PA173-6-Genionic

Genionic: Automatización del desarrollo de aplicaciones móviles con el framework híbrido Ionic a partir de modelos ISML.

JUAN CARLOS ALCALÁ BUSTOS

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
MAESTRÍA EN INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
BOGOTÁ, D.C.
2017

PA173-6-Genionic

Genionic: Automatización del desarrollo de aplicaciones móviles con el framework híbrido Ionic a partir de modelos ISML.

Autor:

Juan Carlos Alcalá Bustos

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

MAGÍSTER EN INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

Director

Ing. Jaime Andrés Pavlich Mariscal Ph.D

Comité de Evaluación del Trabajo de Grado

Santiago Gil

Carlos Parra

Página web del Trabajo de Grado

http://pegasus.javeriana.edu.co/~PA173-6-genionic

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
MAESTRÍA EN INGENIERIA DE SISTEMAS Y COMPUTACIÓN
BOGOTÁ, D.C.
Noviembre, 2017

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
MAESTRÍA EN INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

Rector Magnífico

Jorge Humberto Peláez Piedrahita, S.J.

Decano Facultad de Ingeniería

Ingeniero Jorge Luis Sánchez Téllez

Director Maestría en Ingeniería de Sistemas y Computación

Ingeniera Angela Carrillo Ramos

Director Departamento de Ingeniería de Sistemas

Ingeniero Efraín Ortíz Pabón

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

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

AGRADECIMIENTOS

Doy gracias a Dios por brindarme la oportunidad de cursar este programa de Maestría, y por ser el guía de todas mis acciones.

Doy gracias a mis padres Elizabeth Bustos Ojeda y Gines Alcalá Lucas por su gran apoyo en el transcurso de mis estudios.

Agradezco a mi director de este trabajo de grado, Ing. Jaime Pavlich por su dedicación, apoyo y dirección durante el desarrollo de este trabajo de grado.

Agradezco a todos los profesores que con su dedicación, motivación y enseñanza dejaron en mí el conocimiento para el desarrollo de este trabajo de grado.

Juan Carlos Alcalá

Contenido

Listado de Tablas

Listado de Figuras

ABSTRACT

The increasing adoption of mobile devices by people around the world has led to an increment in the development of software for these devices, which has led companies developing these projects to require automation in development and offer low costs to be more competitive in this market.

As a support to companies, this project has developed the Genionic generator, which, using the model-driven engineering (MDE) approach, based on models in the ISML language and the hybrid Ionic framework, generates mobile applications for multiple operating systems. The use of this generator may improve development times, reducing the number of lines of code to be programmed, thus lowering the production costs of mobile applications.

RESUMEN

La creciente adopción de dispositivos móviles por las personas en todo el mundo ha generado un incremento en el desarrollo de software para estos dispositivos, lo que ha conllevado a las empresas que desarrollan estos proyectos a requerir una automatización en el desarrollo y ofrecer bajos costos de producción para ser más competitivos en este mercado.

Como apoyo a las empresas, este proyecto ha desarrollado el generador Genionic, el cual aplicando el enfoque de la ingeniería dirigida por modelos (MDE), a partir de modelos en el lenguaje ISML y el framework híbrido Ionic, permiten la generación de aplicaciones móviles para múltiples sistemas operativos móviles. El uso de este generador permitirá la optimización en los tiempos de desarrollo, al reducir la cantidad de líneas de código a programar, bajando así los costos de producción y la complejidad en el desarrollo de aplicaciones móviles.

RESUMEN EJECUTIVO

En el mundo se ha generado una creciente necesidad por el uso de dispositivos móviles, aplicaciones y todos los servicios asociados. Las ventas mundiales de teléfonos inteligentes alcanzaron 432 millones de unidades en el cuarto trimestre de 2016, con lo que aumentaron un 7 por ciento en comparación con el mismo período de 2015 [1]. La masificación del uso de los dispositivos móviles ha generado una mayor necesidad de producción de aplicaciones, y sumado al hecho de que existen varios sistemas operativos con mayor o menor presencia en el mercado [1] (Ej. Android, IOS, Windows), se genera una mayor dificultad para su producción. En cada una de las plataformas se requiere de un conocimiento específico, el uso de diferentes herramientas, siendo necesario la codificación de la misma aplicación en las diferentes plataformas para poder dar cobertura a todo el mercado. Esto implica mayores tiempos de producción y costos dependiendo del número de plataformas que se quieran alcanzar.

Cuando se aborda la generación de algún producto de software, inicialmente se genera una codificación de manera manual del sistema, a partir de requerimientos y un diseño especificado por los ingenieros de software [2]. Dentro de este proceso se pueden presentar varios riesgos como los requerimientos ambiguos o inconclusos [3] y la inclusión de errores humanos involuntarios, que pueden repercutir en mayores costos y productos defectuosos.

Una de las razones principales de la complejidad en desarrollo de aplicaciones es la falta de mecanismos adecuados de abstracción. Los desarrolladores tienen que abordar muchos detalles para desarrollar y desplegar correctamente las aplicaciones empresariales [5].

Una solución planteada a este problema es la ingeniería impulsada por modelos (MDE) [6]. MDE resuelve problemas y relacionados con el software en modelos, utilizando herramientas automáticas de generación de código para crear parte o todo el código que implementa esas soluciones [5].

En una implementación de MDE llevada a cabo durante los años 2012-2015 la Pontificia Universidad Javeriana y la empresa Heinsohn Business Technology (HBT) [4], desarrollaron los proyectos Lion y Lion2, los cuales lograron el aumento de la productividad dentro de la organización por medio de la automatización del desarrollo de las aplicaciones de software. Con ese proyecto, se construyó el ambiente de modelado MDE denominado ISML (Information Systems Modeling Language), el cual les permitió expresar modelos de aplicaciones en lenguaje textual, con una sintaxis muy simple, y luego transformar estos modelos en código fuente de tecnologías específicas.

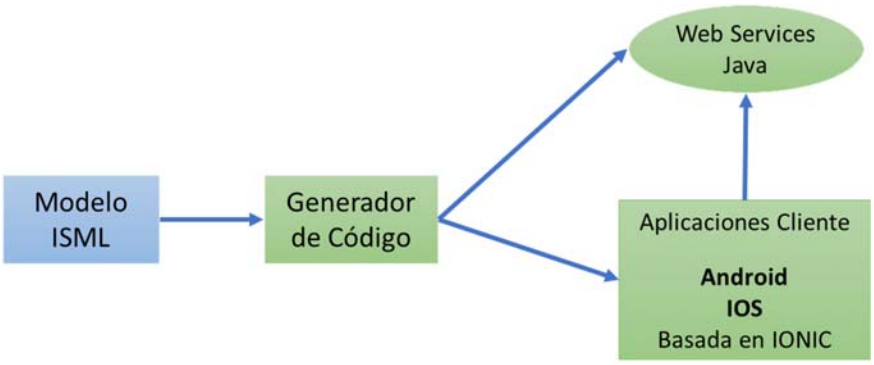


Figura 1. Modelo general del Generador

Basándose en los resultados de estos 2 proyectos mencionados, se desarrolló el generador de código Genionic (Figura1), que a partir de modelo especificados en ISML y el framework híbrido Ionic[8], permitirán la generación de aplicaciones móviles en los sistemas operativos móviles más usados en el mercado [1] (IOS, Android), sin la necesidad de tener que desarrollar la misma aplicación para cada uno de ellos.

En este proyecto se ejecutó como primera acción, la identificación de los elementos requeridos para la transformación hacia la arquitectura del framework Ionic, posteriormente se realizó la codificación de todos los elementos identificados, finalmente, se llevó a cabo la verificación de su funcionamiento con un ejercicio empresarial, que incluyó elementos como: cajas de texto, campos de selección, imágenes, tablas, menú, uso de mapas incluyendo el geoposicionamiento a partir de la ubicación del dispositivo móvil y la captura de imágenes desde la cámara del dispositivo, todos estos elementos fueron modelados en leguaje ISML y posteriormente con Genionic, se generó el código para el framework Ionic.

Con este proyecto, las empresas pueden disponer de un ambiente de generación de aplicaciones móviles que funcionen en los diferentes sistemas operativos móviles, que incluirá los elementos más usados en los desarrollos y que posteriormente pueden ser personalizados, ya teniendo una gran parte de trabajo realizado. Obteniendo como beneficio, el desarrollo de proyectos de una forma más ágil, estandarizada y con menores costos de producción.

1. INTRODUCCIÓN

En el mundo se ha generado una creciente necesidad por el uso de dispositivos móviles [1], aplicaciones y todos los servicios asociados, su uso creciente crea una mayor necesidad de producción de aplicaciones, y sumado al hecho de que existen varios sistemas operativos móviles con mayor o menor presencia en el mercado (Ej. Android, IOS, Windows), se genera una mayor dificultad para su producción. En cada una de las plataformas se requiere de un conocimiento específico, el uso de diferentes herramientas, siendo necesario la codificación de la misma aplicación en las diferentes plataformas para poder dar cobertura a todo el mercado. Esto implica mayores tiempos de producción y costos dependiendo del número de plataformas que se quieran alcanzar.

Por consiguiente, surgen dos problemas: el primero evidenciado, es la necesidad de aumento de la producción de aplicaciones móviles con un menor costo para aumentar capacidad de competir en el mercado y el segundo, la dificultad de producir la misma aplicación para los diferentes sistemas operativos móviles existentes.

En una implementación de MDE (*Model Driven Enginnering*) [6] llevada a cabo durante los años 2012-2015 la Pontificia Universidad Javeriana y la empresa Heinsohn Business Technology (HBT) [4], desarrollaron los proyectos Lion y Lion2, el objetivo de este proyecto innovador fue la automatización y reutilización de componentes ya exitosos dentro de la empresa, para el desarrollo de las aplicaciones de software. Con ese proyecto, se construyó el ambiente de modelado MDE denominado ISML (Information Systems Modeling Language), el cual les permitió expresar modelos de aplicaciones en lenguaje textual, con una sintaxis muy simple, y luego transformar estos modelos en código fuente de tecnologías específicas. Por consiguiente, como solución para el primer problema se usó ISML, intentado obtener con él, la reducción del tiempo de desarrollo y los costos asociados en la producción de aplicaciones móviles.

El framework Ionic ha sido seleccionado como la mejor opción dentro de un estudio [7] entre los frameworks de desarrollo móvil, para obtener una aplicación rápida, eficiente y con un alto grado de usabilidad, muy similar a las características de una aplicación nativa. Por tal razón, será utilizado para dar solución al segundo problema encontrado, ya que Ionic permite la generación de una aplicación móvil para diferentes sistemas operativos utilizando el mismo código fuente.

Este trabajo combina estas dos tecnologías, creando un transformador llamado Genionic, que a partir de un modelo expresado ISML genera un proyecto Ionic, el cual permite la generación de aplicación móvil para diferentes sistemas operativos móviles.

El presente documento se compone de los siguientes capítulos: capítulo 2, describe el proyecto, presentando sus objetivos y la metodología que se siguió en la ejecución del proyecto; capítulo 3, presenta un marco teórico que profundiza en los conceptos referentes al proyecto y capítulo 4 que presenta los trabajos relacionados.

Posteriormente, se presentan tres capítulos dedicados a describir el diseño y desarrollo realizado: la arquitectura del transformador Genionic; la arquitectura de la aplicación generada; la

prueba de concepto; el análisis de resultados. Finalmente, se presentan las conclusiones del trabajo.

2. DESCRIPCIÓN DEL PROYECTO

2.1. Objetivo general

Construir un generador de código que a partir de un modelo independiente de plataforma ISML, genere una aplicación móvil multiplataforma (IOS, Android, Windows Phone) por medio del framework Ionic.

2.2. Objetivos específicos

- Especificar los requerimientos del generador de código para aplicaciones móviles modeladas en el lenguaje ISML.
- Diseñar el generador de código para aplicaciones móviles modeladas en el lenguaje ISML.
- Desarrollar el generador de código, basado en la especificación del diseño de aplicaciones móviles modeladas en el lenguaje ISML.
- Probar el generador de código con un modelo en lenguaje ISML.

2.3. Metodología

El proyecto se llevó a cabo con una adaptación de la mitología DAD (*Disciplined Agile Delivery*) [10]. El cual es una metodología adaptable a las necesidades del proyecto, y posee un ciclo de vida orientado a los objetivos [11], las fases que lo componen se muestran la Figura 2.

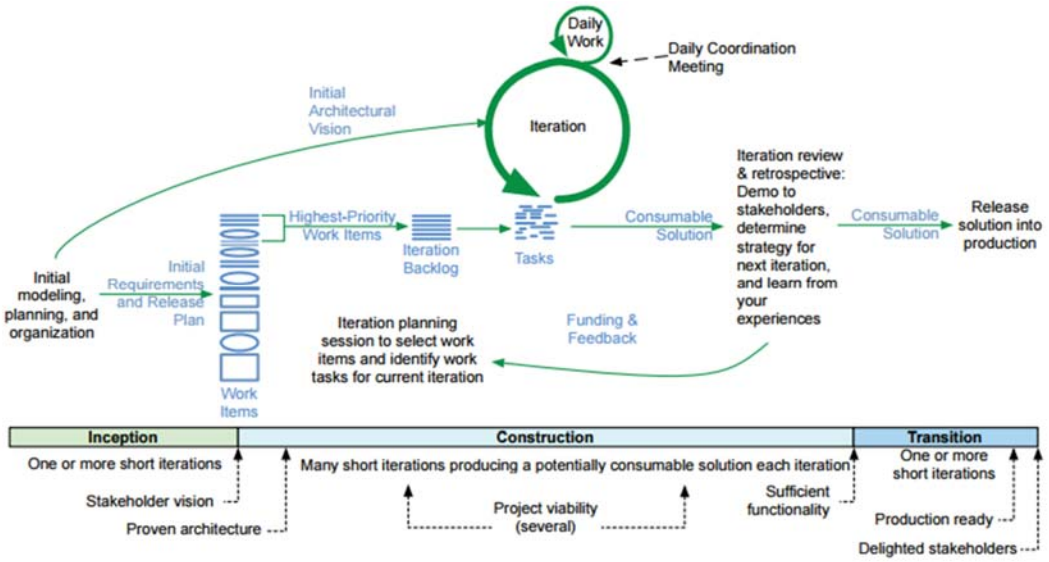


Figura 2. Ciclo de vida DAD

El ciclo de vida de DAD se divide en tres fases: concepción, construcción y transición.

