individual",
      examen-aplicación : 1,
      población [individual],
      componente : Formulario,
      capítulos {
        InformacionPersonal {
          orden-capítulo: 1,
          título: "Información Personal",
          componente : Capitulo,
          secciones{
            DatosAutorización {
              orden-sección : 1,
              título : "Datos Autorización",
              componente : Seccion,
              preguntas {
                AUTORIZA {
                  orden-pregunta : 1,
                  etiqueta : "¿Autoriza la entrega de su información a
                    terceros (Personas naturales o jurídicas, entidades
                    públicas o privadas) que otorgan estímulos o incentivos
                    a los estudiantes con mejores resultados?",
                  tipo-pregunta : selección,
                  respuestas {
                    "NO" { orden-respuesta: 2, valor-respuesta: "No" }
                    "SI" { orden-respuesta: 1, valor-respuesta: "Si" }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}

```

Figura 5-3.: Fragmento de la definición del formulario usando el DSL para la inscripción al examen Saber Pro población individual.

```

def static _generarHTML(Seccion seccion) '''
<fieldset id="fs«seccion.name.toFirstUpper»">
  <legend>«seccion.nombre»</legend>
  «var indice = 0»
  «FOR preg : seccion.preguntas»
  <div class="form-group required col-sm-6">
    <label class="control-label col-sm-4"
      for="«preg.name.toLowerCase»">«preg.etiqueta»</label>
    <div class="col-sm-8">
      «_obtenerEntrada(preg, indice)»
    </div>
  </div>
  «{indice++;null}»
  «ENDFOR»
</fieldset>
...

