FRAMEWORK PARA EL DESARROLLO DE APLICACIONES ORIENTADAS A VISUALIZACIÓN DE DATOS

HENRY ANDRES JIMÉNEZ HERRERA
MARÍA FERNANDA GUERRERO GARCÍA

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2017

FRAMEWORK PARA EL DESARROLLO DE APLICACIONES ORIENTADAS A VISUALIZACIÓN DE DATOS

HENRY ANDRES JIMÉNEZ HERRERA
MARÍA FERNANDA GUERRERO GARCÍA

Trabajo de Grado para optar por el título de Ingeniero de Sistemas

Director
PhD GABRIEL RODRIGO PEDRAZA FERREIRA
Doctor en Informática

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2017

Dedicatorias

Dedicado a Dios que me dio la fortaleza para cumplir este objetivo, a mis padres que siempre me han brindado su apoyo, a todas y cada una de las personas que me dieron la mano durante los momentos difíciles de mi carrera universitaria.

Henry Andrés Jiménez Herrera

Dedicado principalmente a Dios, por haberme permitido llegar hasta este momento tan importante de mi formación profesional, por su infinita bondad y amor, a mis padres por ser los pilares más importantes de mis logros, por su apoyo incondicional y su cariño, a mi hermana por brindarme su apoyo cuando lo necesite, a mi sobrino por darme las fuerzas para cumplir mis objetivos, a todas y cada una de las personas que creyeron en mí y me apoyaron en momentos difíciles, haciendo de este sueño una realidad.

María Fernanda Guerrero García

Agradecimientos

A Dios por guiar la realización de este proyecto, al profesor Gabriel Pedraza por confiar en nosotros y en nuestra capacidad para realizar esta meta y guiarnos en el cumplimiento de la misma, a todos los profesores que nos permitieron adquirir los conocimientos y habilidades propias de un ingeniero a lo largo de nuestra carrera.

A nuestras familias que siempre nos brindaron su apoyo incondicional y por creer en nosotros, a todos nuestros compañeros y amigos de la universidad que nos apoyaron en el transcurso de nuestra carrera.

CONTENIDO

Pág.

Lista de Ilustraciones

Pág.

Lista de Tablas

Pág.

Lista de Anexos

Nota: Los anexos se encuentran en carpeta adjunta en CD.

Lista de Símbolos y Abreviaturas

Abreviatura	Descripción Extendida
ANTLR	Another Tool for Language Recognition
AST	Abstract Syntax Tree
CASE	Computer Aided Software Engineering
CSS	Cascading Style Sheets
DSL	Domain Specific Language
EMF	Eclipse Modeling Framework
HTML	Hyper Text Markup Language
IDE	Integrated Development Environment
IoT	Internet of things
JS	JavaScript
JSON	JavaScript Object Notation
REST	Representational State Transfer
UML	Unified Modeling Language

GLOSARIO

AST: un árbol de sintaxis abstracta es una representación de árbol de la estructura sintáctica abstracta (simplificada) del código fuente, escrito en cierto lenguaje de programación

DSL: Lenguaje de dominio específico (en inglés Domain Specific Language), es un lenguaje de programación o especificación dedicado a resolver un problema en particular.

Eclipse (IDE): Plataforma de desarrollo de software compuesta por un conjunto de herramientas de programación en su mayoría de código abierto.

EMF: Eclipse Modeling Framework, es un marco de modelado y facilidad para la generación de código.

Framework: es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

Xtext: es un entorno de código abierto para el desarrollo de lenguajes de programación y lenguajes de domino especifico(DSL).

Xtend: es un lenguaje de programación de alto nivel de uso general para la Máquina Virtual Java.

RESUMEN

TÍTULO: FRAMEWORK PARA EL DESARROLLO DE APLICACIONES ORIENTADAS A VISUALIZACIÓN DE DATOS*

AUTORES: HENRY ANDRES JIMÉNEZ HERRERA
MARÍA FERNANDA GUERRERO GARCÍA**

PALABRAS CLAVE: Framework, Visualización, Datos, DSL, Gráficos, Xtext, Xtend.

DESCRIPCIÓN:

Nos encontramos en la era de la información que se caracteriza por la capacidad que tenemos actualmente los seres humanos de transferir o consultar información libremente haciendo uso de Internet, adicionalmente cada vez son más los dispositivos conectados a la red, esto implica que cada segundo se generen grandes cantidades de datos que sin un tratamiento y una visualización adecuados, no aportan información relevante para las diferentes organizaciones implicadas, de aquí surge la necesidad de desarrollar metodologías, sistemas y plataformas en general que permitan interpretar estos datos para descubrir la valiosa información que contienen.

Es así como la investigación se centró en crear un prototipo de un entorno de trabajo que permite agilizar el desarrollo de aplicaciones orientadas a la visualización de datos y mejorar la interacción entre los roles implicados en esta tarea, como expertos en visualización de datos e ingenieros de software, diseñando e implementando un lenguaje de dominio específico que permite describir las diversas visualizaciones con un nivel superior de abstracción en comparación con los lenguajes de propósito general, proporcionando una sintaxis más específica y menos extensa. También se programó una herramienta que interpreta el lenguaje y genera código para una plataforma determinada, el prototipo construido genera código base para aplicaciones web y aplicaciones basadas en Python, adicionalmente, se realizaron pruebas de validación y pruebas de usuario, permitiendo identificar las ventajas, desventajas y la posible evolución de la herramienta.

---

*Proyecto de grado.

****** Facultad de Ingeniería Físico-Mecánicas, Escuela de Ingeniería de Sistemas e Informática, Director: PhD Gabriel Rodrigo Pedraza Ferreira

ABSTRACT

TITLE: FRAMEWORK FOR DEVELOPMENT OF APPLICATIONS ORIENTED OF DATA VISUALIZATION*

AUTHORS: HENRY ANDRES JIMÉNEZ HERRERA
MARÍA FERNANDA GUERRERO GARCÍA**

KEYWORDS: Framework, Visualization, Data, DSL, Charts, Xtext, Xtend.

DESCRIPTION:

We are in the information age that is characterized of currently capacity we have the humans of transferring or consult information freely using the Internet, additionally every time are more dispositive connected to network, this implies that every second are generated big data amounts that without appropriate treatment and visualization do not provide relevant information for the different involved organizations, from this originate the need of develop methodologies, system and platforms in general that allow to interpret this data to discover the valuable information contained in them.

That is how the investigation is center in make a prototype of framework which speed up the development of applications oriented to the visualization of data and improve the interaction between the roles involved in this task, such as data visualization experts and software engineers, designing and implementing a domain specific language that allows to describe the various visualizations with a higher level of abstraction, compared with general purpose modeling languages, providing a more specific and less extensive syntax. Also programmed was a tool that interprets the language and generates code for a specific platform, the built prototype generates base code for applications web and applications based on Python, additionally, validation tests and user tests were performed, allowing to identify advantages, disadvantages and the possible evolution of the tool.

*Project of grade

** Faculty of Physical-Mechanical Engineering. School of Engineering and Computer Science. Director: Dr. Gabriel Rodrigo Pedraza Ferreira.

INTRODUCCIÓN

Vivimos en la era de la información, nunca los seres humanos se habían comunicado de una forma tan eficiente como lo hacemos actualmente, esto se debe al desarrollo de redes de comunicación cada vez mejores, que evolucionaron en lo que hoy conocemos como el Internet, la red mundial que permite acceder a una cantidad ilimitada de información y también compartir nuestro conocimiento con el mundo.

Las personas por si mismas no pueden conectarse a la red, necesitan hacer uso de dispositivos electrónicos inteligentes diseñados para tal fin, como computadoras, celulares, tabletas y demás, aunque no son los únicos dispositivos que se puede conectar a la red, existe un fenómeno conocido como internet de las cosas IoT (por su sigla en inglés), un concepto que se refiere a la interconexión digital de objetos cotidianos con Internet, bajo este panorama es fácil deducir que cada vez más dispositivos y personas se conectarán a la red generando un gran volumen de datos que aumenta con el paso del tiempo.

Los datos contienen información, un recurso que en la actualidad es muy valioso para las personas y las empresas en general, ya que con determinada información se puede predecir el comportamiento de un mercado, las condiciones climáticas de un lugar, las tendencias de consumo de algún país, entre otros. Para obtener dicha información, es necesario estudiar los datos, realizar un tratamiento de los mismos y visualizarlos de la mejor forma.

Es por esto que muchas organizaciones o personas en la actualidad, requieren de soluciones tecnológicas que les permitan estudiar los datos que son de su interés

para tomar las mejores decisiones, por tanto, la demanda de software orientado a la visualización de datos es cada vez mayor o al menos la gran mayoría de desarrollos actuales requieren de una componente orientada al estudio de los datos de sus usuarios, la idea de crear una plataforma o marco de trabajo que permita agilizar el desarrollo de estas aplicaciones surge en respuesta a esta necesidad, de manera que se puedan crear aplicaciones de una forma más eficiente y con mayor calidad.

El objetivo de este proyecto es crear un prototipo basado en una metodología de desarrollo por faces, que permite identificar las características que debe poseer el framework o entorno de trabajo, para que sus usuarios puedan describir de manera eficiente la forma de visualizar los datos relacionados con la aplicación que están desarrollando, y que una vez descritas dichas visualizaciones, genere un código base que los usuarios podrán adaptar para obtener los resultados finales deseados en su aplicación.

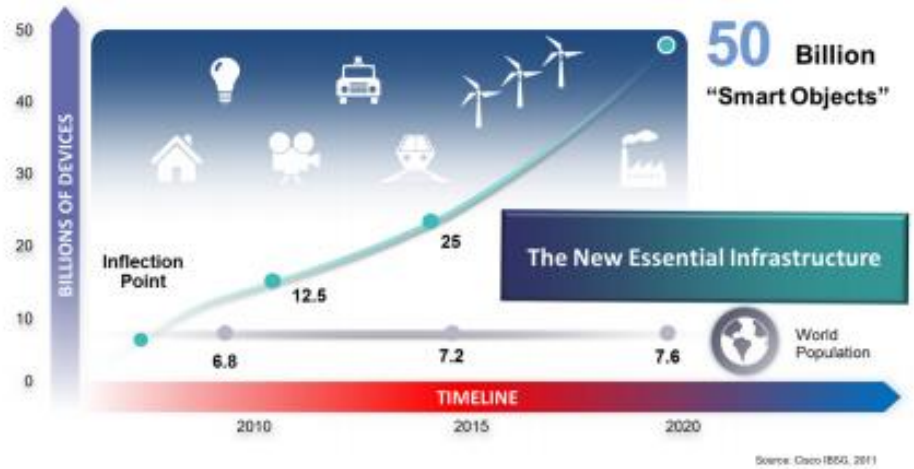
Con un amplio proceso de investigación, diseño, codificación, pruebas y retroalimentación fue posible desarrollar el proyecto descrito en las siguientes páginas.

1. PLANTEAMIENTO Y JUSTIFICACIÓN DEL PROBLEMA

La tecnología actual genera grandes cantidades de datos asociados a fenómenos como, por ejemplo, cambios en las condiciones climatológicas, fluctuación del valor de las acciones en el mercado, el flujo vehicular de una ciudad, entre otros, dado que las redes de transmisión global permiten el envío de datos de manera rápida y eficiente.