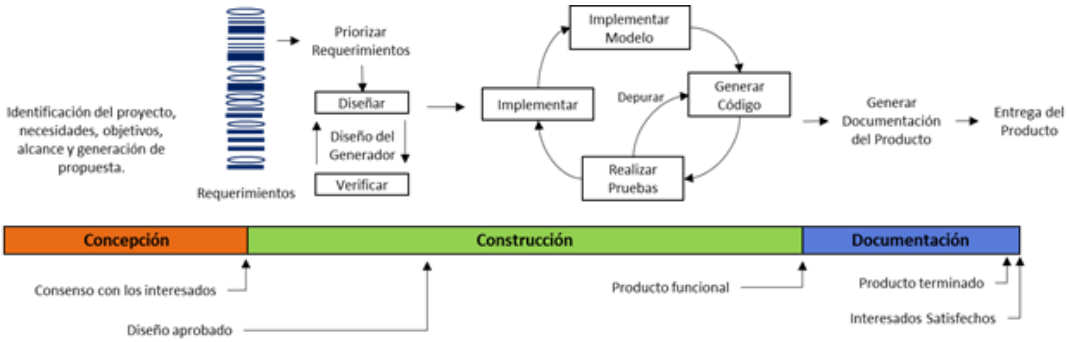


Figura 3. Ciclo de vida del Proyecto (Adaptado del modelo DAD)

Para este proyecto se tendrán en cuenta la fase de concepción y construcción, pero la fase de transición será reemplazada por una fase de documentación con el fin de que en esta fase se lleven a cabo los documentos que realizan la descripción del trabajo realizado.

2.3.1. Concepción

En esta fase se identificó la problemática existente en el mercado, específicamente a la complejidad del desarrollo software y la evolución constante de las tecnologías de desarrollo de software, por lo que sé decidió aportar en el proyecto MDE-ISML con la construcción de un generador adicional para suplir la necesidad creciente del mercado [13], de construir aplicaciones móviles que estén disponibles en los diferentes sistemas operativos.

Actividades

a. Selección del framework para tecnologías móviles.

   Una vez identificada la problemática, se debía establecer un framework con la capacidad de producir las aplicaciones móviles en los diferentes sistemas operativos, para ello se realizó un estudio en el mercado de las herramientas disponibles que cumplieran con los criterios:
   - Uso Libre
   - Soporte para Android, IOS y Windows Phone como mínimo.
   - Acceso a las diferentes funcionalidades del dispositivo móvil.

b. Revisión del material bibliográfico de ISML y Ionic.
c. Se establecieron los objetivos del proyecto.
d. Se estableció el alcance del proyecto.
e. Se llevo a cabo la construcción del documento de la propuesta.
f. Se construyo el documento de requerimientos.

Resultados

Los resultados de esta fase fueron: Documento de propuesta de trabajo de grado y documento de especificación de requerimientos.

2.3.2. Construcción

En esta fase se abordaron la priorización requerimientos, el diseño, la construcción del generador y pruebas.

Actividades

a. Priorización de requerimientos
b. Definición de arquitectura del proyecto Ionic.
c. Especificación del modelo ISML para el Frontend y Backend
d. Especificación de las reglas de transformación para Ionic.
e. Especificación de las reglas de transformación para el Web Services.
f. Implementación de plantillas de generación de código.
g. Depuración y validación de código mediante pruebas.

Resultados

Los resultados de esta fase fueron: documento de diseño, código fuente del generador de código y documento de las pruebas sobre el generador de código.

2.3.3. Documentación

Actividades

a. Elaboración del manual de usuario.
b. Elaboración del documento de memoria del Trabajo de Grado
c. Elaboración del Sitio web en Pegasus.

Resultados

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

3. MARCO TEÓRICO / ESTADO DEL ARTE

En esta sección se realiza la descripción las referencias teóricas y conceptos que se tuvieron en el desarrollo de este proyecto:

3.1. Ingeniería Dirigida por Modelos (MDE)

El Desarrollo de Software Dirigido por Modelos (En inglés "Model-Driven Development") es una disciplina que viene generando muchas expectativas como alternativa a los métodos convencionales de producción de software [18]. Es un paradigma que impulsa el uso de modelos y transformadores dentro la ingeniería de software [16].

"Un modelo es una abstracción de un sistema que usualmente representa una vista de un sistema; ésta es parcial y simplificada. Por lo tanto, es necesaria la creación de múltiples modelos para representar el sistema a estudiar" [19].

En la Figura 4, se muestra el ciclo de vida del desarrollo de software dirigido por modelos, en los cuales se identifican diferentes tipos de modelos [18]:

- Modelos con alto nivel de abstracción independientes de cualquier metodología computacional, llamados CIMs (Computational Independent Model).
- Modelos independientes de cualquier tecnología de implementación llamados PIMs (Platform Independent Model).
- Modelos que especifican el sistema en términos de construcciones de implementación disponibles en alguna tecnología específica, conocidos como PSMs (Platform Specific Model),
- Modelos que representan el código fuente en sí mismo, identificados como IMs (Implementation Model).

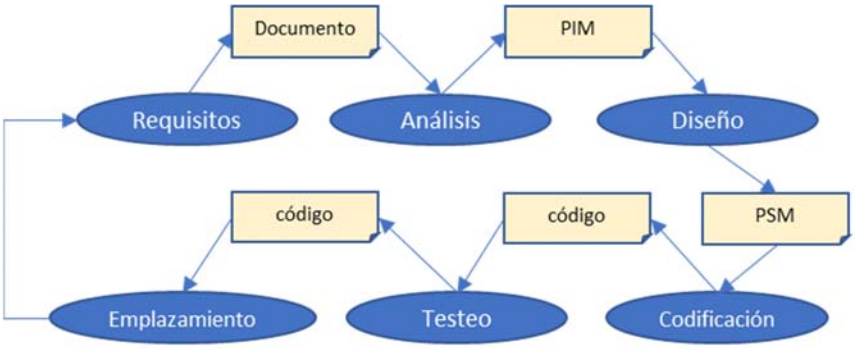


Figura 4. Ciclo de vida del desarrollo de software dirigido por modelos

Una de las principales diferencias entre el ciclo de vida MDE y el ciclo de vida del desarrollo del software tradicional, se da en que las transformaciones de modelo a modelo, o de modelo a código son hechas mayormente con intervención humana a diferencia de MDE, que siempre son realizadas por herramientas [18].

3.2. Ionic

Ionic es un framework de código abierto para el desarrollo de aplicaciones móviles híbridas. La versión original fue lanzada en 2013 y construida sobre AngularJS y Apache Cordova. Las versiones más recientes, conocidas como Ionic 2 o simplemente "Ionic", están basadas en Angular. Ionic proporciona herramientas y servicios para desarrollar aplicaciones móviles híbridas utilizando tecnologías web como CSS, HTML5 y Sass. Las aplicaciones pueden compilarse con estas tecnologías web y luego distribuirse a través de tiendas de aplicaciones nativas para ser instalado en dispositivos con el uso de Cordova. [8]

Dentro de sus características más relevantes se puede mencionar [8]:

- **Es de código y uso libre:** Es un proyecto totalmente libre y de código abierto bajo la licencia MIT, fue concebido para ser siempre de uso libre e impulsado por una comunidad que aporta constantemente en su evolución y mantenimiento.

- **Completamente multiplataforma:** Con Ionic se crean aplicaciones web nativas y compatibles con cada una de las tiendas de aplicaciones más importante del mercado, la cuales son generadas desde un mismo código. Ionic funciona y se ve muy bien en cualquiera de los sistemas operativa que se ejecute.

- **Plugins Nativos:** Ionic cuenta con más de 120 funciones nativas de los dispositivos móviles como Bluetooth, autenticación de huellas dactilares, HealthKit, cámara, geoposicionamiento y más con complementos de Cordova.

- **Amplia Documentación:** La documentación de Ionic cuenta con ejemplos de código y visualización reales, demostración del uso de componentes, guías y procedimientos muy sencillos de entender. Con la documentación de Ionic es fácil comenzar el desarrollo y la generación de aplicaciones móviles usando tecnologías familiares como HTML, JavaScript, CSS.

- **Rendimiento de las aplicaciones:** Ionic ha sido diseñado para funcionar y comportarse de forma excelente en los dispositivos móviles más recientes, implementando las mejores prácticas, como las transiciones aceleradas por hardware eficientes y los gestos táctiles optimizados.

- **Buen Diseño:** El diseño de Ionic tiene como objetivo ser limpio, simple y funcional. Ionic está diseñado para mostrarse de forma agradable en todos los dispositivos y sistemas operativos móviles actuales. Con componentes prefabricados, tipografía y un tema adaptable a cada plataforma.

Ionic se centra principalmente en la apariencia y la interacción de la interfaz de usuario con la aplicación [28]. Eso significa que no es un reemplazo para Cordova u otro. En cambio, Ionic se complementa perfectamente con Cordova para hacer uso de su tecnología ya exitosa y simplificar el desarrollo.

3.2.1. Angular

Es un framework de código abierto para el desarrollo de aplicaciones web programadas en lenguaje TypeScript, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es incrementar el desarrollo de aplicaciones basadas en navegador con el uso del patrón de arquitectura MVC (Modelo Vista Controlador), como esfuerzo para que el desarrollo sea más ordenado y se logren realizar pruebas más fácilmente [29].

Las características y beneficios de Angular [30] se pueden mencionar en tres categorías, en primer lugar, es *Multiplataforma,* ya que permite la creación de aplicaciones para Windows, Mac y Linux por medio de métodos de desarrollo web, con la posibilidad de acceder a las características nativas de cada uno de los sistemas operativos destino. En segundo lugar, su *Velocidad y Rendimiento*, Angular esta optimizado para que las aplicaciones carguen rápidamente con el uso de un enrutador de componentes, el cual proporciona una división de código para que sólo se cargue el código necesario para la vista solicitada y no todo el contenido de la aplicación. Adicionalmente, transforma las plantillas en código optimizado para JavaScript, ofreciendo todas las ventajas del código escrito de forma tradicional con las ventajas del uso de un framework. En tercer lugar, facilita la *Productividad,* a partir del Angular CLI permite un desarrollo rápido con sugerencias de código, detección de errores, permite agregar componentes automáticamente, facilita la realización de pruebas y permite un rápido despliegue.

3.2.2. Cordova

Apache Cordova es un framework de desarrollo móvil de código abierto, el cual permite usar tecnologías web estándar: HTML5, CSS3 y JavaScript para el desarrollo multiplataforma. Las aplicaciones se ejecutan encapsuladas dentro de una estructura dirigidas a cada plataforma [22]. A continuación, en la Figura 5 se muestra la arquitectura de un proyecto de Cordova.

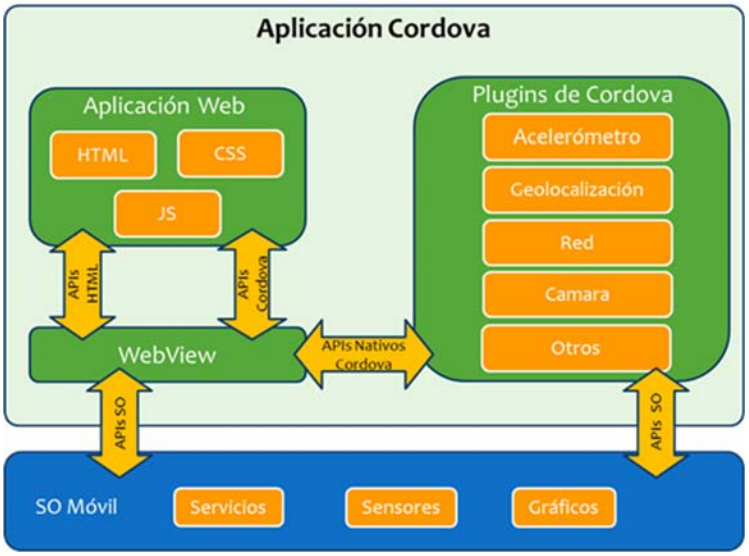


Figura 5. Arquitectura de Cordova.

WebView: Proporciona a la aplicación toda su interfaz de usuario. En algunas plataformas, también puede ser un componente dentro de una aplicación híbrida más grande.

Aplicación Web: Es donde reside el código de la aplicación. La aplicación en sí se implementa como una página web, por defecto un archivo local llamado index.html, que hace referencia a CSS, JavaScript, imágenes, archivos multimedia u otros recursos necesarios para que se ejecute. La aplicación se ejecuta en un WebView.

Plugins Cordova: Proporcionan una interfaz para que los componentes en código nativos puedan ser invocados desde JavaScript. Estos complementos proporcionan acceso a las capacidades del dispositivo, como batería, cámara, contactos, etc.