def static _obtenerEntrada(Pregunta pregunta, int indice) {
  switch (pregunta.tipo) {
    case ABIERTA: {
      '''<input type="text" class="form-control" id="«pregunta.name.toLowerCase»"
        placeholder="Ingrese «pregunta.etiqueta.toLowerCase»"
        [(ngModel)]="seccion.preguntaaplicacions[«indice»].respuesta"'''
    }
    case SELECCION: {
      '''<app-select-respuesta [id]="«pregunta.name.toLowerCase»"
        [pregunta]="seccion.preguntaaplicacions[«indice»]">'''
    }
    case FECHA: {
      '''<input type="date" class="form-control" id="«pregunta.name.toLowerCase»"
        placeholder="Ingrese «pregunta.etiqueta.toLowerCase»"
        [(ngModel)]="seccion.preguntaaplicacions[«indice»].respuesta"'''
    }
    default: {
      '''Tipo de entrada no implementada.'''
    }
  }
}
}

```

Figura 5-4.: Transformación de una sección y sus preguntas para generar un archivo HTML.

Line	Original Implementation (Left)	Generated Code (Right)
1	<fieldset id="fsDatosAutorizacion">	<fieldset id="fsDatosAutorizacion">
2	<legend>Datos Autorización</legend>	<legend>Datos Autorización</legend>
3	<div class="form-group required col-sm-12">	<div class="form-group required col-sm-6">
4	<label class="control-label col-sm-10" for="autori">	<label class="control-label col-sm-4" for="autori">
5	<div class="col-sm-2">	<div class="col-sm-8">
6	<app-select-respuesta [id]="autoriza" [pregunta]>	<app-select-respuesta [id]="autoriza" [pregunta]>
7	</div>	</div>
8	</div>	</div>
9	</fieldset>	</fieldset>
10		

Figura 5-5.: Comparación de la sección de datos de autorización, entre la implementación de referencia (izquierda) y el código generado automáticamente (derecha).

Las diferencias mostradas en la Figura 5-5 se presentan porque las transformaciones están definidas para generar un formulario con preguntas a dos columnas y la implementación de referencia usa todo el espacio de la plantilla cuando la pregunta es muy larga. El resultado obtenido es similar

al de la implementación de referencia.

El código generado a partir del modelo definido para el formulario de inscripción para Saber Pro - Individual es comparado con el código obtenido en la implementación de referencia. Se usó la función *ratio*¹ de la librería *SequenceMatcher* de Python, que retorna una medida de similitud entre 0 y 1. La interpretación de la medida indica que los archivos son idénticos cuando es 1, mientras que si no tienen nada en común es 0. La medida de similitud se obtiene con la fórmula $2 * M / T$, donde T es el total de número de caracteres de ambos archivos y M es el número de coincidencias. La función *ratio* fue usada después de eliminar espacios de ambos archivos. Los resultados se muestran en la Tabla 5-1.

Tabla 5-1.: Similitud entre el código generado a partir del modelo y la implementación de referencia.

Capa	Paquete	Implementación de referencia	Código generado	Similitud
Lógica de negocio	Configuración	407	384	0.71
	Entidades	15792	13523	0.83
	Repositorios	1548	1548	0.99
	Servicios	6309	6308	0.99
	Controladores	2887	2886	0.99
Presentación	Configuración	5051	4638	0.93
	Entidades	3284	3693	0.84
	Servicios	1635	1653	0.99
	Formulario	2061	2069	0.99
	Capítulos	3380	3412	0.99
	Secciones	15498	15516	0.87
Datos	Script	0	168006	0

El código generado usando MDD es similar al de la implementación en una proporción grande, lo que reduce el esfuerzo de crear o modificar formularios de inscripción. Adicionalmente, reduce el error por inconsistencias en la parametrización del formulario en base de datos, porque el script generado es consistente con el código generado. En este sentido, el modelo se convierte en el principal artefacto de software.

Los artefactos de software generados a partir de las instancias del DSL de los formularios consiste en aplicación, que es el código generado en la capa de negocio y la de presentación, y los datos que es el SQL generado para la almacenar la configuración de los formularios en base de datos. La tabla 5-2 muestra el número de líneas usadas en el DSL para definir los formularios y el código generado para la aplicación y los datos.

¹<https://docs.python.org/3/library/difflib.html#difflib.SequenceMatcher.ratio>

Tabla 5-2.: Líneas de código (LC) generadas a partir de las instancias del DSL de los formularios.

Formulario	LC del DSL	LC de la Aplicación	LC de Datos
SaberPro - Individual	622	2016	4976
SaberPro - Exterior	325	1743	1368
Saber11 - Estudiante	1019	2225	9430
Saber11 - Individual	737	2038	6240

La desarrollo de software de los formularios se centra en el modelo, en la definición de los capítulos, secciones y preguntas, más que en los detalles de los lenguajes de programación. De esta manera se reducen los tiempos de desarrollo y los errores al realizar un cambio porque se mantiene la consistencia entre todas las capas de la implementación mediante la generación automática de los artefactos de software.

6. Conclusiones y trabajo futuro

6.1. Conclusiones

Se generó una familia de productos de software para el proceso de inscripción del ICFES usando MDD. El proceso consistió en usar la metodología propuesta con MDD y posteriormente generar la familia de productos con el editor del DSL obtenido.

La metodología contiene las etapas necesarias para analizar, diseñar y construir una herramienta útil para modelar los formularios y puede ser usada en otros dominios de negocio. La implementación de referencia es un punto de partida para identificar elementos del dominio, de la arquitectura y la tecnología, que luego son clasificados en componentes genéricos, frecuentes e individuales. El metamodelo es el primer elemento de abstracción importante porque permite validar tempranamente conceptos y relaciones. Posteriormente, diseñar el DSL como herramienta de definición de los modelos facilita la interacción y entendimiento de los formularios modelados. Diseñar las transformaciones es una actividad importante porque permite la generación automática de varios artefactos de software. Por último, el editor facilita la actividad de modelado al validar y sugerir fragmentos del DSL. La etapa de validación permite determinar el porcentaje de código generado para identificar los artefactos que no se están automatizando y tomar la decisión de automatizarlos o codificarlos posteriormente de forma manual.

Se modelaron los cuatro formularios para el proceso de inscripción: Saber Pro - Individual, Saber Pro - Exterior, Saber 11 - Estudiante y Saber 11 - Individual. Esta actividad concentra los esfuerzos en definir la composición y el comportamiento de cada formulario en términos del dominio de negocio. Luego de terminar el modelado, se inicia la generación automática de artefactos de software y este resultado es integrado con éxito a los proyectos de software que despliegan de forma consistente, generando una aplicación funcional en corto tiempo.

6.2. Trabajo futuro

Los siguientes aspectos pueden ser considerados como trabajo futuro: incluir validaciones de campos del formulario mediante expresiones regulares, modelar la dependencia de campos de selección (por ejemplo, los campos país, departamento y municipio), definir la dependencia de visualización de preguntas de acuerdo con la respuesta a otras preguntas. Por ejemplo, en la pre-

gunta ¿Pertenece usted a un grupo étnico?; si responde "Si", mostrar una pregunta de selección adicional para grupo étnico y en caso de responder "No", no se muestra la pregunta de selección adicional. También se podrían generar automáticamente los proyectos de Spring y Angular vacíos para integrar el código generado, y establecer un mecanismo usando patrones de software para integrar el código generado automáticamente con el código escrito de forma manual.

A. Anexo: Modelo definido en el DSL del formulario Saber Pro - Individual

En este anexo se muestran algunos fragmentos del modelado en el DSL del dominio, del formulario, las entidades, la arquitectura y la tecnología.

A.1. Fragmento del modelo del formulario