Esto ha generado el surgimiento de teorías como el internet de las cosas IoT (por su sigla en inglés), donde se habla de la creciente cantidad de dispositivos conectados a la red (50 Billones de dispositivos inteligentes para el año 2020), con la generación de volúmenes de datos cada vez mayor, la necesidad de analizarlos para extraer información de ellos también crece cada día, debido a la gran magnitud de los datos, las aplicaciones informáticas tradicionales de tratamiento de datos no son suficiente, así surge el análisis de datos a gran escala conocido como Big Data, que permite analizar estos conjuntos de datos compuestos por millones de registros y extraer información que puede ser muy valiosa para la comprensión de los fenómenos asociados a los datos.

Figura 1. Dispositivos Conectados a Internet y su Futura Evolución



Fuente: CISCO ,2011

Pero incluso los resultados de los análisis del Big Data generan más datos y cuando poseemos un conjunto de datos, se hace necesario establecer una forma óptima de visualizarlos para poder realizar una interpretación, contrastación y comparación de los mismos, es allí donde surge la visualización de datos, cuyo objetivo es el de apoyar el proceso de análisis e interpretación de la información, debido a esto, cada día se incorporan nuevas formas de visualización y el desarrollo de proyectos enfocados a la visualización de datos es cada vez mayor. Por otra parte, en los proyectos de desarrollo de software que posea una componente de visualización de datos, se puede identificar tres roles asociados: El experto en visualizaciones de datos, el ingeniero de software y el experto en análisis y tratamiento de datos.

Este proyecto propone el diseño de un framework que permite configurar la forma en que se muestra la información contenida en los datos, sin tener conocimiento profundo del manejo de librerías, desarrollo para determinadas plataformas y demás especificaciones técnicas con las cuales tuviera que lidiar el experto en visualizaciones. Agilizando el desarrollo para una plataforma en específico y permitiendo que se configure la o las conexiones con los diferentes servidores en los cuales esté almacenada la información que se desea visualizar, de esta manera facilitando la labor del ingeniero de software.

El proyecto se enfoca en la visualización de datos, omitiendo así el tratamiento de los mismos y no enfocando la herramienta al rol del experto en tratamiento de los datos, asumiendo que los datos con los cuales vamos a trabajar han sido procesados previamente y se encuentran en un formato determinado.

Los expertos en visualizaciones generalmente no son expertos en el desarrollo de software, por lo tanto, trabajan de la mano con el ingeniero de software, esta integración genera dos problemas fundamentales: (1) el experto en visualización debe transmitir de forma comprensible al desarrollador la manera en la que desea mostrar los datos, asimismo, el experto debe considerar las limitantes de la

tecnología utilizada, como lo puede ser una librería, una plataforma, entre otros. (2) el desarrollador tiene que contextualizarse para comprender e interpretar lo que expresa el experto en visualizaciones y de esta forma desarrollar una aplicación a partir de lo que se desea. Lograr esta labor requiere de tiempo y esfuerzo por parte de los roles implicados, debido principalmente a la cantidad de plataformas y opciones disponibles en el mercado.

Existen una gran cantidad de herramientas desarrolladas para la visualización de los datos, cada herramienta posee su propia metodología de trabajo y está ligada generalmente a una plataforma en específico, cuando se elige una librería es necesario aprender su funcionamiento y la forma en la cual están definidas las visualizaciones allí, esta propuesta abstrae los aspectos fundamentales al momento de definir una visualización, concentrándose en la forma en la cual se visualizan los datos y el origen de los mismos, para que de esta forma el experto en visualizaciones pueda enfocarse en su labor mediante una herramienta dinámica que le permite trabajar en su especialidad, sin la necesidad de preocuparse o comprender aspectos técnicos, la herramienta también agiliza el desarrollo que lleva a cabo el ingeniero de software el cual podrá generar el código base de las visualizaciones definidas en la herramienta para una determinada plataforma como un proyecto, el cual podrá modificar para incluir las demás características que el desarrollo requiera. La herramienta puede mejorar significativamente la productividad del desarrollador, la fiabilidad de la generación automática de código en comparación con la codificación manual también reducirá el número de defectos en los programas resultantes, mejorando así la calidad.

2. OBJETIVOS

2.1 OBJETIVO GENERAL

Diseñar un Framework que disminuya la complejidad y el tiempo de desarrollo de aplicaciones orientadas a la visualización de datos.

2.2 OBJETIVOS ESPECÍFICOS

- Identificar las características principales del framework de visualización de datos.
- Plantear un lenguaje que permita describir las visualizaciones definidas por el usuario.
- Diseñar e implementar un prototipo del framework que genere proyectos para al menos una plataforma.
- Construir y realizar un conjunto de pruebas que validen el funcionamiento del framework.
- Diseñar y realizar un conjunto de pruebas de usuario para validar la disminución de complejidad en la tarea propuesta y la optimización en el tiempo de desarrollo de las aplicaciones.

3. MARCO DE REFERENCIA

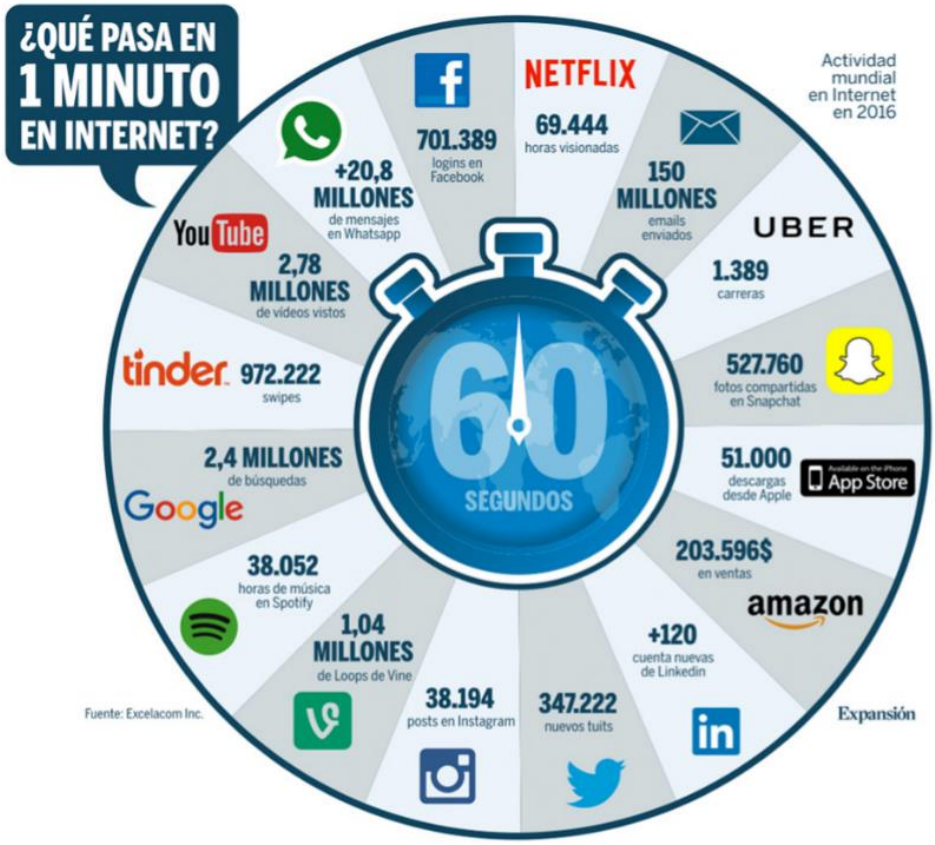
Esta sección presenta los conceptos de importancia que se requieren para comprender el desarrollo del proyecto, desde la teoría hasta el rol de ciertas tecnologías utilizadas.

3.1 VISUALIZACIÓN DE DATOS

Los datos nos permiten obtener información sumamente valiosa para comprender nuestro entorno, el mercado en el cual estamos trabajando, nuestros clientes, el clima, etc., sin embargo, descubrir estos patrones, esta información oculta en los datos no es una tarea sencilla, y es allí donde surge la visualización de los datos como un proceso de apoyo para comprender en profundidad y detalle los mismos, de forma que se transformen en información comprensible para el usuario. El término visualización de datos surge a partir del nacimiento de la web 2.0, debido a que la cantidad de datos generados son de grandes magnitudes, lo cual hace compleja la búsqueda e interpretación de los mismos, esto ha generado una necesidad de crear herramientas que permitan comprender y asimilar los datos de una mejor forma, por esta razón, la visualización de datos se ha convertido en una poderosa herramienta que transforma estas relaciones numéricas presentes en los datos, en impactos visuales, en los cuales se pueda apreciar claramente las características o tendencias que queremos mostrar.

La figura 2 permite apreciar la cantidad de datos que se generan por minuto en internet y dado que estamos en la era de la información, es importante resaltar la importancia del correcto análisis de los datos generados ya que supone ventajas muy importantes para el desarrollo y la competitividad de las diferentes organizaciones a nivel mundial.

Figura 2. Internet en un Minuto



Fuente. Diario la Expansión España [1]

3.2 ARQUITECTURA DE SOFTWARE

La arquitectura de software de un sistema informático se define como "*el conjunto de estructuras necesarias para razonar sobre el sistema, que comprende los elementos de software, sus propiedades, y las relaciones entre ellos*" [2]

Del mismo modo, los sistemas software se componen de varias estructuras, las cuales conforman un conjunto de elementos que se encuentran unidos por una relación. Una estructura por sí sola no constituye toda la arquitectura.

Por otra parte, no todas las estructuras son arquitectónicas, sólo se consideran aquellas que se basan en el razonamiento sobre el sistema y las propiedades del mismo, es decir, que se tiene en cuenta aquellas estructuras que están relacionadas con un atributo del sistema que es importante para alguna de las partes interesadas. En resumen, una arquitectura de software es una abstracción de un sistema que selecciona ciertas estructuras y omite otras que no son favorables para comprender su funcionamiento.

3.3 FRAMEWORK

Un Framework es una estructura conceptual y tecnológica de soporte definido, normalmente contiene unidades de software que agilizan y sirven de manera eficiente al desarrollo de aplicaciones. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto. El framework tiene como objetivo principal ofrecer una funcionalidad definida, contenida en sí misma, siendo construida utilizando patrones de diseño y su característica principal es su alta cohesión y bajo acoplamiento, es decir, que tenemos un conjunto de unidades de software muy bien acopladas, pero que la dependencia entre estos módulos es baja.

Muchas veces los framework en especial los comerciales, son cajas negras para los desarrolladores de software, de manera que cumplen sus funciones, pero no se sabe de qué forma lo hacen, esto conlleva a que muchas veces existan errores de software cuya causa no es fácil de identificar, sin embargo, también existen muchas opciones disponibles cuyo funcionamiento y metodología es totalmente pública, o que tienen una documentación muy buena, permitiendo aprovechar al máximo estas herramientas. Los framework no necesariamente están ligados a un lenguaje de programación, aunque así sea en muchas ocasiones, es posible que el framework

defina una estructura para una aplicación completa o solo para una parte de ella, esto depende del objetivo con el cual se desarrolle dicho framework.