3.3. Java Empresarial JEE

Java Empresarial, proporciona un entorno de desarrollo basada en estándares para la generación de aplicaciones empresariales y web. Estas aplicaciones comúnmente son diseñadas bajo arquitecturas multicapa como se muestra en la Figura 6, las cuales se encuentran constituidas por una capa de presentación, negocio, y dominio (entidades respaldadas por una base de datos gracias a un framework de persistencia) [23].

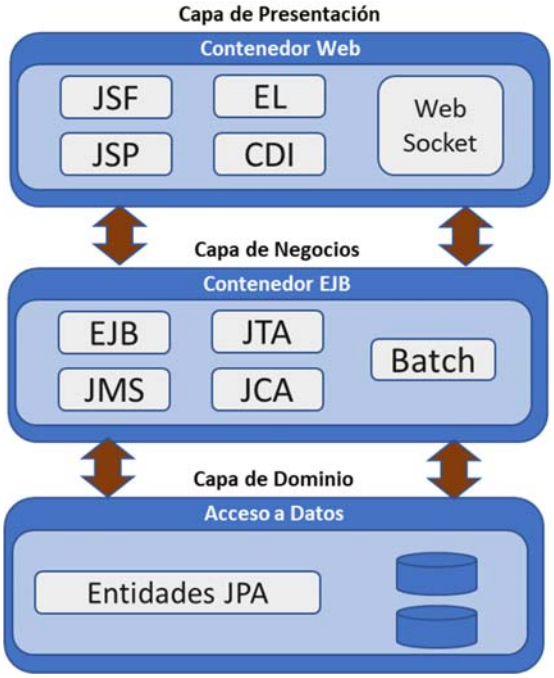


Figura 6. Arquitectura de Java EE.

Cada una de las capas mencionadas anteriormente, está soportada por un conjunto de APIs que define la plataforma Java EE además de un grupo de servicios transversales como invocación, inyección y administración de recursos que son desplegados en contenedores encargados de suplir soporte en tiempo de ejecución.

3.3.1. Invocación de servicios REST

Para la implementación de servicios web se hace uso REST (*Representation State Transfer*), donde los servicios son vistos como URIs (*Uniform Resource Identifiers*)[26].

Los principios de REST se definen como [31]:

- **Recursos direccionables**: cada recurso debe ser direccionable a través de una URI.
- **Orientación a la representación:** un recurso referido por un URI puede tener diferentes formatos de representación (por ejemplo, JSON, XML, etc.).
- **Apátrida**: Los servidores no pueden mantener el estado de una sesión de cliente. la comunicación entre el cliente y el servidor siempre contiene toda la información necesaria para realizar la solicitud.
- **Interfaz uniforme y restringida**: se usa un pequeño conjunto de métodos bien definidos para manipular recursos (es decir, métodos HTTP).

Los dos últimos principios se caracterizan porque cada solicitud se trata como una transacción independiente y solo debe basarse en el conjunto de operaciones del protocolo HTTP. Los métodos HTTP se usan en REST de la siguiente manera:

GET se usa para recuperar una representación de un recurso. Es una operación de solo lectura, idempotente y segura.

PUT se usa para actualizar una referencia a un recurso en el servidor y también es idempotente.

POST se utiliza para crear un recurso en el servidor según los datos incluidos en la solicitud del cuerpo. Es la única operación no volátil e insegura de HTTP.

DELETE se usa para eliminar un recurso en el servidor.

HEAD es similar a GET pero devuelve solo un código de respuesta y el encabezado asociado con la solicitud.

OPTIONS se utiliza para solicitar información sobre las opciones de comunicación del recurso direccionado.

3.3.2. Entidades persistentes JPA

Las aplicaciones Java EE regularmente usan persistencia, el API JPA (Java Persistence API), específicamente que establece como se puede aplicar la persistencia con las entidades en Java: objetos que representan una tabla y sus relaciones en una base datos [27].

Las entidades Java son administradas por el EntityManager, el cual es quien provee un conjunto de métodos:

Método	Descripción
persist	Permite almacenar una entidad en la base de datos.
find	Localiza una entidad en la base de datos por su llave primaria.
merge	Permite sincronizar o actualizar una entidad en memoria con su versión en la base de datos.
refresh	Asigna a una entidad en memoria los valores almacenados en base de datos, cancelando posibles modificaciones en memoria durante la transacción.
remove	Se encarga de eliminar una entidad de la base de datos.
flush	Permite aplicar inmediatamente a la base de datos modificaciones realizadas sobre las entidades.
Contains	Comprueba si una entidad está gestionada por el EntityManager devolviendo true o false según corresponda

Tabla 1: Métodos del EntityManager.

4. TRABAJOS RELACIONADOS

4.1. Trabajos relacionados con el enfoque MDE

Se ha llevado a cabo la búsqueda de trabajos relacionados con el enfoque MDE aplicados a tecnologías móviles, presentados a continuación:

- **A model driven approach for Android applications development [42]**

  Este proyecto presentó el desarrollo de aplicaciones de Android basado en el enfoque MDE. El modelo utilizado, estaba basado en UML y contaba con un generador para producir el código en Android, y su objetivo era facilitar y acelerar el desarrollo de aplicaciones móviles para esta plataforma. Para generar código, utilizaron la extensión de GenCode, la cual realizó una transformación automática de la clase UML. Como trabajo futuro, plantearon la optimización del código generado.

  Este proyecto solo tenía como alcance la producción aplicaciones en Android, y no tuvo en cuenta las demás plataformas móviles del mercado.

- **Modeling and generating the user interface of mobile devices and web development with DSL [37]**

  Este proyecto propone crear un enfoque que haga frente a la tarea tediosa de crear una misma aplicación para gran cantidad y variedad de tecnologías móviles existentes en el mercado, mediante el diseño de la interfaz de usuario para la creación de aplicaciones móviles y web, a partir de un Lenguaje Especifico de Dominio (DSL).

  Como resultado se presentó un estudio de la plataforma móvil Android y el Framework de Java Server Faces, donde adoptan un enfoque MDA para generar automáticamente la interfaz gráfica de usuario (código GUI) para las dos plataformas mediante la definición de una nueva GUI.

  El proyecto no tuvo como alcance la generación de código fuente, como la estructura de las clases, la gestión de las transiciones entre las pantallas de la GUI, los cuales fueron planteadas como trabajos futuros.

  El proyecto no llegó a generar ninguna aplicación para sistema operativo móviles.

- **A GUI modeling language for mobile applications [38]**

  Este Proyecto se centra en la generación de aplicaciones móviles que requieren interfaces de alta usabilidad, que a menudo se basan en pequeños conjuntos de patrones. El proyecto propuso un método para modelar interfaces móviles bajo el enfoque MDE. Como resultado presentó a MIM, un lenguaje de modelado de interfaz móvil diseñado teniendo en cuenta las últimas buenas prácticas de usabilidad de los proveedores de software para dispositivos móviles.

- **Generating Android graphical user interfaces using an MDA approach [39]**

  Este trabajo propuso un nuevo enfoque para el diseño de la interfaz de usuario, basado en el enfoque de Arquitectura Dirigida por Modelos (MDA), para la generación de aplicaciones móviles, específicamente aplicado a la plataforma Android. Para lo cual se usó un Lenguaje Específico de Dominio (DSL), como tecnología neutra que generaba código nativo para varias plataformas.

  El proyecto solo logró generar código nativo en la Plataforma Android, y tiene como trabajo futuro la implementación de transformadores para otras plataformas.

- **Model-driven development of mobile applications for Android and iOS supporting role-based app variability [40]**

  En este proyecto se llevó a cabo el desarrollo de un lenguaje de modelado bajo el enfoque MDE, para la generación de aplicaciones nativas en Android e iOS. En donde las aplicaciones generadas presentaban características flexibles en diferentes niveles de abstracción (modelado de elementos estándar y el modelado elementos personalizados), además consideraron el manejo de roles de usuario y el contexto. Para trabajos futuros plantean incorporar generadores de códigos para plataformas adicionales y incorporar extensiones de lenguaje hacia nuevas características como el manejo flexible de sensores y la realidad aumentada avanzada.

- **A model-driven approach to generate context aware applications [41**]

  Este documento muestra un lenguaje específico de dominio visual para modelar información contextual que se utilizará en el desarrollo de aplicaciones. Este lenguaje tuvo como objetivo generar un proyecto de Android, el proyecto generado, estaba configurado correctamente para el uso de middleware, incluido el acceso a la información contextual, permitiendo al desarrollador una mayor abstracción en el acceso al middleware. Como trabajos futuros tienen como objetivo lograr un mayor refinamiento del lenguaje, al incluir más acciones que puedan ser generadas automáticamente por el lenguaje.

4.2. Trabajos relacionados con ISML

Dentro de la Universidad Javeriana se han interesado por estudiar el enfoque MDE como herramienta de desarrolla de software, desarrollando trabajos de grado sobre este tema en niveles de maestría y doctorado. Estos proyectos han utilizado el lenguaje de modelado ISML como

herramienta para la generación de código para diferentes tecnologías, esto con el objetivo de reducir los tiempos y costos de desarrollo de productos de software. A continuación, se presentan los trabajos realizados:

MIDAS

MiDAS (Model-Driven Approach for Adaptive Systems), es un framework que ofrece un enfoque basado en modelos para el desarrollo adaptativo [32]. En concreto, MiDAS ofrece:

- Un nuevo lenguaje para especificar de manera formal y abstracta los requerimientos.
- Otro lenguaje para especificar el diseño de un sistema adaptativo.
- Un mecanismo para generar el código de un sistema adaptativo de manera automática, a partir de las especificaciones de requerimientos y diseño.

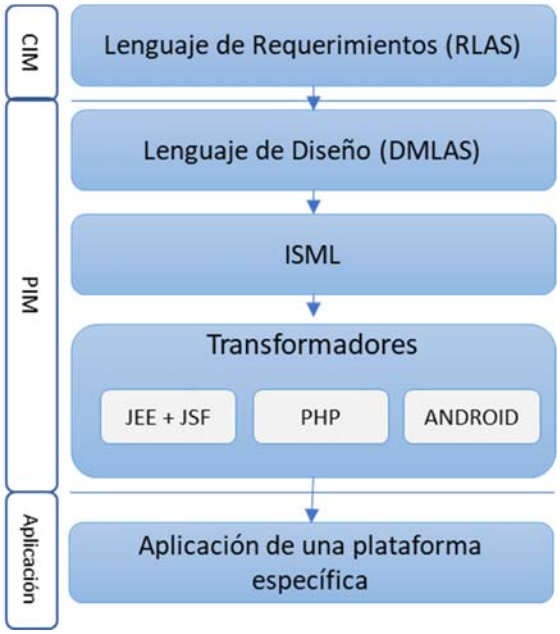


Figura 7. Componentes del proyecto MiDAS

En la Figura 8 se presenta una descripción gráfica de los componentes contemplados en el proyecto MiDAS. El proyecto parte de la definición del lenguaje RSLAS (Requirements Specification Language for Adaptive Systems) que permite expresar los requerimientos de una aplicación para sistemas adaptativos, éste es convertido al lenguaje de diseño DMLAS (Design Mode-ling Language for Adaptive Systems). Posteriormente, DMLAS es convertido a un modelo independiente de plataforma expresado en el lenguaje ISML. Finalmente, el modelo independiente de plataforma expresado en ISML es convertido a un lenguaje de programación específico a través de un transformador.

LITYERSES

Este proyecto desarrolló un transformador denominado Lityerses, el cual permitió generar aplicaciones móviles con componentes adaptativos, fue ejecutado en dos frentes de trabajo: el primero fue la exploración de los tipos de requerimientos de software adaptativo que podían ser

expresados en un lenguaje de modelado y el segundo, la construcción de un transformador que convertía un modelo independiente de plataforma, expresado en *ISML*, a un proyecto Android [33].

En la Figura 9, se presenta el modelo de transformación ejecutado en este proyecto, el cual tenía como entrada un modelo textual expresado en *ISML*, como salida una aplicación móvil para dispositivos Android y un proyecto de servicios web.

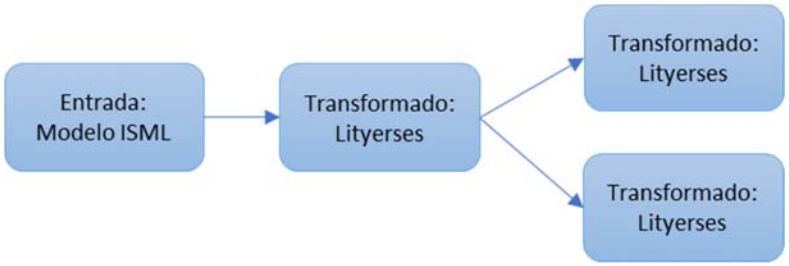


Figura 8. Modelo de transformación general

El transformador *Lityerses* contempló herramientas adaptativas, denominadas "componente adaptativo", las cuales fueron basadas en el análisis del perfil de usuario y del contexto, construido para aplicaciones móviles adaptativas genéricas. Los componentes adaptativos desarrollados fueron basados en:

- Priorización del contenido según la ubicación geográfica.
- Priorización del contenido según uso.
- Ajuste del contenido según características de conexión.
- Visualización de contenido en diferentes formatos.
- Realce de componentes según la prioridad.
- Filtración de contenido según las preferencias del usuario.

ANCHURUS-GEN

Anchurus-GEN fue generado como una alternativa a generadores de código ya existentes que permitían la generación de código en PHP desde un modelo de alto nivel, multiplataforma y gratuito. Esta herramienta fue construida utilizando la plataforma Eclipse, mediante el plugin Xtext, el cual le permitió generar páginas, migraciones, modelos, controladores y el archivo de rutas propios de Laravel como framework PHP, a partil de modelos ISML. Y como forma de validación del correcto funcionamiento del generador, se ejecutaron pruebas con un modelo ISML y se probó el correcto funcionamiento de la aplicación generada [34].