```
inscripción-icfes {
  dominio{
    formulario-inscripción Inscripcion{
      título : "Saber Pro Individual",
      examen-aplicación : 1,
      población [individual],
      componente : Formulario,
      capítulos {
        InformacionPersonal {
          orden-capítulo: 1,
          título: "Información Personal",
          componente : Capitulo,
          secciones{
            DatosAutorizacion {
              orden-sección : 1,
              título : "Datos Autorización",
              componente : Seccion,
              preguntas {
                AUTORIZA {
                  orden-pregunta : 1,
                  etiqueta : "¿Autoriza la entrega de su información a terceros
(Personas naturales o jurídicas, entidades públicas o privadas)
que otorgan estímulos o incentivos a los estudiantes
con mejores resultados?",
                  tipo-pregunta : selección,
                  respuestas {
                    "NO" { orden-respuesta: 2, valor-respuesta: "No" }
                    "SI" { orden-respuesta: 1, valor-respuesta: "Si" }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
  Datospersonales {
    orden-sección : 2,
    título : "Datos personales",
    componente : Seccion,
    preguntas {
      PRIMER {
        orden-pregunta : 1,
```

```

        etiqueta : "Primer Nombre",
        tipo-pregunta : abierta
    }
SEGUNDO {
    orden-pregunta : 2,
    etiqueta : "Segundo Nombre",
    tipo-pregunta : abierta
}
PRIMER {
    orden-pregunta : 3,
    etiqueta : "Primer Apellido",
    tipo-pregunta : abierta
}
SEGUNDO {
    orden-pregunta : 4,
    etiqueta : "Segundo Apellido",
    tipo-pregunta : abierta
}
TIPO {
    orden-pregunta : 5,
    etiqueta : "Tipo de Documento",
    tipo-pregunta : selección,
    respuestas {
        "NES" { orden-respuesta: 1741, valor-respuesta: "NES" }
        "PEP" { orden-respuesta: 1740, valor-respuesta: "PEP" }
        "TI" { orden-respuesta: 27, valor-respuesta: "TI" }
        "PE" { orden-respuesta: 32, valor-respuesta: "PE" }
        "PC" { orden-respuesta: 31, valor-respuesta: "PC" }
        "CR" { orden-respuesta: 30, valor-respuesta: "CR" }
        "CE" { orden-respuesta: 29, valor-respuesta: "CE" }
        "CC" { orden-respuesta: 28, valor-respuesta: "CC" }
    }
}
NUMERO {
    orden-pregunta : 6,
    etiqueta : "Número de Documento de Identidad",
    tipo-pregunta : abierta
}
NACIONALIDAD {
    orden-pregunta : 7,
    etiqueta : "Nacionalidad",
    tipo-pregunta : abierta
}
GENERO {
    orden-pregunta : 8,
    etiqueta : "Género",
    tipo-pregunta : selección,
    respuestas {
        "F" { orden-respuesta: 33, valor-respuesta: "Femenino" }
        "M" { orden-respuesta: 34, valor-respuesta: "Masculino" }
    }
}
FECHA {
    orden-pregunta : 9,
    etiqueta : "Fecha de Nacimiento",
    tipo-pregunta : fecha
}
}
}

```

A.2. Fragmento del modelo del entidades