Algunas de las ventajas cuando utilizamos un framework son:

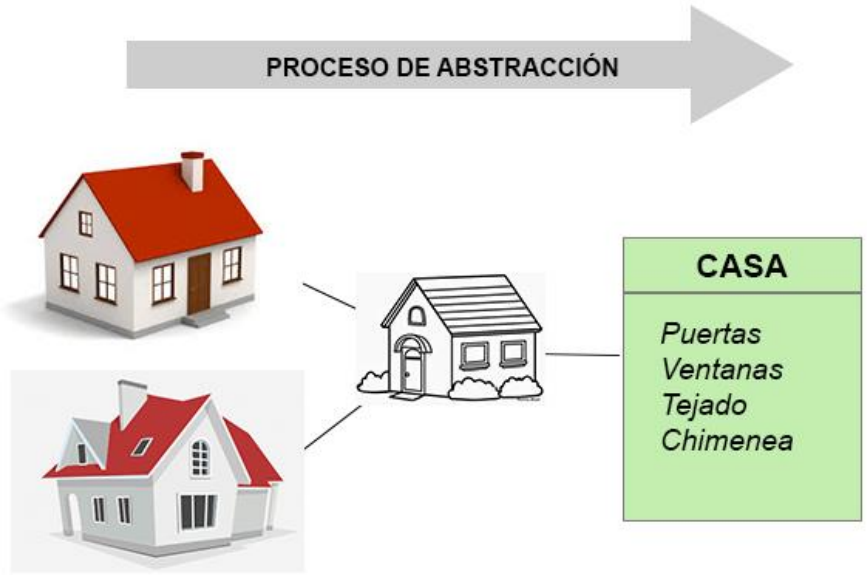
- El desarrollador no necesita plantearse una estructura general de la aplicación ya que es el framework quien le proporciona un esqueleto con dicha estructura en la cual se trabaja posteriormente para lograr la aplicación deseada.
- Facilita la colaboración entre los desarrolladores implicados, ya que, siempre existen problemas a la hora de interpretar el diseño o el código realizado por otro desarrollador, es por esto que seguir un estándar y una estructura definida por el framework ayuda a mejorar esta colaboración entre las partes implicadas.

3.4 ABSTRACCIÓN Y MODELOS

La abstracción es una característica preponderante en el desarrollo de software que proporciona la posibilidad de aislar los elementos de un contexto determinado para enfocar el propósito de donde se extrae, restando importancia en este punto a la forma de cómo hacerlo e invertir los esfuerzos en el qué hacer [3]. Este concepto vincula el proceso de implementación con la generación de modelos que se enfocan en una capa de abstracción superior y que generaliza una solución aceptable dentro del problema específico planteado.

En la Figura 3 se representa de forma gráfica el modelado, elevando el nivel de abstracción de una representación puntual tomando características conceptuales.

Figura 3. Proceso de Abstracción



3.5 HERRAMIENTAS CASE

Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadora) son diversas aplicaciones o programas informáticos destinados a aumentar la productividad en el desarrollo de software, estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software dando soporte a procesos como el diseño del proyecto, cálculo de costos, implementación del código automáticamente, compilación automática, entre otros. De acuerdo con las fases del ciclo de desarrollo que cumplen, estas herramientas pueden ser clasificadas como:

Herramientas de caso superior o Upper case: herramientas que ayudan en las fases de planificación, análisis de requisitos y estrategia del desarrollo, usando, entre otros diagramas UML.

Herramientas de caso medio o Middle Case: herramientas para automatizar tareas en el análisis y diseño de la aplicación.

Herramientas de caso inferior o Lower Case: herramientas que semi-automatizan la generación de código, crean programas de detección de errores, soportan la depuración de programas y pruebas. Además, automatizan la documentación completa de la aplicación. Aquí pueden incluirse las herramientas de desarrollo rápido de aplicaciones.

3.6 LENGUAJE DE DOMINIO ESPECÍFICO

Un lenguaje de dominio específico (en inglés domain-specific language, DSL) es un lenguaje de programación creado para resolver problemas de un dominio en particular, no siendo un concepto nuevo ya que siempre han existido los lenguajes de propósito específico.

A diferencia de los lenguajes de propósito general como Java o C, los lenguajes de dominio específico no son creados para representar cualquier tipo de problemas, ya que como su nombre lo indica se limitan a resolver los problemas que se encuentran en un solo campo o dominio, lo cual le proporciona a este lenguaje ciertas ventajas y desventajas respecto a un lenguaje de propósito general, crear un DSL junto con un software que lo pueda interpretar vale la pena siempre y cuando los problemas que resuelve no puedan ser modelados fácilmente en los lenguajes de propósito general existentes, dentro de los DSLs más populares que podemos encontrar están, el Lenguaje de Consulta Estructurado (SQL) y el Lenguaje de Marcado Hipertextual (HTML) que definen elementos y reglas para modelar bases de datos y páginas web respectivamente.

3.6.1 Componentes de un DSL. Un lenguaje de dominio específico posee 3 componentes:

Semántica: La semántica de un lenguaje de programación es el conjunto de reglas que determinan el significado o la interpretación que se da al código escrito en el lenguaje.

Sintaxis Abstracta: La sintaxis abstracta especifica la estructura del lenguaje, independientemente de cualquier representación o codificación particular, es decir, las construcciones, propiedades y conectores que pueda tener dicho lenguaje.

Sintaxis Concreta: La sintaxis concreta es necesaria para especificar la notación específica con la que los usuarios del lenguaje podrán utilizarlo, es importante resaltar que una misma sintaxis abstracta podría tener diferentes sintaxis concretas.

3.6.2 Principios de un DSL. Un DSL debe contar con los siguientes principios [4]:

- Proveer buenas abstracciones para el desarrollador, ser simple, intuitivo y facilitar el trabajo.
- No depender de la experticia de una sola persona para su adopción y uso.
- Debe evolucionar y mantenerse actualizado respecto a las necesidades del usuario y el contexto.
- Debe acompañarse de herramientas y métodos para maximizar la productividad de los expertos en el dominio.
- Estar abierto a extensiones, pero cerrado a modificaciones.

3.6.3 Clasificación de un DSL. Los DSL se pueden clasificar así [4]:

Según su enfoque:

- **Vertical:** para un campo específico de la industria (Agricultura, Automotriz)
- **Horizontal:** Para varios grupos de aplicaciones de diferentes campos (SQL, HTML)

Según su estilo:

- **Declarativo:** expresa la lógica computacional sin describir su flujo de control, el qué antes que el cómo.

- **Imperativo:** describe el flujo de control para guiar la ejecución.

Según su Notación:

- **Gráfica:** Modelos visuales.
- **Textual:** Modelos textuales.

Según su Interioridad:

- **Interno:** Se basa en la modificación de otra sintaxis o lenguaje.
- **Externo:** Tiene su propia sintaxis.

Según su Ejecución:

- **Interpretado:** el modelo se lee y se ejecuta en tiempo real.
- **Generado:** se genera el código fuente a partir del modelo para ser compilado y ejecutado después.

3.7 ENTORNO PARA EL DESARROLLO DE UN DSL

Desarrollar un lenguaje de dominio específico es una tarea en la cual se debe tener en cuenta la semántica del lenguaje, si es un lenguaje de tipado estático o dinámico, el paradigma del lenguaje, los problemas que se deben resolver con el lenguaje (Dominio), si es interpretado o compilado y demás características del lenguaje, finalmente se debe construir un analizador sintáctico para que advierta de errores o fallos en los programas escritos en el lenguaje.

En este orden de ideas, diseñar el lenguaje y crear las herramientas para hacerlo funcionar de forma que pueda resolver la mayor parte de los problemas presentes en su dominio, fácilmente se puede convertir en un proyecto de grado por sí mismo, sin embargo, existen herramientas que facilitan esta tarea, proporcionando un entorno de desarrollo en el cual es posible diseñar y probar el DSL.

El entorno de desarrollo Eclipse en conjunto con el framework Xtext proveen un conjunto de herramientas muy poderosas para dicha tarea, generando un poderoso

entorno de desarrollo tanto para lenguajes de propósito general, como para lenguajes de dominio específico.

3.7.1 Eclipse. Eclipse es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma, proporciona un entorno de trabajo para desarrollar software en una amplia variedad de plataformas como Java, C++, C, PHP, entre otras.

3.7.2 Xtext. Es un framework para el desarrollo de lenguajes de programación y lenguajes de dominio específico, en el cual los usuarios pueden definir su lenguaje utilizando un lenguaje gramatical proporcionado por Xtext, en el cual se especifica la semántica y la sintaxis del lenguaje que se desea describir, cuenta con una completa infraestructura que incluye un analizador sintáctico, un enlazador, un inspector de código y un compilador, actualmente Xtext cuenta con soporte para plataformas de desarrollo como Eclipse o IntelliJ IDEA.

Analizador Léxico: Xtext utiliza un potente analizador conocido como ANTLR por sus siglas en inglés y cuya traducción sería otra herramienta para el reconocimiento de idiomas, es una herramienta ampliamente utilizada para construir lenguajes, herramientas y marcos de trabajo, el analizador se encarga de asegurarse que el programa respeta la sintaxis del lenguaje.

Con este objetivo, el analizador se encarga de romper el programa en partes o tokens, cada símbolo es un elemento atómico en el lenguaje, puede ser una palabra clave (Como la palabra "class" en Java), un identificador o un valor fijo. Otros tipos de tokens del lenguaje son operadores (como los operadores aritméticos y operadores de comparación) y separadores (como paréntesis).

El proceso de convertir una secuencia de caracteres en una secuencia de tokens se denomina análisis léxico y el programa o procedimiento que realiza ese análisis se llama analizador léxico o simplemente un escáner. Este análisis se implementa generalmente mediante la sintaxis de expresiones regulares. [5]

Analizador Sintáctico: Tener la secuencia de tokens del archivo de entrada no es suficiente, debemos asegurarnos de que forman una declaración válida en nuestro lenguaje, es decir, respetan la estructura sintáctica esperada por el lenguaje. Esta fase se llama análisis sintáctico. El programa o procedimiento que realiza este análisis se llama analizador. [5]

Árbol de Sintaxis Abstracta (AST): El análisis sintáctico es solo la primera etapa en la implementación de un lenguaje de programación, cuando comprobamos que el programa es correcto desde el punto de vista sintáctico, la implementación tendrá que decidir qué hacer con los elementos del programa.