ZOE-GEN

Este proyecto tuvo como objetivo la construcción del transformador ZOE-GEN, el cual a partir de modelos especificados en ISML generaba código de Java EE. Se baso en las actividades de trabajo futuro del proyecto Lion2 [35].

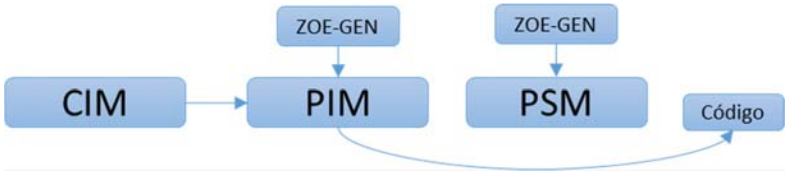


Figura 9. Arquitectura MDA

El proyecto tuvo como referencia el estándar MDA mostrado en la Figura 10. En donde ISML como modelo independiente de plataforma (PIM), permitió el uso de modelos que posteriormente fueron transformados en código específico para la plataforma de destino, en este caso Java EE bajo el marco de trabajo Primefaces.

El transformador ZOE-GEN fue implementado como parte del framework MiDAS (Model-Driven Approach for Adaptive Systems). El aporte específico para este framework, se vio reflejado en la construcción de componentes que facilitaron la parametrización para el desarrollo adaptativo, tales como: mapas, plantillas, paneles que resaltaban su ubicación y componentes que soportaban distintos medios como: videos y audios. Además, se presentaron servicios que facilitaban la construcción de sentencias SQL para facilitar la presentación de la información en componentes gráficos y servicios.

GENGULAR

Este proyecto construyó un transformador de código denominado Gengular, que a partir de modelos definidos en lenguaje ISML, permitió la creación de aplicaciones alineadas con el stack tecnológico HTML5-AngularJS-JEE7 y el paradigma de arquitectura de aplicaciones web SPA (Single-Page Application). La implementación de este paradigma les permitió a los usuarios interactuar con aplicaciones similares a aplicaciones nativas, mejorando sustancialmente el rendimiento de aplicaciones web mediante técnicas de diseño de cargue parcial de páginas, que eran compatibles tanto para PC como para dispositivos móviles [37].

Adicionalmente el proyecto contribuyó con la recuperación de componentes que son valiosos para cualquier empresa y que para Heinsohn Business Technology (HBT) (tomado como caso de estudio), representaban activos tecnológicos y un medio de automatización en el desarrollo de software mediante la reutilización en los proyectos de componentes útiles de seguridad, manejo de menús, auditoría, notificaciones, procesamiento de archivos, etc.

Con este proyecto, la empresa caso de estudio puedo disponer de un ambiente de generación de aplicaciones web que funcionaban con la eficiencia de las últimas tecnologías SPA y adicionalmente, de modelos de componentes empresariales que se transformaban en código fuente de esa misma tecnología para luego acoplarse a una aplicación previamente generada.

5. DESARROLLO DEL PROYECTO

En este capítulo se presenta las definiciones, artefactos y actividades relevantes realizadas en la ejecución del proyecto.

5.1. Tecnología para el Desarrollo de la Aplicación Móvil.

Con el fin de seleccionar la tecnología más adecuada para el desarrollo de la aplicación móvil, se llevó a cabo un análisis de los diferentes métodos de desarrollo y sus características asociadas. Posteriormente se analizaron los framework asociados al método seleccionado, que ofrecieran el mayor número de ventajas en la reducción de tiempo y costos de desarrollo, garantizando la calidad del producto.

En el mercado existen muchas maneras de realizar aplicaciones móviles, para las cuales existen muchos enfoques, los cuales han sido clasificados en 3 métodos [20]:

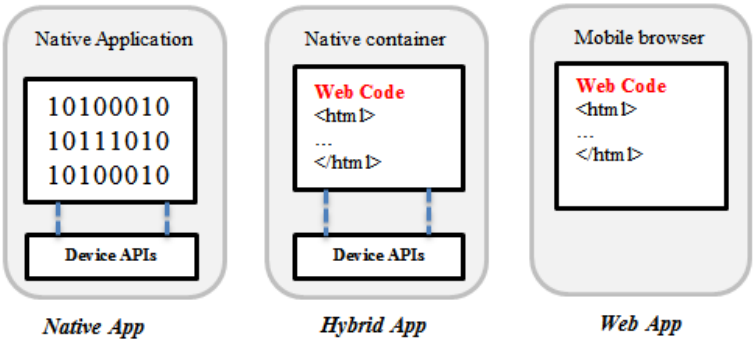


Figura 10. Métodos de desarrollo móvil

Aplicación Nativa: Son aplicaciones que se construyen en el lenguaje soportado por plataforma específica, por lo que tiene acceso a los IDE, que proporciona las mejores herramientas para el desarrollo, así como una rápida depuración del proyecto. Adicionalmente, tienen un mayor rendimiento, toman la apariencia nativa y tienen acceso total a las capacidades de hardware del dispositivo.

Aplicación Web: Son aplicaciones desarrolladas con tecnologías Web, normalmente HTML5, JavaScript y CSS. Son fáciles de desarrollar, pero cuentan con poco acceso a características de hardware del dispositivo.

Aplicación Híbrida: Son una combinación entre el desarrollo web y desarrollo nativo. Tienen algunas diferencias en funcionamiento con las aplicaciones nativas. A pesar de que son compiladas como aplicaciones nativas, no son aplicaciones nativas y no tienen el mismo rendimiento.

	Nativa	Híbrida	Web
Acceso al hardware	Total	Total	Parcial
Velocidad	Muy rápida	Rápida	Rápida
Costo de desarrollo	Alto	Razonable	Razonable
AppStore	Si	Si	No
Proceso de aprobación	Obligatorio	Gastos indirectos bajos	No
Calidad gráfica	Excelente	No es tan bueno como aplicaciones nativas	Muy bueno
Calidad de las aplicaciones	Alto	Medio a Bajo	Medio
Seguridad	Alta	No muy buena	Depende del Navegador
Usuarios potenciales	Limitado a una plataforma móvil específica	Alta - llega a los usuarios de diferentes plataformas.	Máximo incluyendo teléfonos inteligentes, tablets y otros teléfonos con funciones.
Acceder a funciones específicas del dispositivo	Alta	Media	Baja
Lenguaje de desarrollo	Solo nativo	Nativo y web o web solo	Web solo
Herramientas necesarias para aplicaciones multiplataforma	Objective-C, Java, C, C++, C#, VB.net	HTML, CSS,JavaScript, Mobile development framework (like Ionic)	HTML, CSS, JavaScript

Tabla 2. Métodos de desarrollo y sus características [20]

Para el análisis se tuvieron en cuentas los siguientes frameworks según su enfoque:

Framework	**Url**
Nativo	
Appcelerator Titanium Mobile	http://www.appcelerator.com/
Xcode	https://developer.apple.com/xcode/ide/
Android Studio	https://developer.android.com
Visual Studio Express	https://developer.microsoft.com
Gideros	http://giderosmobile.com/
Moai	http://www.getmoai.com
NeoMAD	http://neomades.com
Rhodes	http://www.rhomobile.com/
V-Play	http://www.v-play.net
Web	
CNET iPhone UI	http://code.google.com/p/ciui-dev/
DHTMLX Touch	http://www.dhtmlx.com/touch/
Handheld Designer	http://handhelddesigner.com
iUI	http://www.iui-js.org/documentation.html
JQ Touch	http://jqtouch.com/
jQuery Mobile	http://jquerymobile.com/
Mobify.js	http://www.mobifyjs.com

qooxdoo	http://qooxdoo.org
Sencha Touch	http://www.sencha.com/products/touch/
Wink	http://www.winktoolkit.org
Zepto.js	http://zeptojs.com/
Híbrido	
Apache Cordova	https://cordova.apache.org/
Agate	http://applusform.com/en
AppGyver Steroids	http://www.appgyver.com/steroids
Application Craft	http://www.applicationcraft.com
ChocolateChip-UI	http://www.chocolatechip-ui.com
eMobc	http://www.eMobc.com
Ionic	http://ionicframework.com/
Monocross	http://monocross.net/
Onsen UI	http://www.onsen.io/
PhoneGap	http://www.phonegap.com/
ViziApps	http://viziapps.com
AppConKit	http://www.weptun.com
Xamarin	http://xamarin.com/

Tabla 3. Métodos y sus Frameworks

Para las aplicaciones híbridas solo se tuvieron en cuenta los frameworks que pueden generan código para Android y IOS, ya que actualmente estas plataformas representan el 99% del mercado [1].

Para la selección del método a aplicar se tuvo en cuenta la necesidad de generación de código a múltiples plataformas y el esfuerzo requerido para este fin, adicionalmente, según una investigación realizada [7], se muestra que algunos framework híbridos ofrecen resultados muy similares a las aplicaciones nativas a un menor costo y con un menor esfuerzo en su desarrollo.

Adicionalmente, el desarrollo híbrido se ha convertido en uno de los principales métodos de desarrollo para aplicaciones móviles. Gartner predecía que para finales de 2016 el 50% de las aplicaciones serían desarrolladas con el método híbrido [21].

Por las consideraciones anteriormente mencionadas se ha tomado la decisión de continuar con la selección del framework solo con los métodos híbridos. Se ha realizado una comparación de los diferentes frameworks según el acceso a características de hardware y su tipo de licencia de uso, los resultados pueden ser vistos en la Tabla 4:

Framework	Libre	Acelerómetro	Cámara	Brújula	Conexión	Contactos	Archivos	GPS	Notificaciones	Almacenamiento	Gestos Multitáctil	SMS	Bluetooth	NFC	Vibración
Apache Cordova	si	si	si	si	si	si	si	si	si	si	si	si	no	si	si
Agate	si	si	si	si	si	si	si	si	si	si	si	si	no	no	si
AppGyver Steroids	si	plugin	plugin	plugin	plugin	plugin	plugin	plugin	plugin	plugin	plugin	plugin	plugin	plugin	plugin
Application Craft	si	plugin	plugin	plugin	plugin	plugin	plugin	plugin	plugin	plugin	no	no	no	no	plugin
ChocolateChip-UI	si	no	no	no	no	no	no	no	no	no	si	no	no	no	no
eMobc	si	no	si	si	si	si	no	si	si	si	si	no	no	si	si
ionic	si	si	si	si	si	si	si	si	si	si	si	no	no	si	si
Monocross	si	si	si	si	si	si	si	si	si	si	si	si	no	si	si
Onsen UI	si	si	si	si	si	si	si	si	si	si	si	no	no	no	si
PhoneGap	si	si	si	si	si	si	si	si	si	si	si	si	no	si	si
ViziApps	si	si	si	si	si	si	si	si	si	si	si	si	no	si	si
AppConKit	si	si	si	si	si	si	si	si	si	si	si	si	plugin	si	si
Xamarin	no	si	si	si	si	si	si	si	si	si	si	si	si	si	si

Tabla 4. Frameworks con su tipo de uso y acceso a características nativas del dispositivo

Se estableció como siguiente criterio de selección que el framework fuese de uso libres y que adicionalmente contara con acceso al mayor número de funciones del dispositivo móvil. Obtenido como resultado: Apache Cordova, Ionic y PhoneGap. Ya que Apache Cordova es la versión de código abierto de PhoneGap se continua solo con Apache Cordova y Ionic.

Como Ionic se centra en la apariencia y la interacción de la interfaz de usuario con la aplicación [28], complementándose con el uso de Apache Cordova para hacer uso de su tecnología, se seleccionó el Framework Ionic como la mejor opción para el desarrollo del proyecto.

5.2. Requerimientos del Sistema

A continuación, se especificarán los requerimientos funcionales y no funcionales del generador Genionic. Cada requerimiento cuenta con número del requerimiento, tipo de requerimiento y descripción.

# Req	Tipo	Descripción
1	Funcional	El transformador debe genera código en las tecnologías destino IONIC-JEE7. Para ello deberá tomar como insumo modelos construidos en el lenguaje ISML.
2	Funcional	El transformador debe generar un módulo Frontend en el cual irán los artefactos asociados a capa de presentación.
3	Funcional	El transformador debe generar un módulo backend en tecnología Java EE (Web Services) que supla las necesidades de la lógica de negocio.
4	Funcional	El transformador debe generar una serie de componentes gráfico como: campos de texto, botones, listas, imágenes y tablas. Estos deben tener propiedades para especificar etiquetas, valores y propiedades propias del framework de presentación.
5	Funcional	El transformador debe permitir la generación de código que permita la incorporación de plugins de Geoposicionamiento y Cámara para la aplicación generada.
6	No Funcional	El código generado por el trasformador debe ser compatible para la generación de aplicaciones que funcionen en diferentes sistemas operativos móviles (Android, IOS, Windows Phone)
7	No Funcional	El transformador debe generar los artefactos en la tecnología destino en un tiempo razonable, no mayor a 1 minutos.
8	No Funcional	Se debe realizar las pruebas de software del transformador como de los componentes generados.
9	No Funcional	Se debe crear un manual de usuario que detallen la manera en que puede ser utilizado el transformador.

10	No funcional	El sistema incluirá un procedimiento de autorización de usuarios, en el cual los usuarios deben identificarse usando un nombre de usuario y contraseña. Sólo los usuarios autorizados de esta forma podrán acceder a los datos del sistema.
11	No funcional	Los permisos de acceso al sistema podrán ser cambiados solamente por el administrador de acceso a datos.
12	No funcional	Todas las comunicaciones entre el servidor y la aplicación deben ser transportadas por un canal seguro.
13	No funcional	La base de datos debe respaldarse cada 24 horas. Los respaldos deben ser almacenados en una localidad segura distinta en la que reside el sistema.