```

entidades {
  entidad AgrupadorInscripcion [tabla="AGRUPADORINSCRIPCION"] {
    número agrupadorId {llave:verdadero columna : "AGRUPADOR_ID"}
    cadena tipo {columna : "TIPO"}
  }
  entidad Aplicacion [tabla="APLICACION"] {
    número aplicacionId {llave:verdadero columna : "APLICACION_ID"}
    cadena estado {columna : "ESTADO"}
    cadena nombre {columna : "NOMBRE"}
    ExamenAplicacion examenaplicacions UnoAMuchos [atributo:"aplicacion"]
  }
  entidad Capitulo [tabla="CAPITULO"] {
    número capituloId {llave:verdadero columna : "CAPITULO_ID"}
    cadena nombre {columna : "NOMBRE"}
  }
  entidad CapituloAplicacion [tabla="CAPITULOAPLICACION"] {
    número capapId {llave:verdadero columna : "CAPAP_ID"}
    número orden {columna : "ORDEN"}
    Capitulo capitulo MuchosAUno {columna : "CAPITULO_ID"}
    SeccionAplicacion seccionaplicacions UnoAMuchos [atributo:"capituloaplicacion"]
    {ordenar: 'orden' asc}
    FormularioAplicacion formularioaplicacion MuchosAUno
    {columna : "FORAP_ID" incluirFront:false}
  }
}

```

A.3. Modelo de la arquitectura

```

arquitectura {
  back{
    controladores{
      controlador FormularioController {
        servicios {
          servicio FormularioService {
            repositorios {
              repositorio FormularioAplicacionRepository [FormularioAplicacion]
              repositorio FormularioInscritoRepository [FormularioInscrito]
              repositorio InscpcionRepository [Inscpcion]
            }
          }
        }
      }
    }
    controlador InscpcionController {
      servicios {
        servicio InscpcionService {
          repositorios {
            repositorio InscpcionRepository [Inscpcion]
            repositorio AplicacionRepository [Aplicacion]
            repositorio ExamenAplicacionRepository [ExamenAplicacion]
          }
        }
      }
    }
  }
}
front{

```

```
componentes {
  componente Formulario { servicios { Formulario }}
  componente Capitulo { servicios { Formulario }}
  componente Seccion
}
serviciosFront {
  servicioFront Formulario
  servicioFront Inscripcion
}
}
```

A.4. Modelo de la tecnología