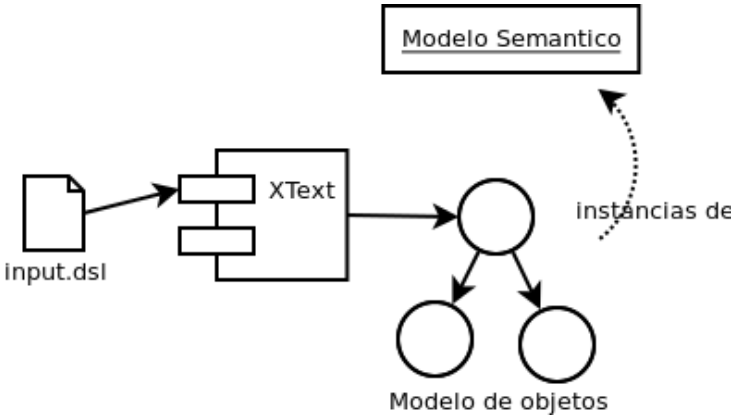
En principio, la corrección general de un programa no siempre se puede realizar durante la etapa de análisis sintáctico, por ejemplo, una de las comprobaciones que no se pueden realizar durante esta etapa por ejemplo en java, es que el usuario no asigne un valor de cadena a una variable entera, estas comprobaciones hacen parte del análisis semántico de un programa, que implica manipular las variables que se declaran en el programa, es por esto que durante el análisis sintáctico se crea una representación del programa en memoria, esta representación es una estructura de árbol llamada árbol de sintaxis abstracta (AST) por sus siglas en ingles.

El AST representa la estructura sintáctica abstracta del programa, y una vez se encuentra cargado en memoria podemos realizar todas las comprobaciones semánticas adicionales que requiera el programa, de forma que si todo es correcto podemos utilizar el AST para la etapa final de implementación del programa, que puede ser la interpretación del programa o generación de código.

Xtext hace uso de EMF (Eclipse Modeling Framework) para analizar la gramática del lenguaje y crear un modelo semántico en la forma de un conjunto de clases Java que representan los elementos de nuestro lenguaje, la representación del AST creado por Xtext es un conjunto de objetos instancias de las clases del modelo semántico. [5]

3.7.3 Eclipse Modeling Framework. Eclipse Modeling Framework (EMF) es una herramienta de modelado basada en eclipse que facilita la generación de código para construir herramientas y aplicaciones basadas en un modelo de datos estructurado, EMF proporciona herramientas y soporte en tiempo de ejecución para producir un conjunto de clases java para el modelo. [6]

Figura 4. Gramática Xtext y Modelado Semántico.



Programación Avanzada con Objetos [7]

3.8 ENTORNO PARA EL DESARROLLO DE HERRAMIENTA CASE

Cuando el código escrito por el usuario es sintácticamente correcto y aceptado por el lenguaje, se crea el modelo de objetos instancias de las clases del modelo semántico creado por Xtext, debemos programar la herramienta para procesar nuestro modelo, la programación de dicha herramienta se hace haciendo uso de un lenguaje de programación de alto nivel de uso general basado en Java, conocido como Xtend.

3.8.1 Xtend. Xtend es un lenguaje de programación de alto nivel de uso general para la máquina virtual de java, sintácticamente y semánticamente Xtend tiene sus raíces en el lenguaje de programación Java.

4. ESTADO DEL ARTE

Con el creciente movimiento del llamado Open-Data [8] cada vez son más las organizaciones privadas y gubernamentales que publican sus datos, permitiendo a cualquier persona interesada hacer uso de ellos. Por otra parte, el gobierno colombiano apuesta por una plataforma de intercambio de datos conocida como Datos Abiertos Colombia [9] impulsada por el ministerio de las Tecnologías de la Información y la Comunicación - TIC, donde tanto entidades gubernamentales como personas pueden publicar sus conjuntos de datos de forma que sean accesibles al público en general y donde ya se han gestado propuestas interesantes para el análisis de los fenómenos que ocurren en nuestro país, basándose en una plataforma conocida a nivel mundial que provee servidores y una completa infraestructura tecnológica para el manejo y la publicación de los datos, la plataforma de datos abiertos de Socrata [10], que es utilizada ampliamente por las ciudades y gobiernos en el mundo; actualmente la plataforma presta sus servicios a la cuidad de New York, el gobierno de México, el gobierno Colombiano y muchas más entidades interesadas en publicar su información.

Gran parte de las organizaciones en Colombia y el mundo utilizan herramientas como Excel para visualizar y manejar sus datos, estas organizaciones no requieren de soluciones a la medida, puesto que no son lo suficientemente grandes para manejar un volumen de datos que lo amerite, por otra parte, las organizaciones con una envergadura más grande han descubierto la ventaja competitiva que les brinda contar con soluciones tecnológicas para el manejo de sus datos, es allí donde surgen plataformas como Plotly [11] que son herramientas especializadas en tomar las fuentes de datos y crear visualizaciones adecuadas proporcionando interfaces de usuario sencillas para el manejo de los datos, también cabe mencionar la plataforma Tableau [12] que al igual que Plotly es una plataforma diseñada para el manejo y la interpretación de los datos; existen muchas más herramientas en el

mercado que buscan brindar soluciones a esta necesidad creciente de interpretar correctamente los datos, pero todas estas herramientas en su mayoría son de pago y funcionan como una caja negra para sus usuarios, además todas requieren de una curva de aprendizaje para aprovechar todo su potencial y por lo general se enfocan en una plataforma en específico, lo que hace que muchas organizaciones generen software a la medida para satisfacer sus necesidades, creando así un mercado para el desarrollo de aplicaciones orientadas a la visualización de datos.

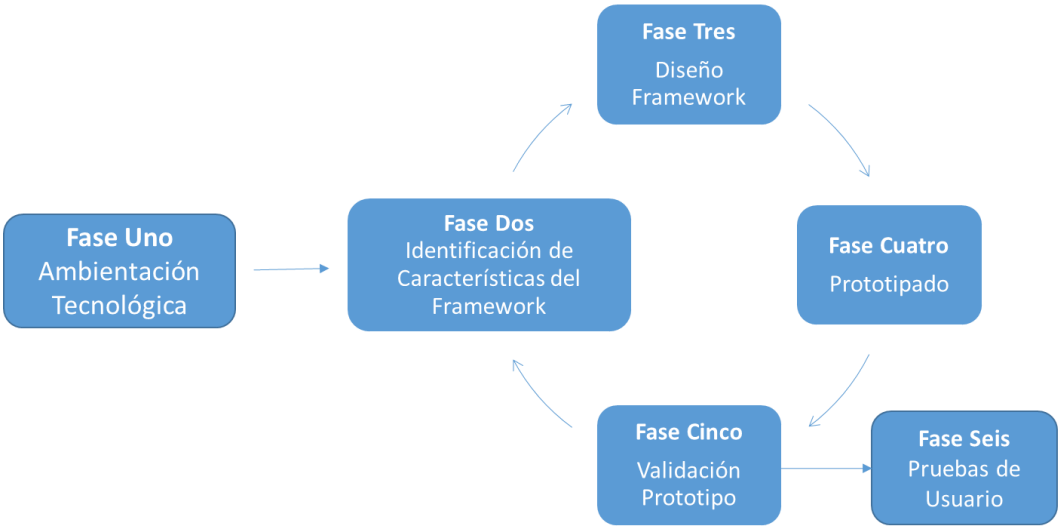
Desde la revisión bibliográfica realizada y la evaluación de las alternativas existentes en el mercado, se observa que existe gran variedad de librerías que permiten realizar gráficos a partir de datos para diferentes plataformas, pero que cada librería requiere de un proceso de aprendizaje y adaptación de los datos al formato utilizado, además, generalmente el resultado solo puede ser utilizado en una plataforma, aquellas herramientas que permiten generar el gráfico para dos o más plataformas trabajan a manera de caja negra para el desarrollador y generan dependencia de dicha herramienta, limitando las posibilidades para los desarrolladores, sin mencionar que la gran mayoría de herramientas son de pago.

Por tanto, el valor agregado de este proyecto es ofrecer la posibilidad de generar código base para distintas plataformas completamente adaptable para los desarrolladores con base en un mismo modelo, y aprovechando el nivel superior de abstracción brindado por los lenguajes de dominio específico mejorar significativamente la productividad, además de la calidad del código, ya que la fiabilidad de la generación automática en comparación con la codificación manual reducirá el número de errores en el resultado, todo el código generado permite visualizar los datos directamente desde los servicios web que proveen los datos de interés para la organización.

5. METODOLOGÍA

La metodología de investigación y desarrollo que se trabajará a lo largo del proyecto está conformada por 6 fases, la primera fase busca realizar una ambientación tecnológica, seguida por cuatro fases basadas en el desarrollo de prototipos evolutivos que permita desarrollar un prototipo que durante la fase 6 será llevado a pruebas de usuario, con el fin de verificar el cumplimiento de los objetivos propuestos, a continuación se puede apreciar un esquema del flujo de trabajo que tendrán cada una de estas fases:

Figura 5. Esquema Metodología de Trabajo



5.1 FASE 1: AMBIENTACIÓN TECNOLÓGICA

En esta fase se investigará todo el marco teórico y el estado del arte bajo el cual se enmarca el desarrollo del proyecto, de manera que los implicados en el mismo puedan apropiarse del tema conociendo a profundidad los conceptos claves para el desarrollo del proyecto, además de estudiar las herramientas disponibles en el mercado que puedan cumplir un objetivo similar al que cumpliría la herramienta en desarrollo.

Actividades:

- A1.1 Investigación fundamentos teóricos relacionados con el proyecto.
- A1.2 Estudio de herramientas a utilizar para conocer su uso y funcionamiento.
- A1.3 Sondeo de herramientas similares disponibles en el mercado.

Productos

- P1.1 Fundamentos teóricos y estado del arte.

5.2 FASE 2: IDENTIFICACIÓN CARACTERÍSTICAS DEL FRAMEWORK

La segunda fase está enfocada a identificar el dominio y el alcance del proyecto, definiendo las características que posea el Framework, las funcionalidades para cada uno de los usuarios, las tecnologías que se van a utilizar y la arquitectura de la plataforma a desarrollar.

Actividades:

- A2.1 Definición del alcance del proyecto y características del Framework.
- A2.2 Definición del conjunto de funcionalidades del Framework.
- A2.3 Especificación de la arquitectura del Framework.

Productos

- P2.1 Descripción del Framework, tecnología en la cual se basará y funcionalidades.
- P2.2 Arquitectura del framework.

5.3 FASE 3: DISEÑO FRAMEWORK

El Framework consta de dos componentes fundamentales, un lenguaje de descripción de visualizaciones y una herramienta generadora de visualizaciones, en esta fase diseñamos cada uno de los componentes, de manera que cada uno pueda cumplir sus objetivos y puedan integrarse.

Actividades:

- A3.1 Diseñar el lenguaje de descripción de visualizaciones.
- A3.2 Diseñar herramienta generadora de visualizaciones.
- A3.3 Revisión de arquitectura del framework.

Productos

- P3.1 Especificación y documentación de la gramática del lenguaje de descripción.
- P3.2 Diseño de la herramienta generadora de visualizaciones.

5.4 FASE 4: PROTOTIPADO

En esta fase se implementó el diseño creado en las fases anteriores mediante el desarrollo de un prototipo, este prototipo cumple parcial o totalmente las funcionalidades del framework dependiendo de la madurez del mismo, cabe recordar que el desarrollo está basado en modelos evolutivos por lo cual cada modelo desarrollado se espera que sea más maduro y más acertado al objetivo final.

Actividades:

- A4.1 Desarrollo de prototipo del Framework.

Productos

- P4.1 Prototipo del Framework.