Tabla 5. Requerimientos del sistema

5.3. Diseño del Generador Genionic

El sistema presentado corresponde a un transformador que convierte un modelo expresado en ISML en una aplicación móvil. En la Figura 12 se presenta el proceso descrito.

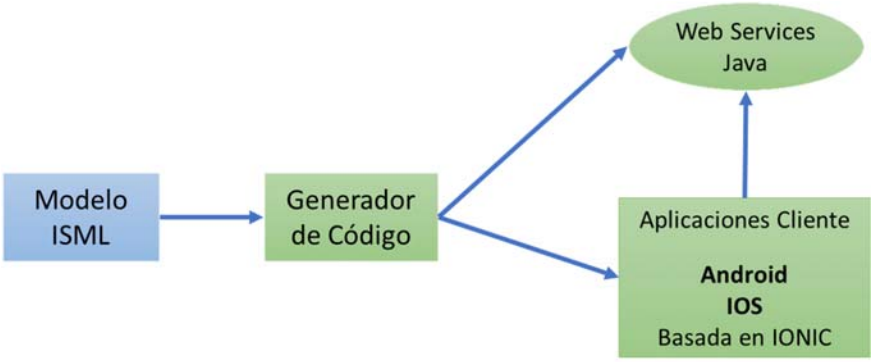


Figura 11. Modelo de transformación general

En la Figura 13 se presenta el modelo de transformación detallado, en donde se especifica cómo cada componente del modelo ISML se transforma en código.

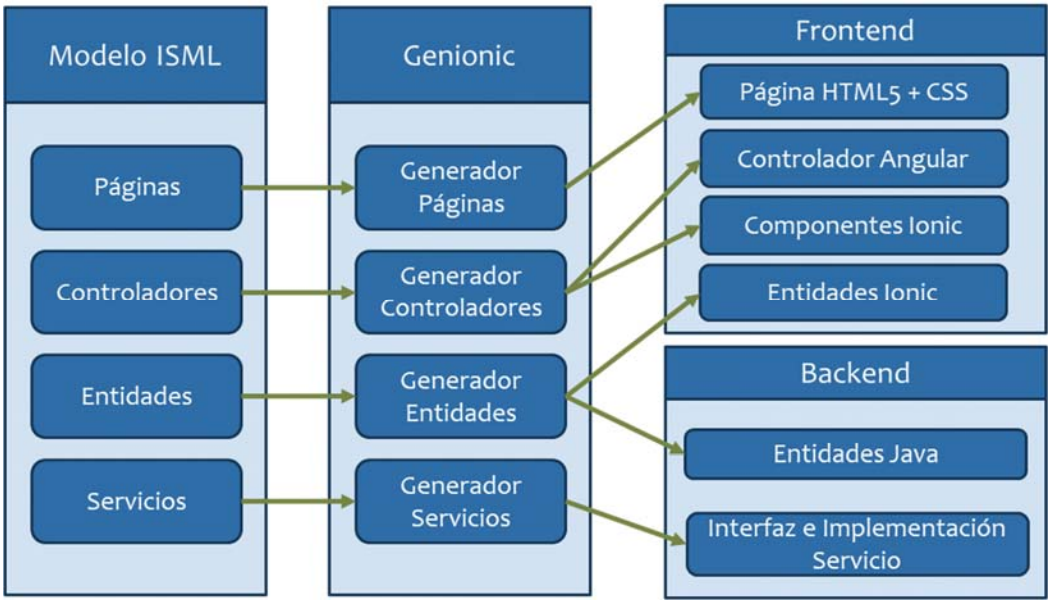


Figura 12. Modelo de transformación detallado

5.3.1. Vista de Procesos del Sistema

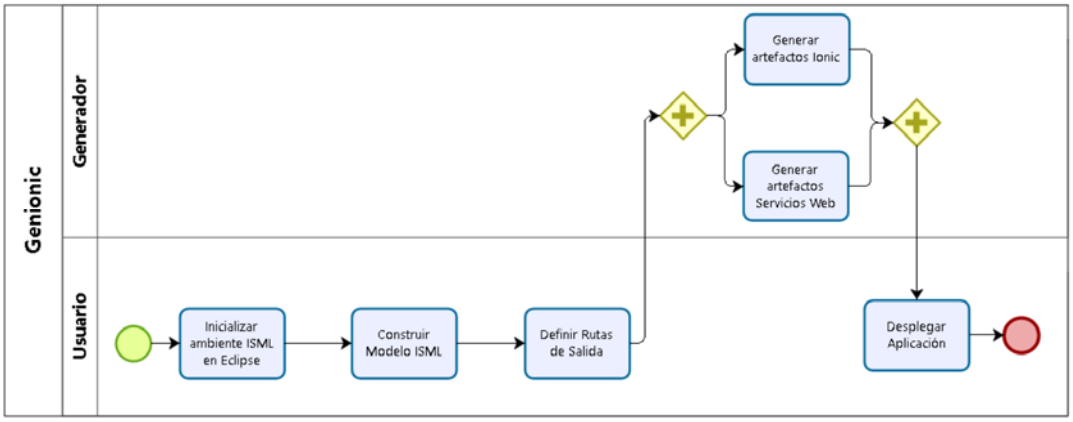


Figura 13. Vista de procesos del sistema

A continuación, se explica cada una de las actividades mencionadas diagrama del proceso anterior:

1. **Inicializar ambientes ISML en Eclipse**: Se inicializa el entorno de Eclipse.
2. **Construir modelo ISML**: El desarrollador deberá expresar la aplicación que quiere generar a través del lenguaje de modelado ISML, utilizando componentes como: Entidades, Controladores, Páginas y Servicios.
3. **Definir rutas de Salida**: Una vez construido el modelo ISML, se deberá configurar el archivo genera generation.conf.json. Este archivo contiene la ubicación en la que se generará los componentes.
4. **Generar artefactos Ionic**: Se generan todos los artefactos de un proyecto Ionic.
5. **Generar artefactos Servicios Web**: Se generan las entidades y clases necesarias para el Servicio Web
6. **Desplegar Aplicación**: Se compila la aplicación en Ionic para los diferentes sistemas operativos móviles requeridos.

5.3.2. Vista Lógica del Sistema

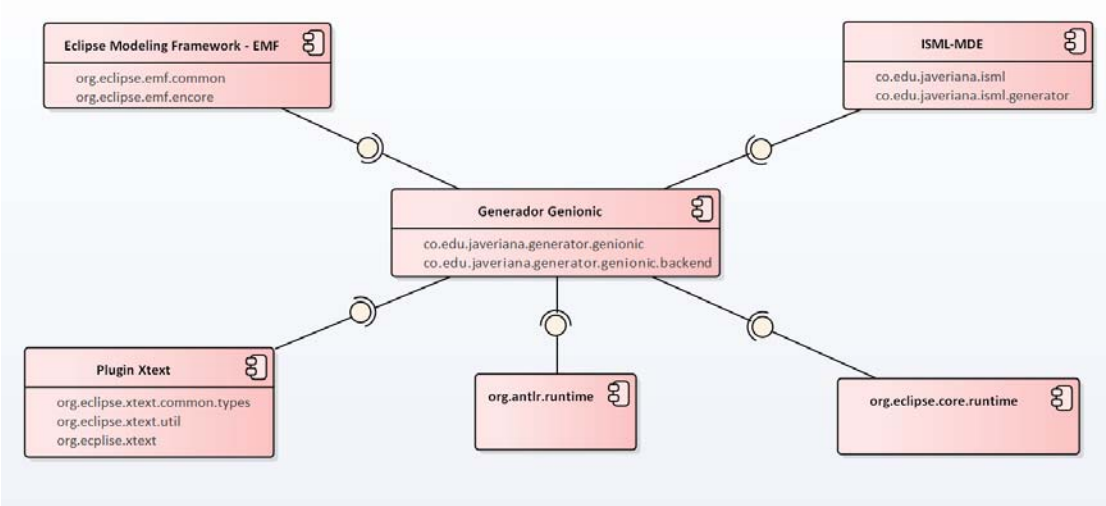


Figura 14. Vista lógica del sistema

El generador Genionic está compuesto por dos elementos y requiere de interfaces con seis elementos externos.

Elementos internos:

- **co.edu.javeriana.genionic.generator.backend:** Elemento que contiene las plantillas y generadores para construir un proyecto de servicios web y persistencia a partir de un modelo expresado en ISML.
- **co.edu.javeriana.genionic.generator:** Elemento que contiene las plantillas y generadores para construir un proyecto Ionic a partir de un modelo expresado en ISML.

Elementos Externos:

- **Eclipse Modeling Framework (EMF):** Framework que permite el trabajo con modelos y meta modelos, en el cual se encuentra construido el lenguaje ISML. Sus paquetes principales son org.eclipse.emf.common y org.eclipse.emf.ecore.
- **Plugin XText:** Framework para desarrollar lenguajes de programación y lenguajes de dominio específico, en donde es soportado el lenguaje ISML. Sus principales paquetes son: org.eclipse.xtext.util, org.eclipse.xtext y org.eclipse.xtext.common.types.
- **ISML-MDE Plugin:** Plugin necesario para hacer uso del lenguaje ISML y sus componentes. Sus principales paquetes son: co.edu.javeriana.isml.generator y co.edu.javeriana.isml.

- **org.eclipse.core.runtime:** Componente que permite ejecutar otra instancia del IDE de desarrollo Eclipse, necesaria para crear el proyecto ISML que se transformará con Genionic.

- **org.antlr.runtime:** Componente que realiza un análisis léxico y sintáctico del proyecto ISML. Este utiliza la herramienta ANTLR.

5.3.3. Vista Física del Sistema

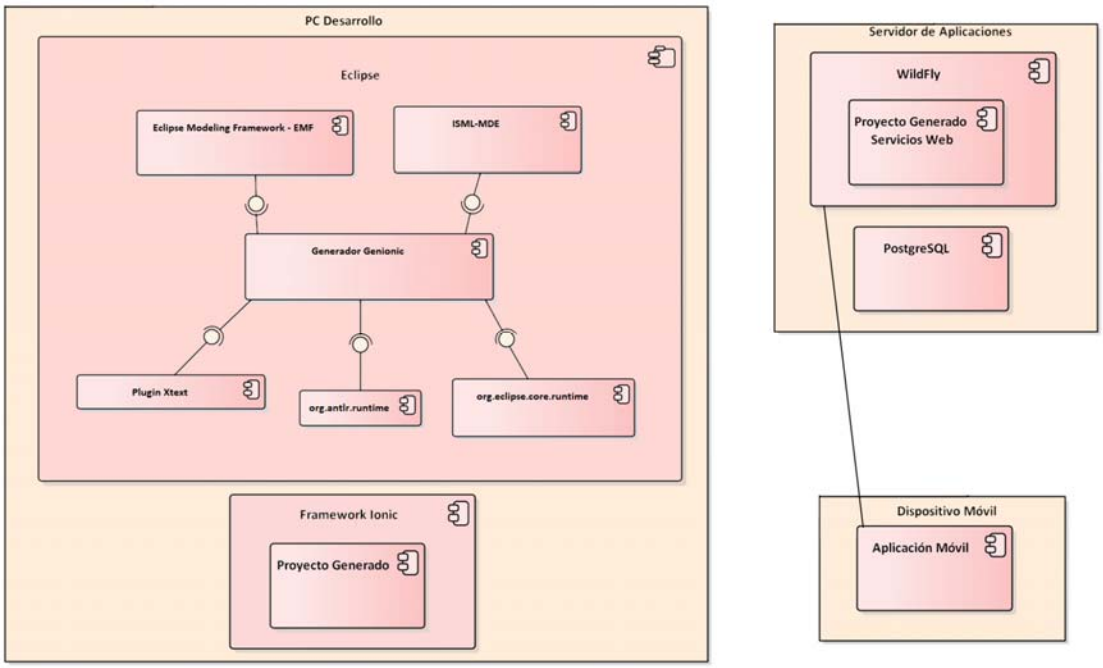


Figura 15. Vista física del sistema

El sistema se encuentra conformado por 3 nodos:

- **Nodo de desarrollo:** Nodo donde se desarrollará el modelo ISML que será transformado con Genionic, para obtener el proyecto Ionic. El framework Ionic permitirá la generación de la aplicación para los diferentes Sistemas Operativos Móviles.

- **Nodo de Dispositivo Móvil:** hace referencia al dispositivo móvil en donde se instalará la aplicación generada.

- **Nodo Servidor de Aplicaciones:** Nodo que contendrá un servidor WildFly y una base de datos PostgreSQL, en donde se desplegarán los servicios web generados.

5.3.4. Estructura detallada del Sistema

El sistema que soporta la funcionalidad del generador Genionic se encuentra conformado por dos grandes paquetes: **co.edu.javeriana.generator.genionic**, que contiene las clases necesarias para generar la interfaz gráfica, el controlador y los servicios en el framework Ionic. El paquete **co.edu.javeriana.generator.genionic**.**backend** contiene las clases necesarias para generar las entidades y los servicios necesarios para la generación de los Servicios Web.