```
tecnología {
  configuración {
    back {
      propiedades {
        spring.datasource.url : 'jdbc:oracle:thin:@localhost:1521:xe'
        spring.datasource.driver-class-name : 'oracle.jdbc.OracleDriver'
        spring.datasource.username : 'icfes'
        spring.datasource.password : 'icfes'
        spring.jpa.database-platform : 'org.hibernate.dialect.Oracle10gDialect'
        spring.jpa.hibernate.ddl-auto : 'none'
      }
    }
    front {
      propiedades {
        urlApi : "http://localhost:8080"
      }
    }
  }
}
```

Bibliografía

- [1] Instituto Colombiano para la Evaluación de la Educación ICFES, “Misión y Visión del Icfes,” 2018.
- [2] Instituto Colombiano para la Evaluación de la Educación ICFES, “Guía de orientación Saber 11 para estudiantes,” pp. 1–41, 2018.
- [3] D. L. Parnas, “Software Product-Lines: What To Do When Enumeration Won’t Work,” in *International Workshop on Variability Modelling of Software-intensive Systems (Va-MoS’07)*, pp. 9–14, 2007.
- [4] L. M. Northrop, P. C. Clements, F. Bachmann, J. Bergey, G. Chastek, S. Cohen, P. Donohoe, L. Jones, R. Krut, R. Little, J. McGregor, and L. O’Brien, *A Framework for Software Product Line Practice, Version 5.0*. Software Engineering Institute | Carnegie Mellon, 2012.
- [5] C. Pons, R. Giandini, and G. Pérez, *Desarrollo de software dirigido por modelos: Conceptos teóricos y su aplicación práctica*. 2010.
- [6] T. Stahl and M. Völter, *Model-Driven Software Development: Technology, Engineering and Management*. John Wiley & Sons, 2006.
- [7] M. Voelter, S. Benz, C. Dietrich, B. Engelmann, M. Helander, L. Kats, E. Visser, and G. Wachsmuth, “DSL Engineering: Designing, Implementing and Using Domain-Specific Languages,” p. 558, 2013.
- [8] J. A. Cruz Castelblanco, “A modular model-driven engineering approach to reduce efforts in software development teams,” pp. 1–88, 2014.
- [9] J. A. Vergara-Vargas, “A model-driven deployment approach for applying the performance and scalability perspective from a set of software architecture styles,” 2017.
- [10] D. d. J. Martínez Acosta, “Herramienta para la generación automática del código fuente para aplicaciones con arquitectura modelo vista controlador (MVC) bajo desarrollo dirigido por modelos textuales (MDD),” no. Universidad Nacional de Colombia, p. 87, 2014.
- [11] D. Distanto, P. Pedone, G. Rossi, and G. Canfora, “Model-Driven Development of Web Applications with UWA, MVC and JavaServer Faces,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4607 LNCS, pp. 457–472, 2007.

-
- [12] J. Navón and P. Bustos, "Web application development: Java, .net and lamp at the same time," in *Lecture Notes in Computer Science* (G. M. Lowe D., ed.), vol. 3579, (Sydney), pp. 185–190, 2005.
- [13] J. Agüero, M. Rebollo, C. Carrascosa, and V. Julián, "Developing Virtual Organizations using MDD," in *CEUR Workshop Proceedings*, vol. 635, (Sevilla), pp. 130–141, CEUR-WS, 2009.
- [14] M. L. Bernardi, G. A. Di Lucca, and D. Distanto, "Model-driven fast prototyping of RIAs: From conceptual models to running applications," in *Proceedings of the 2014 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2014* (M. P. C. D. E. M. B. S. A. T. S. M. Mukherjea S. Krishnaswamy D., ed.), pp. 250–258, Institute of Electrical and Electronics Engineers Inc., 2014.
- [15] A. Schuler and B. Franz, "Rule-based generation of mobile user interfaces," in *Proceedings of the 2013 10th International Conference on Information Technology: New Generations, ITNG 2013*, (Las Vegas, NV), pp. 267–272, 2013.
- [16] J. Vergara-Vargas and H. Umana-Acosta, "A model-driven deployment approach for scaling distributed software architectures on a cloud computing platform," in *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pp. 99–103, IEEE, nov 2017.
- [17] K. J. Garcés Pernet and R. Casallas, *Administración de variabilidad en una línea de productos basada en modelos*. PhD thesis, Universidad de los Andes, Bogotá, 2007.
- [18] G. M. Arango Ríos, J. S. Montaña Ortega, and N. López, *MD-SPL CUIP2 : una línea de productos basada en modelos : versión 2.0*. PhD thesis, Universidad de los Andes, Bogotá, 2007.
- [19] M. Asadi, B. Mohabbati, N. Kaviani, D. Gašević, M. Bošković, and M. Hatala, "Model-driven development of families of Service-Oriented Architectures," *Proceedings of the First International Workshop on Feature-Oriented Software Development - FOSD '09*, p. 95, 2009.
- [20] M. Anastasopoulos and A. Balogh, "Model-driven development of Particle System Families," 2007.
- [21] N. E. Fenton and M. Neil, "Software metrics," in *Proceedings of the conference on The future of Software engineering - ICSE '00*, (New York, New York, USA), pp. 357–370, ACM Press, 2000.
- [22] L. M. Laird, M. C. Brennan, and IEEE Computer Society., *Software measurement and estimation : a practical approach*. Wiley-Interscience, 2006.