5.5 FASE 5: PRUEBAS DE VALIDACIÓN DE PROTOTIPO

Una vez desarrollado un prototipo es necesario conocer si cumple con las funcionalidades para las cuales fue desarrollado, o si las funcionalidades que ya se tenían se vieron afectadas por los cambios realizados, es por esto que se realizan pruebas al prototipo para conocer estos factores.

Actividades:

- A5.1 Diseño de un plan de pruebas para el prototipo.
- A5.2 Realización las pruebas del prototipo.

Productos

- P5.1 Plan de pruebas del prototipo
- P5.2 Resultados pruebas del prototipo.

5.6 FASE 6: PRUEBAS DE USUARIO.

Cuando se cuente con un prototipo que cumpla las funcionalidades mínimas, es necesario validar que dicho prototipo permite reducir la complejidad en el diseño de visualizaciones de datos, y al mismo tiempo disminuya el tiempo de desarrollo de las aplicaciones que implementen dichas visualizaciones para una plataforma en específico, por esto se proponen un conjunto de pruebas de usuario para validar estos objetivos del proyecto.

Actividades:

- A6.1 Diseñar pruebas de usuario.
- A6.2 Realizar pruebas de usuario.

Productos

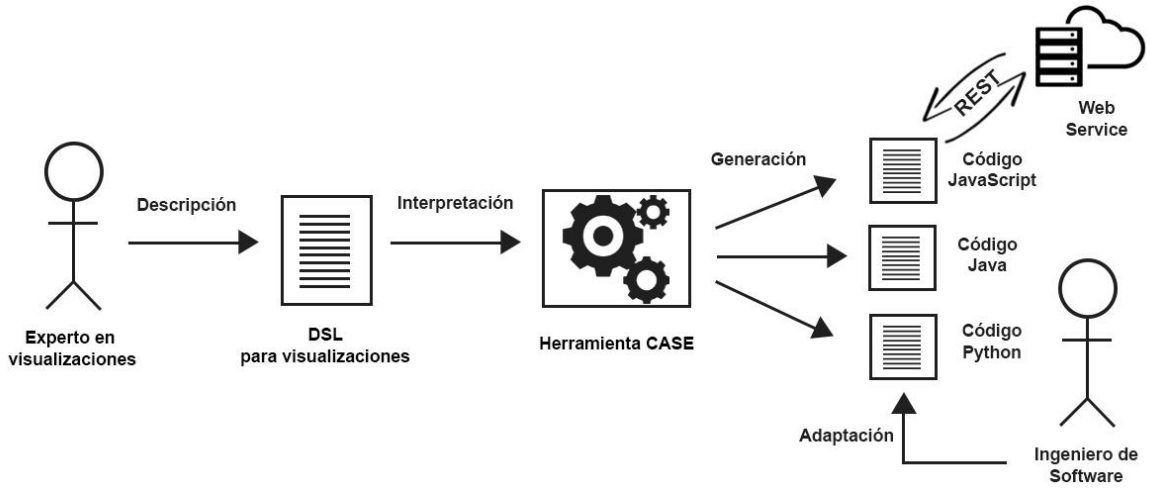
- P6.1 Diseño de las pruebas de usuario.
- P6.2 Resultados pruebas de usuario.

6. DESARROLLO DEL PROYECTO

Este apartado presenta el enfoque general proyecto que contempla cada una de las etapas del diseño general del framework, posteriormente se aborda el diseño especifico de cada uno de los componentes del sistema y finalmente la noción de extensibilidad para la herramienta propuesta.

6.1 ENFOQUE DEL PROYECTO

Figura 6. Enfoque general del proyecto.



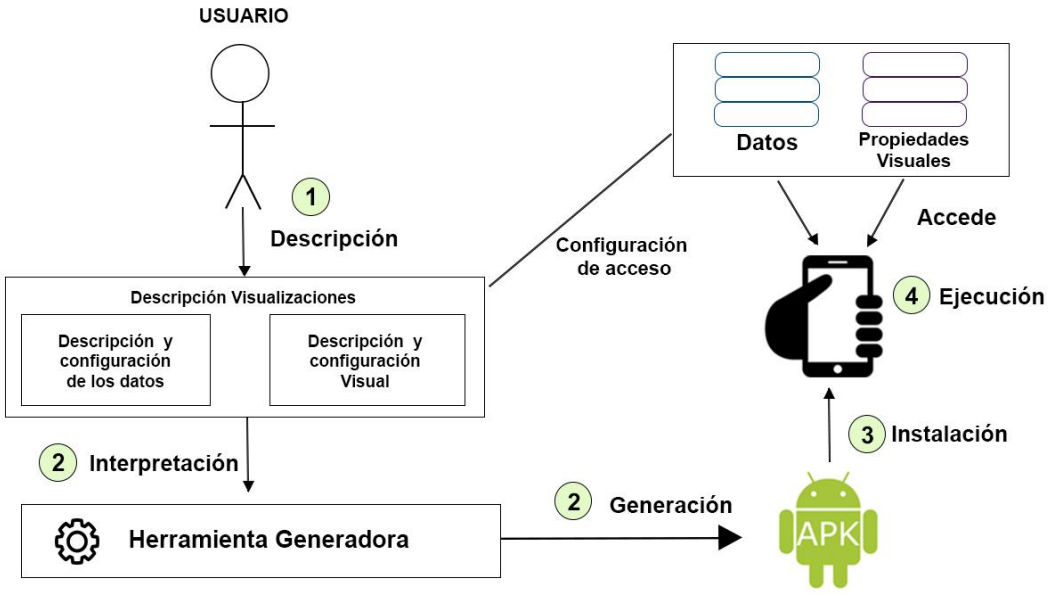
El enfoque general del proyecto es crear un framework para optimizar el desarrollo de aplicaciones orientadas a la visualización de datos, el cual está conformado por un lenguaje de domino específico - DSL, que permite al experto en visualizaciones realizar una descripción textual de los gráficos o visualizaciones necesarios en la aplicación a desarrollar, una herramienta CASE capaz de interpretar el DSL y generar código base para diferentes plataformas de uso general, el código generado estará programado para conectarse a determinados servicios web y obtener los datos necesarios para generar las visualizaciones descritas, de esta forma el ingeniero de software podrá tomar el código base para incluir nuevas características de la aplicación o realizar las modificaciones necesarias para obtener la aplicación terminada.

Para crear la herramienta que cumpliera el objetivo del proyecto se realizaron 4 diseños que fueron evolucionando a partir del enfoque presentado, en las siguientes páginas se aprecia cada uno de los diseños y el enfoque de los mismos.

6.1.1 Diseño del framework. Los diseños realizados durante el proyecto son los siguientes:

6.1.1.1 Primer Diseño. El primer diseño de la herramienta está basado en el enfoque que tenía el proyecto en un primer momento, el cual buscaba crear una herramienta enfocada en aplicaciones móviles fáciles de generar y cuyas visualizaciones se generan de forma dinámica con base en una serie de parámetros establecidos en un repositorio de propiedades visuales, el flujo de trabajo de este diseño estaba representado en 4 fases que pueden apreciarse en la figura 7:

Figura 7. Primer diseño flujo de trabajo framework.



Fase 1 - Descripción: El usuario describe la configuración de los datos y la configuración visual de la aplicación a generar, en este diseño aún no se especifica la forma en la cual el usuario realiza esta descripción.

Fase 2 - Interpretación y Generación: La herramienta generadora interpreta la configuración establecida por el desarrollador y genera un instalador APK (Android Application Package o aplicación empaquetada de Android).

Fase 3 - Instalación: Se transfiere el APK al dispositivo móvil y se instala en el mismo. El APK cuenta con un código base que interpreta los datos obtenidos y las propiedades visuales proporcionadas.

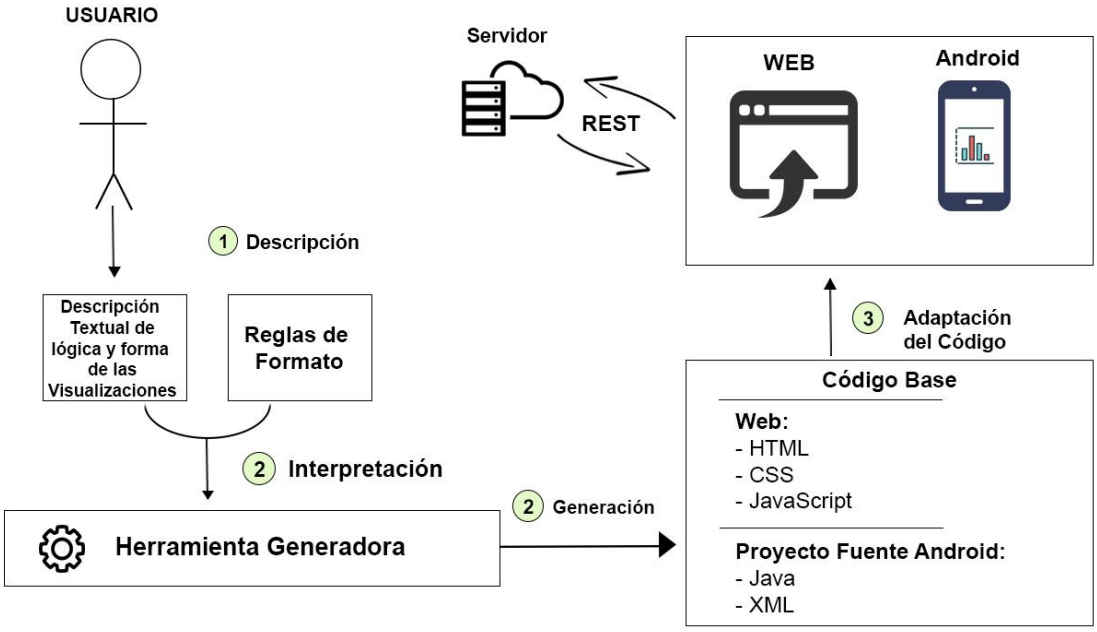
Fase 4 - Ejecución: El usuario final accede a la aplicación, la cual se conecta al repositorio de datos y al repositorio donde están definidas las propiedades de la visualización, para generarlas al usuario en tiempo real.

Este primer diseño buscaba sentar las bases para comprender el flujo de trabajo del framework sin entrar en detalle en los tipos de usuario del sistema, la noción de la generación estaba pensada en un primer momento como la creación de un paquete de instalación para Android, y la herramienta en general buscaba dar soporte únicamente a esta plataforma, la aplicación generada se complementaba con el repositorio de propiedades visuales que permitía manipular la forma en la cual se generaban las visualizaciones en las aplicaciones ya generadas.

Se identificaron varias falencias en este enfoque, entre las cuales resaltaba que, al generarse directamente un paquete de instalación este no podría ser modificado para añadir más características a la aplicación generada, limitando así el alcance y la utilidad de la herramienta.

6.1.1.2 Segundo Diseño. El segundo diseño considero un rol de usuario como desarrollador o como experto en visualizaciones, en este diseño se estableció que la descripción de las visualizaciones se realizaría de forma textual, y que la herramienta generaría código base para diferentes plataformas que pudiera ser adaptado a las necesidades de un desarrollo en particular, la arquitectura del segundo diseño se puede contemplar en la figura 8.