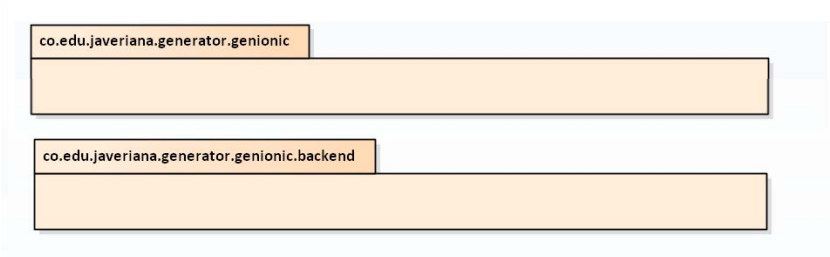


Figura 16. Estructura general del sistema

5.3.4.1. Paquete co.edu.javeriana.generator.genionic

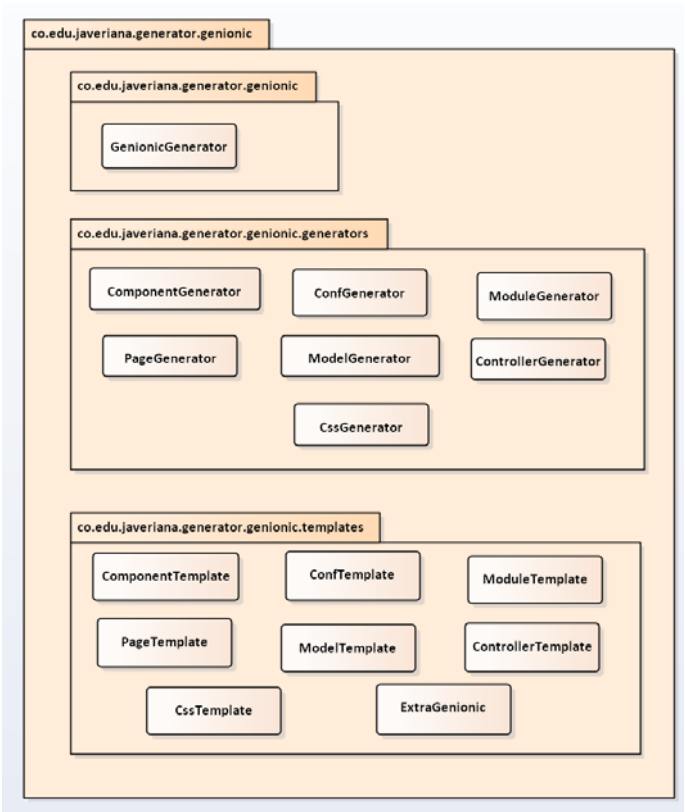


Figura 17. Estructura del paquete co.edu.javeriana.generator.genionic

El paquete **co.edu.javeriana.generator.genionic**, como se muestra en el diagrama anterior, se encuentra conformado por tres paquetes:

- **co.edu.javeriana.generator.genionic**: contiene la clase GenionicGenerator, la cual se encarga de ejecutar cada uno de los generadores propuestos en el paquete co.edu.javeriana.generator.genionic.generators.
- **co.edu.javeriana.generator.genionic.generators**: contiene los generadores que transforman las páginas, controladores y servicios a un proyecto Ionic y sus servicios web correspondientes. A continuación, se explica cada uno de los generadores contenidos:
- **ComponentGenerator**: genera el código del archivo app.component.ts en donde se declaran la página principal y las páginas que hacen parte del menú de la aplicación dentro del proyecto Ionic, es generado a partir de los controladores modelados en ISML.
- **ConfGenerator**: genera el código del archivo app.module.ts en donde se declaran todas las páginas que serán usados en el proyecto Ionic, es generado a partir de los controladores modelados en ISML.
- **ControllerGenerator:** genera el código del controlador de una página de un proyecto Ionic, es generado a partir de cada uno de los controladores modelados en Ionic.
- **CssGenerator**: genera un archivo scss, el cual contiene la hoja de estilos de una página de un proyecto Ionic, y es generado a partir de una página modelada en ISML.
- **ModelGenerator**: genera el código una clase en TypeScript a partir de una entidad modelada en ISML.
- **ModuleGenerator**: genera el código del archivo page.module.ts que hace parte de cada página de un proyecto Ionic, es generado a partir de cada uno de los controladores modelados en Ionic.
- **PageGenerator**: Genera el código de la plantilla html de una página en Ionic, y es generado a partir de una página modelada en ISML.

EL paquete **co.edu.javeriana.lityerses.generator.android.templates**, contiene una plantilla por cada generador definido en el paquete co.edu.javeriana.generator.genionic.generators, los cuales se detallaron anteriormente. Además, contiene las siguientes clases, que son utilizadas únicamente al interior de este mismo paquete:

- **ExtraGenionic**: Contiene elementos comunes que puedes ser reutilizadas en las demás plantillas.

5.3.4.2. Paquete co.edu.javeriana.generator.genionic.backend

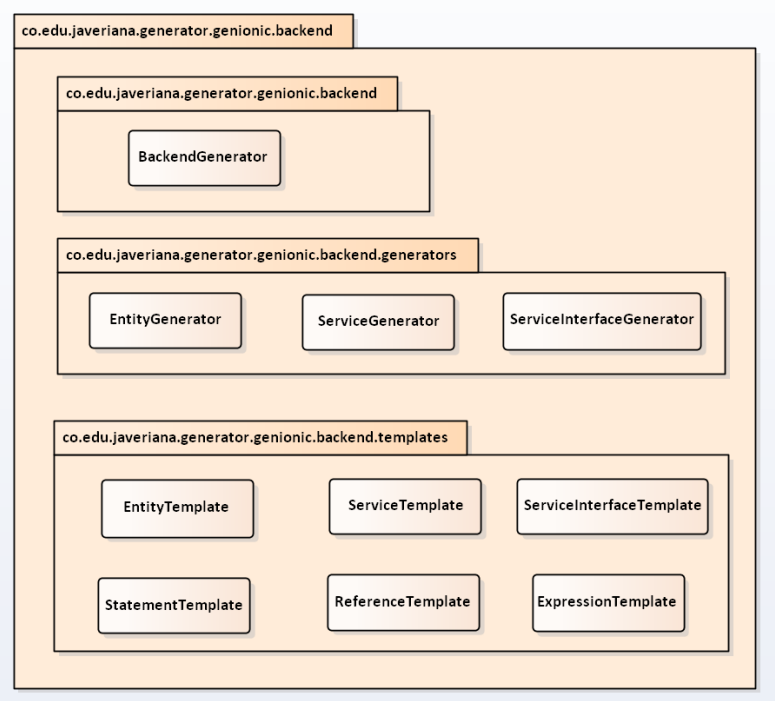


Figura 18. Estructura del paquete co.edu.javeriana.generator.genionic.backend

El paquete **co.edu.javeriana.generator.genionic.backend**, como se muestra en el diagrama anterior, se encuentra conformado por tres paquetes:

- **co.edu.javeriana.generator.genionic.backend**: contiene la clase BackendGenerator, la cual se encarga de ejecutar cada uno de los generadores propuestos en el paquete co.edu.javeriana.generator.genionic.backend.generators.

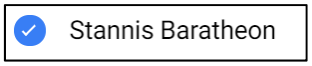
- **co.edu.javeriana.generator.genionic.backend.generators**: contiene los generadores que transforman las entidades y servicios a un proyecto Java de servicios web. A continuación, se explica cada uno de los generadores contenidos:
  - **EntityGenerator**: Genera una entidad modelada en ISML en el proyecto que corresponde a los servicios web de la aplicación.
  - **ServiceInterfaceGenerator**: genera una interface con cada uno de los servicios modelados en ISML.
  - **ServiceGenerator**: genera una clase con la implementación de la interface de cada uno de los servicios modelados en ISML.

- El paquete **co.edu.javeriana.generator.genionic.backend.templates**, contiene una plantilla por cada generador definido en el paquete co.edu.javeriana.generator.genionic.backend.generators, los cuales se detallaron anteriormente. Además, contiene las siguientes clases, que son utilizadas únicamente al interior de este mismo paquete:
  - **ExpressionTemplate**: Contiene elementos comunes que puedes ser reutilizadas en las demás plantillas
  - **ReferenceTemplate**: contiene métodos de navegación a través del lenguaje ISML que pueden ser reutilizados en las demás plantillas
  - **StatementTemplate**: provee métodos utilitarios que son utilizados por las demás plantillas

5.4. Componentes gráficos incluidos en el transformador

Los componentes gráficos que pueden ser generados por Genionic son:

Componente	Descripción	Ejemplo
Button	Botón que recibe como parámetro la cadena de texto que se mostrará.	
Label	Elemento textual que visualiza el texto especificado desde el modelo ISML.	
Image	Componente que permite visualizar una imagen.	
Link	Elemento gráfico que presenta un hipervínculo una url especificada.	
Text	Elemento gráfico que permite visualizar una caja de texto.	
Password	Elemento gráfico que muestra un campo de texto para ingresar contraseñas que no pueden ser visualizadas en el momento de ingresarlas.	

CheckBox	Componente gráfico que presenta una caja de chequeo.	
ListChooser	Elemento gráfico que permite visualizar una lista de opciones.	
ComboChooser	Elemento gráfico que permite visualizar una lista desplegable con las opciones.	
RadioChooser	Elemento gráfico que muestra una lista de opciones tipo radio.	
Calendar	Elemento gráfico que permite visualizar un calendario.	
Map	Elemento gráfico que muestra un mapa con la ubicación actual, obtenida del gps del dispositivo.	

Camera	Elemento que permite tomar una fotografía desde el dispositivo móvil.	
Media	Elemento gráfico que permite visualizar un video contenido en una *url*	
DataTable	Permite visualizar una tabla con los elementos especificados en el modelo *ISML*.	

Tabla 6. Componentes gráficos del transformador

6. PRUEBA DE CONCEPTO

Una vez terminada la construcción del generador Genionic, se realiza una prueba de concepto que valide la capacidad del transformador para generar un proyecto funcional en el framework Ionic.

En las siguientes secciones se describe la aplicación construida para validar el transformador Genionic.

6.1. Descripción de la Aplicación

Para validar el transformador se ha adaptado el modelo ISML Gestión Estudiantes [37] para incluir algunas funcionalidades adicionales de componentes nativos de los dispositivos móviles (Visualización de Mapas con el geoposicionamiento del dispositivo móvil y la captura de imagen de la cámara), esta aplicación está conformada por tres casos de uso tipo CRUD y un caso de negocio para gestionar el proceso de inscripción de asignaturas en una institución de educación superior, permitiendo consultar, crear, editar y eliminar facultades, asignaturas y estudiantes, así como asociar uno o más asignaturas a un estudiante. Adicionalmente en la página de creación de estudiante, se solicitará la captura de la imagen del estudiante y en otra página se mostrará un formulario con la posición del dispositivo en un mapa.

En la siguiente sección se muestra la descripción del Modelo ISML de validación utilizado.

6.2. Descripción del Modelo ISML

El modelo construido en el lenguaje *ISML* que corresponde a la descripción anterior, contiene componentes de tipo entidad, página y controlador, los cuales serán mostrados a continuación:

6.2.1. Componentes de tipo entidad

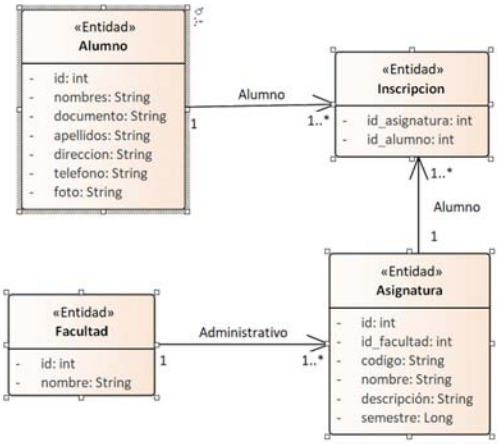


Figura 19. Modelo de Entidades

- Alumno: Entidad que contiene toda la información referente a los alumnos.
- Asignatura: Entidad que contiene toda la información referente a las materias
- Inscripción: Entidad que contiene las materias inscritas para un Alumno.
- Facultad: Entidad que contiene la información referente a una facultad.

Un ejemplo del modelado de la entidad Alumno es el siguiente:

```
package entities;

entity AlumnoE {
        Long   id;
        String documento;
        String nombres;
        String apellidos;
        String direccion;
        String telefono;
        String foto;
}
```

Figura 20. Entidad Alumno en ISML

6.2.2. Componentes de tipo página, controlador y servicio

Las páginas permiten modelar todos los componentes gráficos de la aplicación y se encuentra relacionadas con un controlador, los servicios pueden ser invocados desde los controladores. A continuación, en la Figura 22 se puede visualizar el estereotipo *Página* que permite representar las páginas del modelo, el estereotipo *Controlador* correspondiente a los controladores y el estereotipo *Servicio* que representa los componentes de tipo servicio. Adicionalmente, la se presenta las relaciones *controlledBy* y *has,* la primera indica el controlador asociado a una página, el cual manejará la lógica de este componente. La relación *has* permite identificar qué servicios son instanciados en un controlador.