Figura 8. Segundo diseño flujo de trabajo Framework.



Fase 1 - Descripción: En este diseño se establece que el proceso de descripción de las visualizaciones se realizará de forma textual en algún lenguaje o formato, que será proporcionado al usuario y que estará basado en unas reglas previamente definidas.

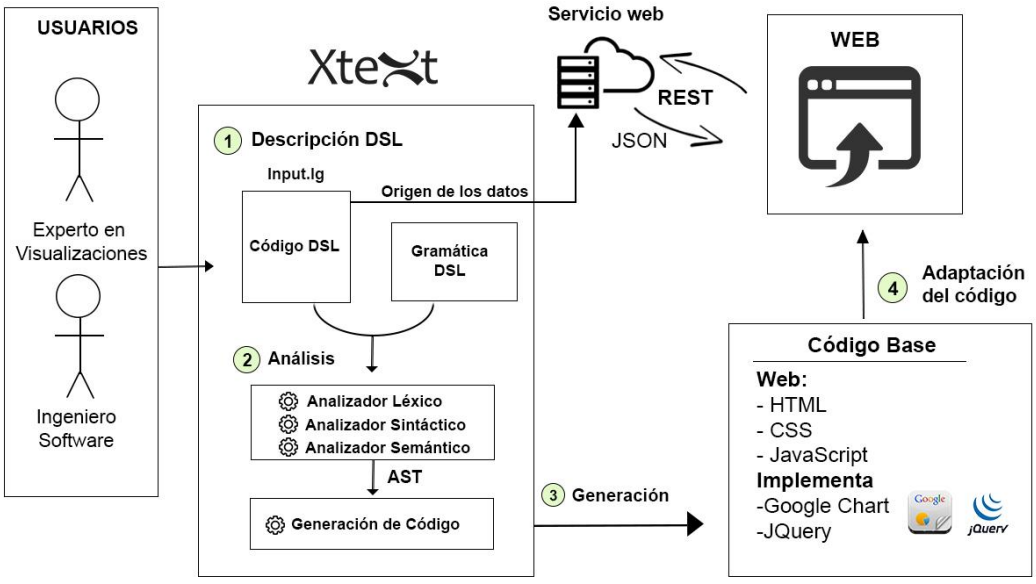
Fase 2 - Interpretación y Generación: La herramienta generadora ya no está pensada para crear un instalador como en el diseño anterior, a partir de este diseño, se enfoca en generar código para la plataforma web y un proyecto base con las clases Java y Archivos XML para Android, con base en las descripciones del usuario y las reglas del formato.

Fase 3 - Adaptación del Código: El código generado permite la visualización de los gráficos con base en los datos configurados, sin embargo, puede que se requiera adaptarlo de acuerdo con las necesidades de la aplicación que se está desarrollando, una vez la aplicación cumpla con los requerimientos del desarrollo podrá ser usada en la plataforma para la cual fue diseñada.

El segundo diseño fue realizado con un enfoque más aproximado al enfoque real del proyecto, tiene muchas ventajas respecto al diseño anterior, entre las cuales resalta la definición de la forma de descripción como textual, igualmente también cuenta con falencias ya que debido a la complejidad y numero de versiones de compilación con las que cuenta Android, generar un código base adaptable al formato de los proyectos de esta plataforma sería poco factible sin dedicar gran parte del proyecto a ello.

6.1.1.3 Tercer Diseño. La tercera propuesta concretó varios aspectos respecto a las tecnologías implicadas durante el desarrollo y la interacción de los usuarios con el sistema, teniendo en cuenta los argumentos planteados en el análisis del diseño anterior se tomó la decisión de descartar Android como plataforma objetivo para la generación de código.

Figura 9. Tercer diseño flujo de trabajo Framework



Se planteó el desarrollo de un lenguaje de dominio específico (DSL) haciendo uso de un Framework diseñado para Eclipse conocido como Xtext, que brinda un

entorno de trabajo para el desarrollo de lenguajes de domino específico y lenguajes de programación en general, el flujo de trabajo de la tercera propuesta está divido en cuatro fases:

Fase 1 - Descripción DSL: Haciendo uso del lenguaje de dominio específico los usuarios describen las visualizaciones de forma textual, el usuario puede realizar descripciones del modelo con un nivel de abstracción superior al utilizado en un lenguaje de uso general, esto aporta grandes ventajas entre la cual se encuentra, la facilidad en la colaboración de los roles de usuarios implicados para el desarrollo del proyecto, todo el código escrito por el usuario será almacenado en ficheros con extensión .lg la cual fue seleccionada por ser fácil de recordar.

Fase 2 - Análisis: Xtext proporciona una serie de analizadores que corroboran en tiempo real la valides del código escrito por el usuario, en caso de encontrar errores en el código el entorno indica al usuario dichos errores y las posibles causas, si el código es correcto se genera un árbol de sintaxis abstracta AST que será utilizado para la generación del código base.

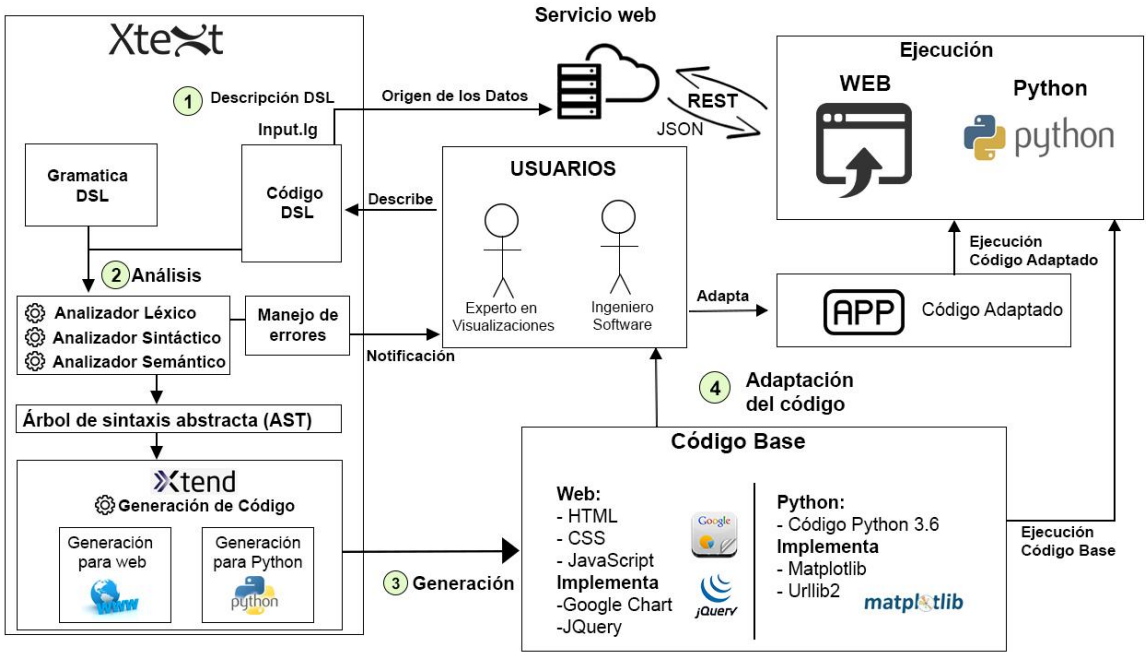
Fase 3 - Generación de Código: En el tercer diseño la generación de código se propone con base en el árbol de sintaxis abstracta generado en la fase 2.

Fase 4 - Adaptación del código: El código base generado puede ser adaptado por el desarrollador a sus necesidades, eliminando o añadiendo características, además, puede aprovechar las funcionalidades que proveen las librerías bases seleccionadas.

El aspecto más relevante en el tercer diseño es la decisión de crear un DSL para que el usuario pueda describir las características del modelo de entrada, que posteriormente será analizado y transformado en el código de salida ejecutable en el entorno web.

6.1.1.4 Diseño Final. El diseño final de la interacción entre todos los componentes de la herramienta gira en torno a los usuarios del sistema.

Figura 10. Diseño Final Framework.



La propuesta final genera código base para aplicaciones destinadas al entorno web o basadas en Python, estas dos plataformas fueron seleccionadas por su amplio uso en el desarrollo de aplicaciones para el mercado, el diseño del marco de trabajo propuesto se explica en 4 fases:

Fase 1 – Descripción del modelo en el DSL: Haciendo uso de la sintaxis del lenguaje de dominio específico, los usuarios realizan la descripción textual de las visualizaciones, que se almacenan en ficheros con extensión lg.

Fase 2 – Análisis: Los analizadores validan el código escrito por el usuario con base en las reglas del lenguaje, si se detectan errores estos son notificados a los usuarios en tiempo real desde el entorno de trabajo, por otra parte, si todo es correcto se genera el árbol de sintaxis abstracta que será utilizado por la herramienta generadora de código.

Fase 3 – Generación: Xtext permitió crear una herramienta generadora de código haciendo uso de un lenguaje de alto nivel basado en Java conocido como Xtend, se requiere de un modelo de entrada que en este caso es el AST, que permite analizar cada visualización y características definidas por el usuario para generar el código base, este código se genera cada vez que el usuario guarda una nueva versión de la descripción.

Fase 4 – Adaptación del código: El código base generado puede ser ejecutado para validar la obtención del resultado deseado, este código debe ser adaptado a las necesidades específicas de las aplicaciones que estén desarrollando los usuarios, la complejidad de esta última fase depende totalmente del resultado final deseado.

6.2 COMPONENTES DEL FRAMEWORK

6.2.1 Lenguaje de dominio específico. Los lenguajes de dominio específico se centran en resolver un único problema dentro de su dominio, por lo tanto, es necesario que el lenguaje permita describir de forma concreta y con un alto nivel de abstracción las visualizaciones que se desean generar en un lenguaje de propósito general como Python, las nociones implicadas en el diseño del lenguaje son las siguientes:

Conjuntos de datos: un conjunto de datos contiene los valores de las variables relacionadas que se desean estudiar con ayuda de la visualización, es importante que el lenguaje permita describir la fuente del conjunto de datos y las variables que se desean graficar, como se especifica en el diseño final de la herramienta, los datos proporcionados por la fuente vendrán en formato JSON.