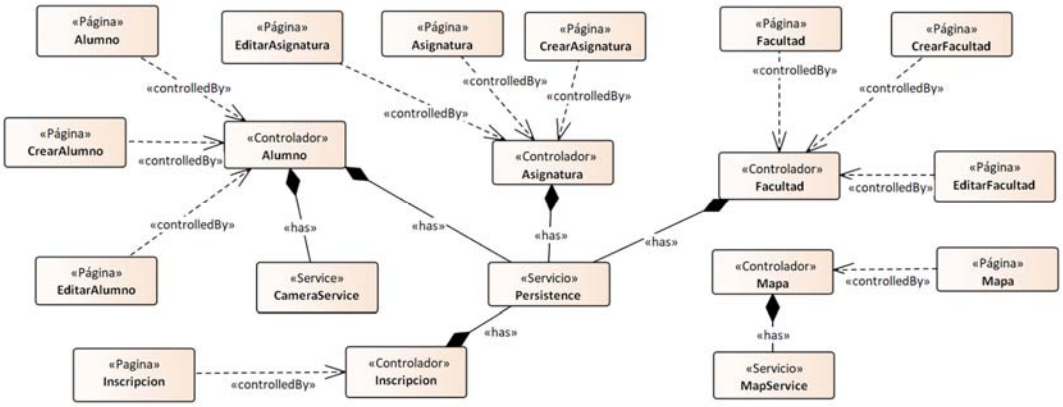


Figura 21. Modelo de páginas, controladores y servicios

Un ejemplo de la página Alumno en ISML, es el siguiente:

```
package asignatura.views;

import asignatura.controllers.*;
import entities.*;


page AlumnoV(Array<AlumnoE> lista) controlledBy AlumnoC  {
        ExtendedMenu("Gestionar Alumno","person");
        Panel("Gestionar Alumno") {

                DataTable("Collection<AlumnoE>", null) {
                        header : {
                                Label("Foto");
                                Label("Documento");
                                Label("Nombres");
                                Label("Apellidos");
                                Label("Dirección");
                                Label("Teléfono");
                                Label("Asignaturas Inscritas");
                    Label("Editar");
                    Label("Eliminar");
                }
                body :
                        for(AlumnoE alumno in lista) {
                        Image("{{alumno.foto}}");
                        Label("{{alumno.documento}}");
                                        Label("{{alumno.nombres}}");
                                        Label("{{alumno.apellidos}}");
                                        Label("{{alumno.direccion}}");
                                        Label("{{alumno.telefono}}");

                        Button("Ver Asignaturas")->asignaturas(alumno) ;
                        Button("Editar")->editar(alumno) ;
                        Button("Eliminar")->eliminar(alumno);
                    }
                }
                Button("Crear Alumno")->crear();

        }

}
```

Figura 22. Pagina Alumno en ISML

Un ejemplo del controlador Alumno en ISML, es el siguiente:

```
package asignatura.controllers;
import entities.*;
import co.edu.javeriana.services.*;
import asignatura.views.*;
import inscripcion.views.*;

controller AlumnoC {

        has Persistence<AlumnoE> persistence;
        has Array<AlumnoE> lista;
        has Any imagen;

        default ionViewDidLoad(){
                lista = persistence.findAll();
        }

        editar(AlumnoE editar){
                show EditarAlumnoV(editar);
        }
        eliminar(AlumnoE eliminar){
                persistence.remove(eliminar);
                show AlumnoV(lista);
        }
        crear(){
                AlumnoE nuevoAlumno =new AlumnoE;
                show CrearAlumnoV(nuevoAlumno);
        }

        asignaturas(AlumnoE cargar){
                show InscripcionV(cargar);
        }
}
```

Figura 23. Controlador Alumno en ISML

6.3. Aplicación Generada

Una vez generado el modelo en el lenguaje ISML, se procede a ejecutar el transformador construido en la ejecución de este proyecto según el manual de usuario. A continuación, se presenta la aplicación generada por el transformador Genionic a partir del modelo ISML:

- **Inicio**

  La siguiente pantalla aparecerá cada vez que el usuario despliegue la aplicación, en ella aparece una referencia al proyecto Genionic haciendo uso de componentes gráficos como: Imagen, botones, y texto. La Figura 25 presenta una impresión de esta pantalla.

Figura 24. Interfaz de Inicio

- **Menú**

  En la siguiente pantalla aparece el menú de la aplicación, desde donde se podrá acceder a las diferentes páginas a las que se les ha definido un acceso desde el menú desde el modelo ISML. En la Figura 26 de puede observar ítems del menú que serán explicados más adelante.

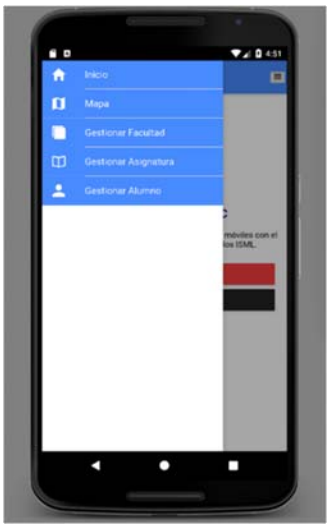


Figura 25. Interfaz del Menú

- **Gestionar Facultad**

En las siguientes pantallas, aparecen las diferentes opciones que se tienen para gestionar facultades en la aplicación, en la primera imagen de izquierda a derecha, se puede observar el listado de facultades que se tienen registradas en la aplicación. En la segunda imagen se observa el formulario de creación de una facultad. La tercera imagen muestra el formulario de edición de una facultad. Las facultades pueden ser eliminadas desde el listado de facultades.

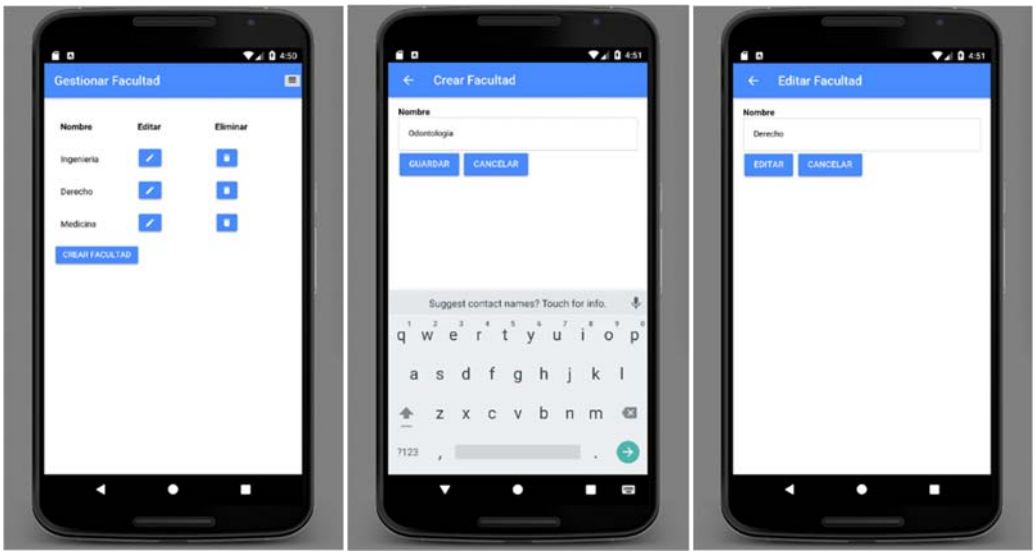


Figura 26. Interfaces de listado, creación y edición de facultades

- **Gestionar Asignatura**

En las siguientes pantallas, aparecen las diferentes opciones que se tienen para gestionar asignaturas en la aplicación, en la primera imagen de izquierda a derecha, se puede observar el listado de asignaturas que se tienen registradas en la aplicación. En la segunda imagen se observa el formulario de creación de una asignatura. La tercera imagen muestra el formulario de edición de una asignatura. Las asignaturas pueden ser eliminadas desde el listado de asignaturas.

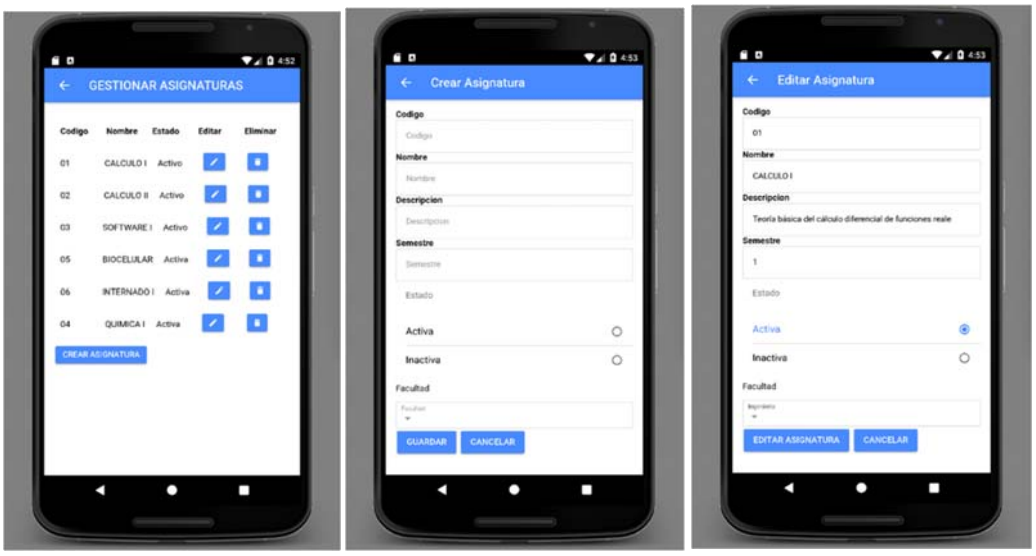


Figura 27. Interfaces de listado, creación y edición de asignaturas.

- **Gestionar Alumno**

En las siguientes pantallas, aparecen las diferentes opciones que se tienen para gestionar alumnos en la aplicación, en esta sección se podrá ver un listado de los alumnos, y desde allí para cada uno de ellos ser podrá editar, eliminar, registrar las materias y retirar materias. Adicionalmente se pueden crear alumnos con la posibilidad de captura su fotografía directamente desde la cámara del dispositivo móvil.

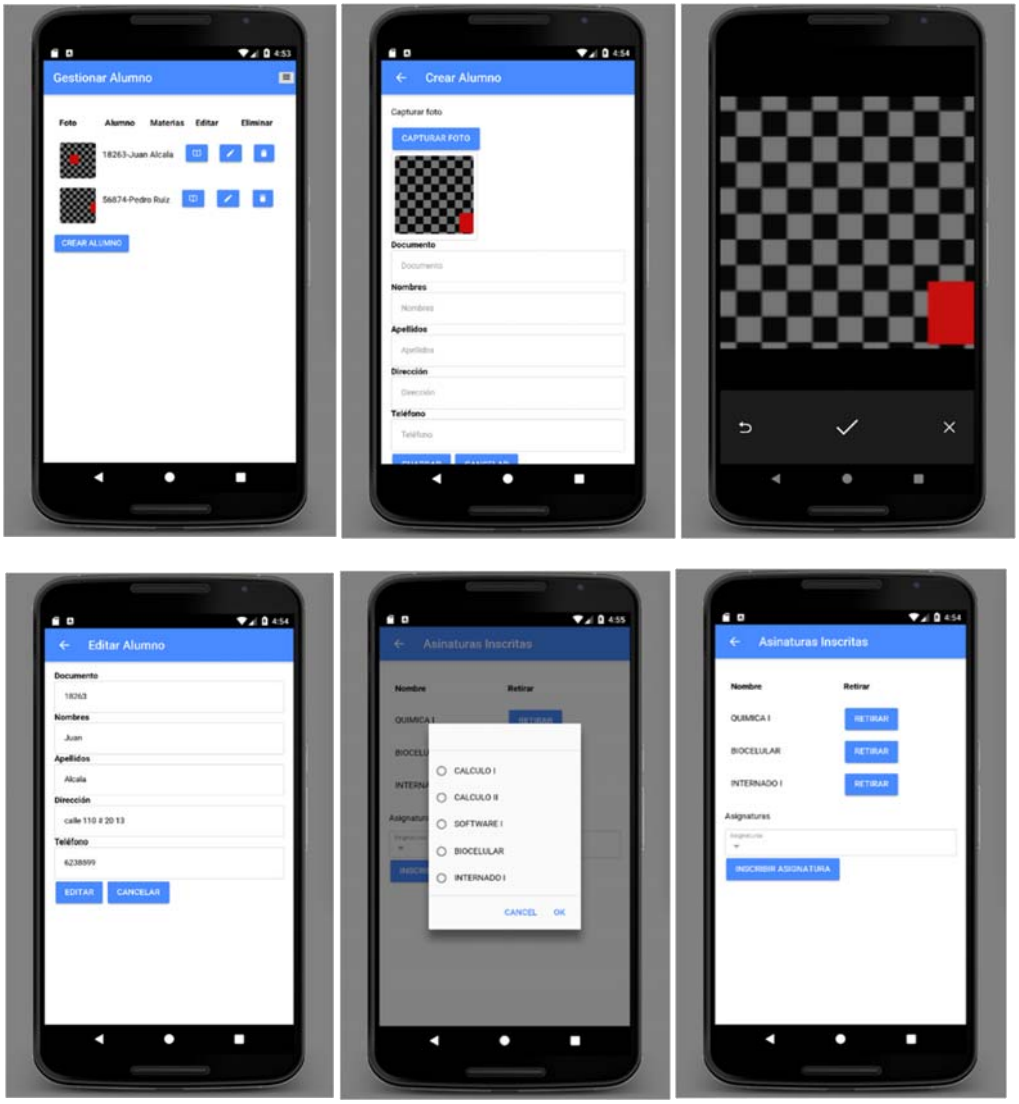


Figura 28. Interfaces de listado, creación, edición y registro de materias de alumnos.

- **Mapa**

En la siguiente pantalla aparece un ejemplo de la utilización de un mapa, donde se muestra la ubicación del dispositivo móvil en punto específico del mapa, sobre este punto se puede observar las coordenadas. El mapa permite interactuar con él, pudiéndose mover el marcador de ubicación y realizar un zoom sobre el mapa cuando se desee.