Gráfico o visualización: Existen muchas formas de representar los datos, por lo cual es importante seleccionar los gráficos adecuados, de acuerdo con la

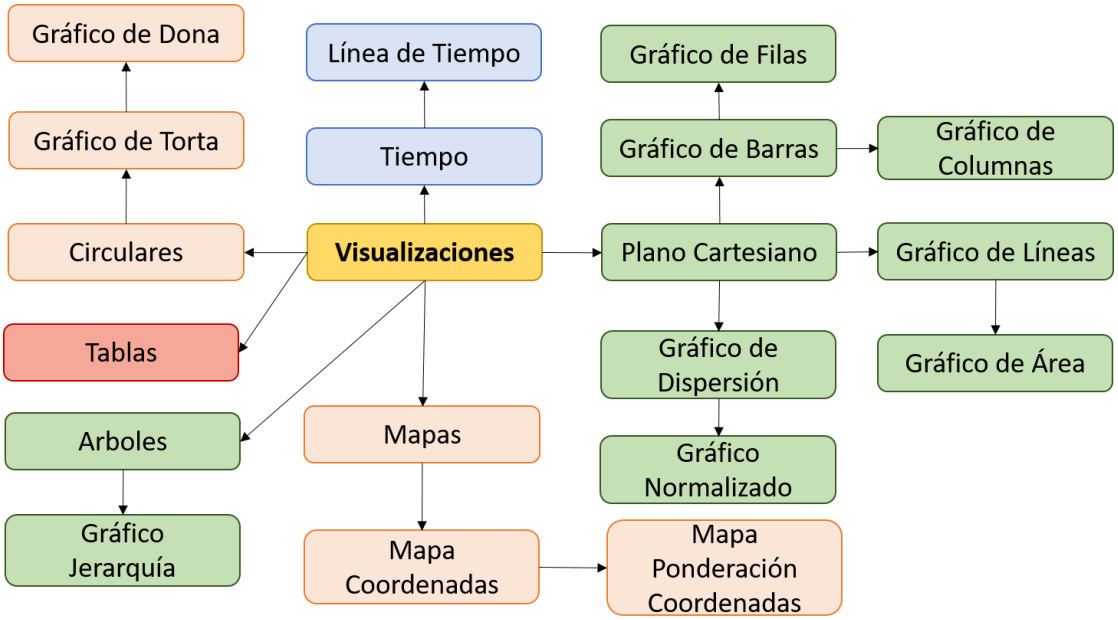
investigación realizada se seleccionó un conjunto de los gráficos más comunes en este tipo de tareas.

Reglas del lenguaje: Es necesario establecer restricciones para el tipo de visualizaciones y la forma en la cual serán descritas dichas visualizaciones, por lo tanto, se establece un conjunto de reglas para la elaboración del lenguaje.

1. Los conjuntos de datos serán representados en el lenguaje como tuplas.
2. Las tuplas podrán asociarse con una única fuente de datos.
3. Dependiendo de la cantidad de variables descritas, la tupla pertenecerá a una clase definida por una palabra clave.
4. Cada tupla será declarada haciendo uso de la palabra clave correspondiente a su clase seguida por el nombre de la tupla.
5. La fuente de datos, el tipo y el nombre de las variables serán declaradas dentro de la declaración de la tupla asociada.
6. Los gráficos serán descritos en el lenguaje haciendo uso de una palabra clave y un nombre.
7. Cada gráfico será asociado a una única tupla que pertenezca a una clase compatible con el gráfico, la asociación con la tupla será la primera característica disponible en el gráfico seguido por demás propiedades del gráfico.
8. Las propiedades de los gráficos serán de dos tipos, las propiedades que asignen un valor definido por el usuario como el título del gráfico, y las propiedades con un conjunto de valores por defecto como true o false.
9. El lenguaje debe poder seleccionar mediante una característica global la plataforma objetivo para la generación de código, por defecto se generará para todas las plataformas disponibles.

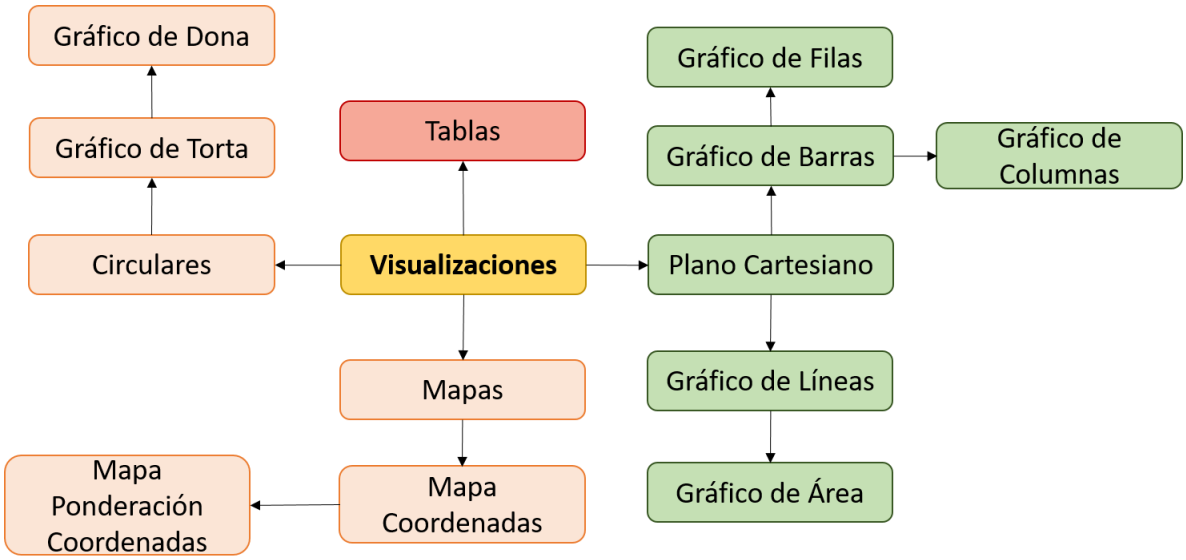
Selección de visualizaciones: Se debe establecer el conjunto de visualizaciones que pueden ser descritas en el DSL, el estudio inicial de gráficos de uso frecuente basado en las herramientas y librerías disponibles en el mercado, determino las familias de gráficos más utilizados para crear estas visualizaciones que pueden ser apreciadas en la figura 11:

Figura 11. Conjunto de visualizaciones más comunes.



Para la propuesta fueron seleccionadas 4 familias de visualizaciones frecuentemente utilizadas, que se muestran en la figura 12:

Figura 12. Familias de visualizaciones frecuentes seleccionadas.



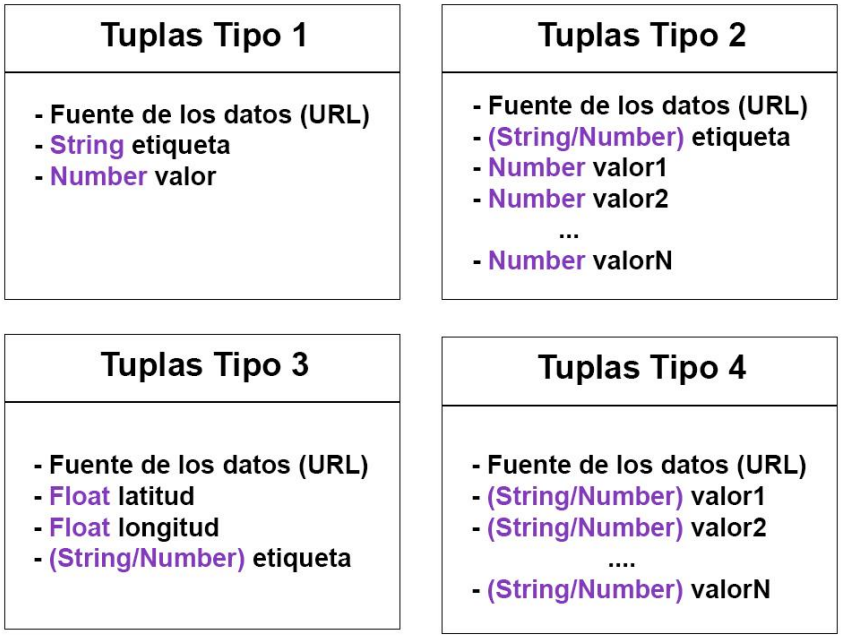
Debido a que algunos gráficos como el gráfico de Dona son variaciones de otros gráficos y pueden ser obtenidos añadiendo, modificando o eliminando alguna

propiedad de otro gráfico, se definieron 5 gráficos bases que pueden ser descritos en el lenguaje: el gráfico de torta o **PieChart**, el gráfico de línea o **LineChart**, el gráfico de barras o **BarChart**, el gráfico de coordenadas o **MapChart** y el gráfico de tabla o **TableChart**.

Adicionalmente se añade una noción de gráfico conocido como **Dashboard** o tablero de visualización, este elemento podrá contener uno o más gráficos y mostrarlos de forma simultánea.

Clases de conjuntos de datos: una vez definidas las visualizaciones que harán parte del lenguaje y las reglas del mismo, es necesario definir los conjuntos de datos que pueden ser descritos en el lenguaje, de acuerdo con las características de los datos requeridos por cada una de las visualizaciones seleccionadas se definieron cuatro clases de tuplas o conjuntos de datos representadas en la figura 13:

Figura 13. Conjuntos de datos aceptados por el lenguaje



Los conjuntos de datos seleccionados representados en cada tipo de tupla corresponden a uno o más gráficos, como se describe en las reglas del lenguaje, las variables son descritas mediante el tipo de dato y el nombre de la variable, la relación entre los gráficos y los conjuntos de datos se puede observar en la tabla 1.

Tabla 1: Relación conjuntos de datos - gráficos seleccionados.

Conjunto de Datos	Descripción	Gráficos Compatibles
Tuplas tipo 1	Conjunto de datos de dos variables (etiqueta, valor) relacionadas.	Gráfico de Torta
Tuplas tipo 2	Conjunto de datos de dos variables o más variables relacionadas, se utiliza en conjunto donde uno o más valores son asignados a una etiqueta.	Gráfico de Barras. Gráfico de Línea.
Tuplas tipo 3	Conjunto de datos de tres variables, contiende registros de coordenadas (latitud, longitud) y etiquetas de dichas coordenadas	Mapa
Tuplas tipo 4	Conjunto de una o más variables.	Tablas

Una vez seleccionado el dominio, los elementos y las reglas del lenguaje, el diseño podría considerarse completo, sin embargo, el diseño aquí presentado es el resultado de varias iteraciones entre diseño e implementación que permitieron definir estas características para que dieran respuesta a las necesidades del proyecto.

6.2.2 Herramienta generadora de código (CASE). La herramienta generadora de código debe trabajar de acuerdo al árbol de sintaxis abstracta de cada programa y al modelo de los datos estructurados representados en el meta-modelo Ecore.

Pero Xtext trabaja en conjunto con Eclipse Modeling Framework (EMF), el cual proporciona un modelo semántico en forma de clases java, posteriormente parsea un archivo en nuestro lenguaje y genera un modelo de objetos instancias de estas clases con el cual trabajar, en primer lugar debemos definir las plataformas objetivo para la generación de código y posteriormente el modelo de trabajo de la herramienta.

Selección de plataformas objetivo: la selección de las plataformas está basada en la facilidad para la generación del código base y la posibilidad de visualizar rápidamente el resultado, las plataformas seleccionadas fueron:

- **WEB:** la web proporciona un entorno de ejecución sencillo que una vez generada la descripción de las visualizaciones en html, css y Javascript, los ficheros que contengan dichas descripciones podrán ser interpretados por la gran mayoría de los navegadores actuales, permitiendo al usuario visualizar rápidamente el resultado.
- **Python**: los programas o script de Python son relativamente sencillos de generar debido a la sencillez y elegancia del lenguaje, de la misma forma el código generado puede ser interpretado fácilmente haciendo uso de la consola del sistema, un par de comandos y por supuesto el intérprete del lenguaje o cargando el fichero a un entorno de trabajo como Jupyter Notebook.