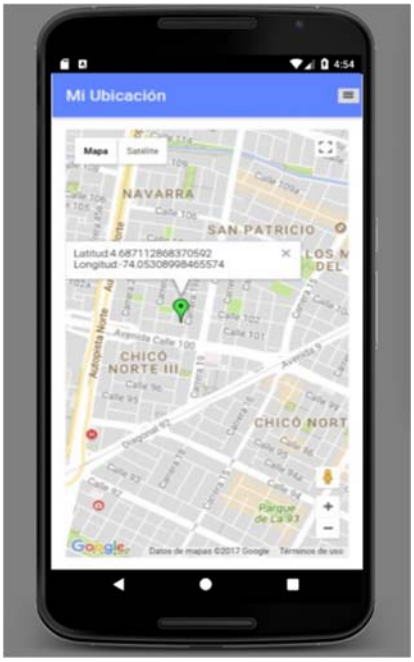


Figura 29. Interfaz geoposicionamiento en mapa.

6.4. Análisis de Resultados

Para llevar a cabo un análisis de los resultados obtenidos con la aplicación detallada en el punto anterior, a continuación, se presentan las mediciones realizadas sobre el modelo del backend y el forntend, que buscan estimar el esfuerzo que se puede reducir con el uso de Genionic, para el desarrollo de aplicaciones móviles, a continuación, se presentan los resultados obtenidos:

Archivos Construidos	Tipo de Componente ISML	Líneas en el Modelo ISML	Líneas Generadas	Reducción de Líneas	
				Cantidad	Proporción
11	Frontend Controlador	292	503	211	1:1.7
4	Frontend Entidad	38	148	110	1:3.9
11	Frontend Página	233	615	382	1:2.6
4	Backend Entidad	38	707	669	1:18.6
6	Backend Services	48	1494	1446	1:31.1
Total de Líneas		**649**	**3467**	**2818**	**1:5.3**

Tabla 7. Comparación de líneas de código generadas

En la Tabla 6, se realizaron mediciones sobre el modelo original expresado en *ISML* con respecto al código generado, tanto en el proyecto Ionic como en el que contiene los servicios web, se estima que para esta prueba de concepto la proporción entre el modelo ISML y las líneas de código generadas es de 1:5.3.

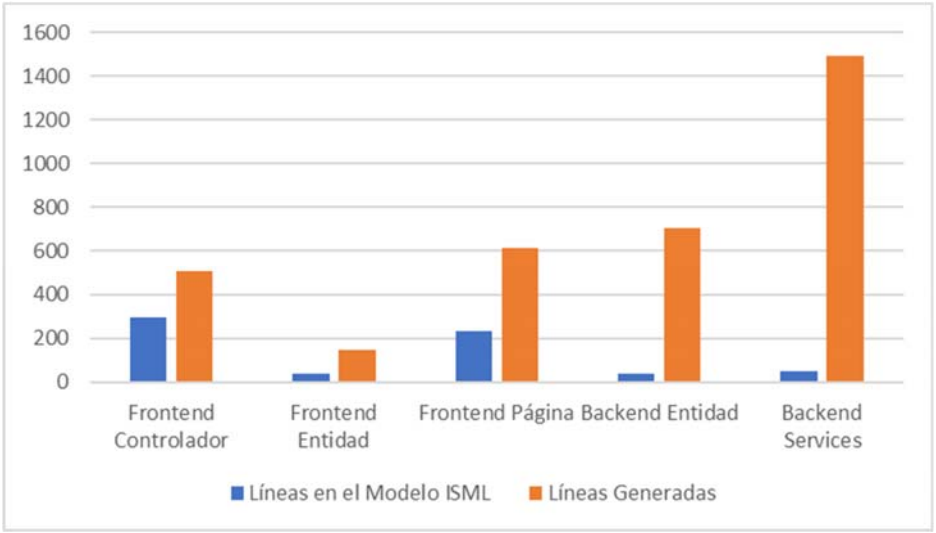


Figura 30. Medición del código generado en la prueba de concepto

La Figura 30, presenta una gráfica que resume las mediciones realizadas en los dos proyectos construidos por el transformador *Genionic* y las compara con el código necesario para su generación.

Los resultados obtenidos a partir de este análisis realizado para el proyecto del backend, se puede observar una gran reducción del número de líneas de código del modelo ISML y el código generado, brindando una alta probabilidad de la reducción de tiempos y costos asociados al desarrollo. La proporción entre el modelo ISML y las líneas de código generadas para el proyecto backend es de **1** a **25.6**.

Por otro lado, los resultados obtenidos a partir de este análisis realizado para el proyecto del frontend, se puede observar una moderada reducción del número de líneas de código entre el modelo ISML y el código generado, lográndose una proporción de **1** a **2.2**. Este resultado como se puede observar, aunque es moderado en comparación al proyecto backend, la reducción de código combinada entre los dos proyectos permite una reducción en proporción de **1** a **5.3**. Esta reducción, aunque no es demasiado alta, pude significar un gran aporte para los tiempos de desarrollo, que sumándose a la capacidad del Framework Ionic de generar la misma aplicación para diferentes plataformas móviles, logran un significativo aporte al tiempo de desarrollo para la producción de una aplicación móvil.

7. CONCLUSIONES

El desarrollo de aplicaciones móviles bajo el enfoque MDE, es prometedor para poder enfrentar la rápida evolución de las tecnologías, en este caso de las plataformas móviles, así como el corto tiempo con el que se cuenta para el desarrollo de las mismas, debido a una creciente necesidad dada por el aumento en las ventas de dispositivos móviles y la industrialización del software en el mercado. En la ejecución de este proyecto, se desarrolló el trasformador Genioic para la generación de aplicaciones móviles multiplataforma con el framework Ionic a partir de un modelo textual expresado en lenguaje ISML.

Según la prueba de concepto realizada, se pudo generar un proyecto para el framework Ionic, desde el cual, a su vez, se generó una aplicación móvil totalmente funcional para la plataforma Android a partir de un modelo ISML, y con un análisis realizado, se pudo identificar que por medio del uso del transformar Genionic existe la posibilidad de lograr una reducción de tiempos y costos de desarrollo.

Dentro del proyecto se logró incluir en el tranformador Genionic la compatibilidad con la cámara, acelerómetro y GPS de los dispositivos móviles, como trabajo futuro se plantea aumentar la capacidad del transformador con componentes de red, contactos, almacenamiento, bluetooth y NFC entre otros.

8. REFERENCIAS

[1] Gartner, — Gartner Says Worldwide Sales of Smartphones Grew 7 Percent in the Fourth Quarter of 2016, http://www.gartner.com/newsroom/id/3609817, Febrary 15, 2017 (Accessed on April 24, 2017).

[2] Fleurey, F., & Solberg, A. (2009, October). A domain specific modeling language supporting specification, simulation and execution of dynamic adaptive systems. In *International Conference on Model Driven Engineering Languages and Systems* (pp. 606-621). Springer Berlin Heidelberg.

[3] Steen, B., Pires, L. F., & Iacob, M. E. (2010, October). Automatic generation of optimal business processes from business rules. In *Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2010 14th IEEE International* (pp. 117-126). IEEE.

[4] Franky, M. C., Pavlich-Mariscal, J. A., Olarte, J. C., Acero, M. C., Zambrano, A., Camargo, J. L., & Pinzón, J. N. (2015, September). ISML: A language and MDE environment to model and generate web applications with integration of existing components. In Computing Colombian Conference (10CCC), 2015 10th (pp. 100-107). IEEE.

[5] Franky, M. C., Franky, M. C., Pavlich-Mariscal, J. A., Pavlich-Mariscal, J. A., Acero, M. C., Acero, M. C., ... & Camargo, J. (2016). ISML-MDE: A practical experience of implementing a model driven environment in a software development organization. International Journal of Web Information Systems, 12(4), 533-556.

[6] S. Kent, "Model Driven Engineering", Lecture Notes in Computer Science, Springer Berlin / Heidelberg, vol. 2335, pp. 286–298, 2002.

[7] Botella, F., Escribano, P., & Peñalver, A. (2016, September). Selecting the best mobile framework for developing web and hybrid mobile apps. In Proceedings of the XVII International Conference on Human Computer Interaction (p. 40). ACM.

[8] Ionic Framework. (2017). Ionic Framework. [online] Available at: https://ionicframework.com/docs/ [Accessed 14 Oct. 2017].

[9] Heinsohn Business Technology. (2017). Available at: http://www.heinsohn.com.co [Accessed 13 Oct. 2017].

[10] Ambler, S. W., & Lines, M. (2012). Disciplined agile delivery: A practitioner's guide to agile software delivery in the enterprise. IBM Press.

[11] S. W. Ambler, «Going Beyond Scrum,» 2013. [En línea]. Available: http://disciplinedagileconsortium.org/Resources/Documents/BeyondScrum.pdf. [Último acceso: 26 Abril 2017].

[12] Palomino Zuluaga, K. C. (2011). Estudio del comportamiento de la industria del software en Colombia ante escenarios de capacidades de innovación y ventajas comparativas por medio de dinámica de sistemas (Doctoral dissertation, Universidad Nacional de Colombia, Sede Medellín).

[13] Gartner, Gartner Says Demand for Enterprise Mobile Apps Will Outstrip Available Development Capacity Five to One, https://www.gartner.com/newsroom/id/3076817, Febrary 15, 2017 (Accessed on Oct 14, 2017).

[14] D. C. Schmidt, Model-driven engineering, Computer-Ieee Computer Society-, vol. 39, pp. 25, 2006.

[15] Kleppe, A. J., Warmer, J., Bast, W. (2003). En The Model Driven Architecture: MDA Explained.

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

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

[18] Pons, C., Giandini, R. S., & Pérez, G. (2010). Desarrollo de software dirigido por modelos.

[19] J. Ludewig, «Modelsin Software Engineering – An Introduction,» SoftwSystModel, p. 5–14, 2003.

[20] Lachgar, M., & Abdali, A. (2017). Decision Framework for Mobile Development Methods. INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS, 8(2), 110-118.

[21] Gartner Says by 2016, More Than 50 Percent of Mobile Apps Deployed Will be Hybrid. (2017). Available at: https://www.gartner.com/newsroom/id/2324917 [Accessed 24 Oct. 2017].

[22] Cordova. (2017). Available at: https://cordova.apache.org/docs/en/latest/guide/overview/index.html [Accessed 24 Oct. 2017].

[23] A. Gupta, Java EE 7 Essentials. Cambridge, Massachusetts O'Reilly c2013, 2013.

[24] Java Specification Request (JSR) 342, Available at: https://www.jcp.org/en/jsr/detail?id=342. Last accessed on 2017-02-19.

[25] A. Goncalves, Beginning Java EE 7. Berkeley, California Apress c2013, 2013.

[26] D. R. Heffelfinger, Java EE 7 with GlassFish 4 Application Server. Birmingham, UK: Packt Publishing, 2014.

[27] P. A. Pilgrim, Java EE 7 Developer Handbook: Develop Professional Applications in Java EE 7 with this Essential Reference Guide. Birmingham: Packt Publishing, 2013, pp. 129-132.

[28] Where does the Ionic Framework fit in?. (2017), http://blog.ionic.io/where-does-the-ionic-framework-fit-in/?_ga=2.228777630.32752571.1508862574-1706578001.1507616091. [Accessed 24 Oct. 2017].

[29] Architecture Overview, https://angular.io/guide/architecture. [Accessed 25 Oct. 2017].

[30] Features & Benefits., https://angular.io/features. [Accessed 25 Oct. 2017].

[31] Ed-Douibi, H., Izquierdo, J. L. C., Gómez, A., Tisi, M., & Cabot, J. (2016, April). EMF-REST: generation of restful apis from models. In Proceedings of the 31st Annual ACM Symposium on Applied Computing (pp. 1446-1453). ACM.

[32] J. Bocanegra, J. A. Pavlich-Mariscal and A. C. Ramos, "MiDAS: A model-driven approach for adaptive software." in Webist, 2015, pp. 281-286.

[33] J. A. Andrade Mayorga, Lityerses: Transformador para generar aplicaciones móviles con componentes adaptativos a partir de un modelo independiente de plataforma, Bogotá: Pontificia Universidad Javeriana, 2016.

[34] F. S. Franco Hernández, Anchurus-GEN: Generador de código PHP a partir de modelos ISML, Bogotá: Pontificia Universidad Javeriana, 2015.

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

[36] Daniel Ramírez Echeverri, Gengular: Hacia la automatización de aplicaciones empresariales bajo el paradigma de arquitectura SPA y el enfoque MDE, Bogotá: Pontificia Universidad Javeriana, 2017.

[37] LACHGAR, M., & ABDALI, A. (2015). MODELING AND GENERATING THE USER INTERFACE OF MOBILE DEVICES AND WEB DEVELOPMENT WITH DSL. Journal of Theoretical & Applied Information Technology, 72(1).

[38] Geiger-Prat, S., Marín, B., España, S., & Giachetti, G. (2015, May). A GUI modeling language for mobile applications. In Research Challenges in Information Science (RCIS), 2015 IEEE 9th International Conference on (pp. 76-87). IEEE.

[39] Lachgar, M., & Abdali, A. (2014, October). Generating Android graphical user interfaces using an MDA approach. In Information Science and Technology (CIST), 2014 Third IEEE International Colloquium in (pp. 80-85). IEEE.

[40] Vaupel, S., Taentzer, G., Gerlach, R., & Guckert, M. (2016). Model-driven development of mobile applications for Android and iOS supporting role-based app variability. Software & Systems Modeling, 1-29.

[41] Duarte, P. A., Barreto, F. M., Gomes, F. G. A., Carvalho, W. V., & Trinta, F. A. (2014, November). A Model-Driven Approach to Generate Context-Aware Applications. In Proceedings of the 20th Brazilian Symposium on Multimedia and the Web (pp. 99-102). ACM.

[42] Parada, A. G., & de Brisolara, L. B. (2012, November). A model driven approach for android applications development. In Computing System Engineering (SBESC), 2012 Brazilian Symposium on (pp. 192-197). IEEE.