Especificaciones de la herramienta: la herramienta de generación de código basada en Xtext debe contar con los siguientes requerimientos:

Requerimientos funcionales:

- La herramienta debe generar código con base en las visualizaciones definidas por el usuario y la plataforma seleccionada.
- La generación de código debe ser ejecutada cada vez que se guarden los cambios en el proyecto.
- Si el usuario no define la plataforma destino, la herramienta debe generar por defecto el código para todas las plataformas permitidas.

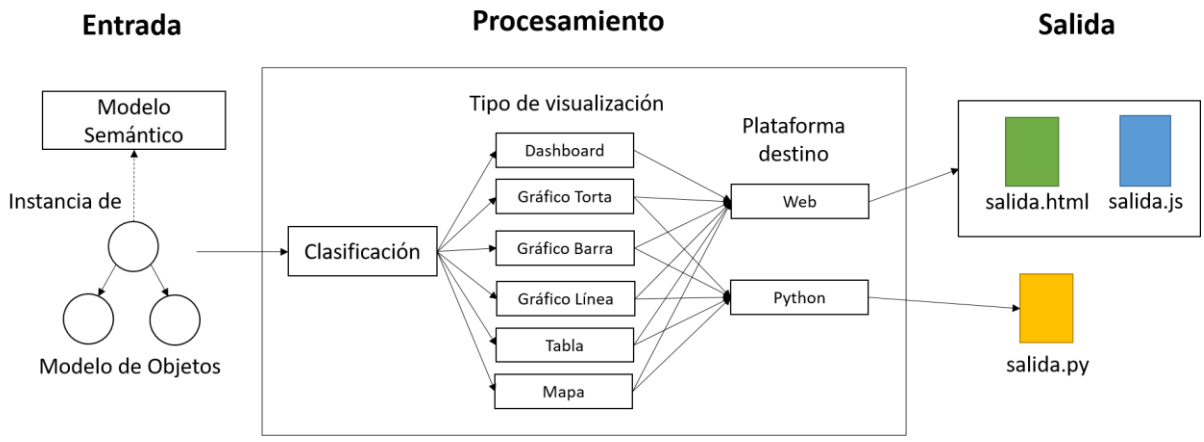
- Se debe generar un directorio con el nombre "Web" para el almacenamiento de los ficheros correspondientes a la plataforma web, en caso de que el directorio exista debe remplazar los ficheros existentes.
- Se debe generar un directorio con el nombre "js" dentro del directorio "Web" y almacenar los ficheros JavaScript correspondientes a las visualizaciones definidas por el usuario, en caso de que el directorio exista debe remplazar los ficheros existentes.
- La herramienta debe generar un directorio con el nombre "python" para el almacenamiento de los ficheros correspondientes a la plataforma Python, en caso de que el directorio exista debe remplazar los ficheros existentes.
- Para cada tipo de visualización definida, la herramienta debe procesar el conjunto de datos asociado, y el conjunto de características de la visualización.
- Para cada conjunto de datos, la herramienta debe procesar la fuente de los datos y el nombre de las variables.

Requerimientos no funcionales:

- La programación de la herramienta será realizada en el lenguaje Xtend.
- La herramienta debe poder crear y modificar ficheros en el sistema.
- Las plataformas de generación de código deben ser Web y Python.
- El código generado será almacenado en ficheros correspondientes a las estructuras de cada plataforma destino.
- Los tipos de visualizaciones del código generado deben corresponder con las visualizaciones permitidas y definidas por el usuario en el DSL.
- Los ficheros generados para la plataforma web serán archivos con extensión (.html) para el código HTML y (.js) para el código JavaScript.
- Los ficheros generados para la plataforma Python serán archivos con extensión (.py).

La figura 14 muestra la arquitectura de la herramienta generadora basada en el modelo de objetos proporcionado por Xtext.

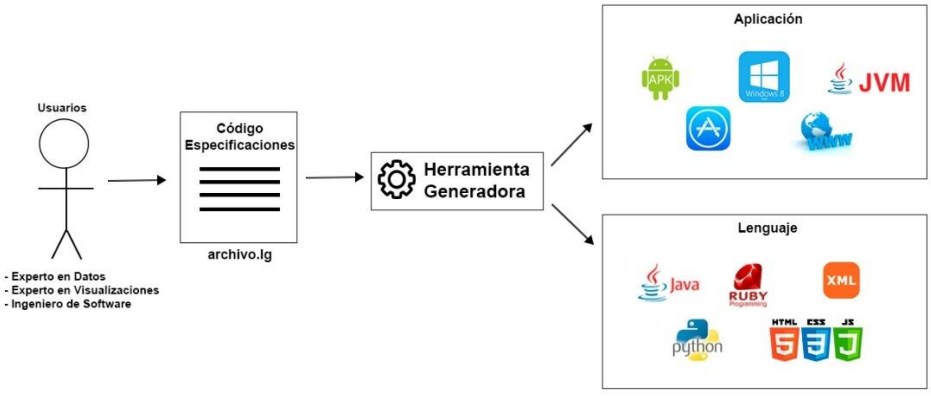
Figura 14. Arquitectura herramienta generadora



6.3 EXTENSIBILIDAD

La extensibilidad como principio de diseño de sistemas expone la capacidad de un sistema para la incorporación de nuevas características, la figura 15 representa la capacidad del framework para evolucionar y adaptarse a nuevas plataformas y lenguajes.

Figura 15. Extensibilidad Framework.



Extensión del DSL: El lenguaje de dominio específico puede evolucionar para mejorar su sintaxis, además de permitir definir más tipos de visualizaciones y manipular más características de las mismas, también pueden mejorar las

descripciones de los conjuntos de datos permitiendo representar más formatos de entrada diferentes a JSON, e incluso configurando conexiones directamente con los servidores, aceptar servicios web basados en arquitecturas diferentes a REST, incorporar funciones aplicables a los conjuntos de datos para obtener determinados resultados, permitir seleccionar la librería con la cual se debe generar el código base y otros parámetros generales de configuración, todo esto sin perder el objetivo de un DSL que es resolver un problema en un dominio específico.

Extensión Herramienta Generadora: Se pueden añadir nuevos módulos para generar código base en lenguajes como Android, Java, Ruby, C++ y demás, adicionalmente también se podría programar la herramienta de forma que genere paquetes de instalación como APK en el caso de Android. También se puede mejorar la robustez del código generado o incluso adaptarlo directamente a las de una organización en específico, al estar programada en Xtend un lenguaje de alto nivel basado en Java la integración con este lenguaje es muy buena, permitiendo añadir más funcionalidades a la plataforma.

Nuevos Roles de Usuario: La herramienta podría mejorar para incluir características útiles para el tratamiento previo de los datos, estas características estarían orientadas a un nuevo rol de usuario que no fue considerado en el diseño propuesto, lo cual haría más completo el Framework.

7. IMPLEMENTACIÓN

Como se indica en la metodología de trabajo la implementación de los diseños del DSL y la herramienta generadora de código se realizó siguiendo un modelo de prototipos evolutivos.

7.1 PROTOTIPO 1

El primer prototipo permitió realizar la primera implementación del concepto identificando componentes claves para el diseño final.

7.1.1 Implementación del lenguaje de dominio específico. Cuando se han determinado los elementos que se van a representar en el lenguaje y las reglas que aplican para estas representaciones es necesario expresar el lenguaje haciendo uso de una gramática formal, que también puede ser conocida como gramática libre de contexto, las gramáticas formales son las representaciones que definen a un lenguaje. [13]

La figura 16 contiene un ejemplo para ilustrar el funcionamiento de la gramática definida para el lenguaje, puede consultar la gramática completa en el Anexo A. Gramática Formal Prototipo 1 DSL.

Figura 16. Gramática de ejemplo simplificada DSL

1. $P \rightarrow Gráfico^{*}$
2. $P \rightarrow Tupla^{*}$
3. $P \rightarrow PP$
4. $Gráfico \rightarrow PlabraClave + Nombre + \{ + Características + \}$
5. $PlabraClave \rightarrow PieChart \mid LineChart \mid MapChart \mid BarChart \mid TableChart$
6. $Nombre \rightarrow [A - Z\ a - z\ 0 - 9][A - Z\ a - z\ 0 - 9]^{*}$
7. $Tupla \rightarrow ClaseTupla + Nombre + \{ + URL\ + Variables + \}$
8. $ClaseTupla \rightarrow Tupla1 \mid Tupla2 \mid Tupla3 \mid Tupla4$

En la gramática descrita en la figura 15 el símbolo + representa concatenación y el símbolo | representa el operador lógico OR.

Por lo tanto, haciendo uso de la gramática simplificada podemos construir representaciones de los gráficos como en la figura 17:

Figura 17. Generación de palabra con Gramática Simplificada

- P
- $Gráfico$ (Haciendo uso de la regla 1)
- $PlabraClave\ Nombre\ \{\ Características\ \}$ (Haciendo uso de la regla 4)
- $PieChart\ MiGrafico\ \{\ Características\ \}$ (Haciendo uso de las reglas 5 y 6)
- $\boldsymbol{PieChart}\ MiGrafico\ \{\ Características\ \}$ Palabra generada.

Este ejemplo muestra el funcionamiento de la gramática formal para la representación de un DSL, Xtext permite realizar descripciones gramaticales basándose en la herramienta ANTLR, la gramática completa del prototipo 1 se encuentra en el Anexo A. Gramática Formal prototipo 1 DSL.

La figura 18 muestra la descripción de un gráfico de torta y su respectivo conjunto de datos en el DSL del prototipo 1.

Figura 18. Código primera versión del DSL.

```
PieChart Torta {
      Title String título = "Titulo gráfico de torta"
      DatosGraficoTorta
}
Tuple1 DatosGraficoTorta {
        String nombre
        String valor1
}
```

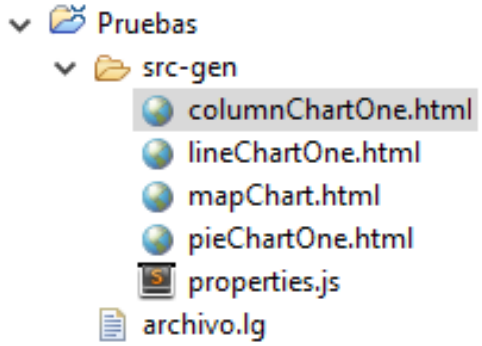
Como se puede apreciar en la figura 18, la primera versión del lenguaje cumplía el objetivo de poder describir las visualizaciones y los conjuntos de datos, sin embargo esta versión del lenguaje poseía varios problemas, la declaración de propiedades

de los gráficos era bastante extensa, poco útil y solo se podía hacer uso de la característica título, adicionalmente todos los conjuntos de datos compartían una fuente de datos genérica que no se expresaba en el lenguaje y era añadida posteriormente en la generación de código, finalmente el DSL permitía describir dos conjuntos de datos diferentes con el mismo nombre lo cual generaba errores de compresión y descripción en el lenguaje, en esta versión del lenguaje solo se podían describir 4 tipos de gráficos y el gráfico de barras recibía el nombre ColumnChart.

7.1.2 Implementación herramienta generadora. La primera versión de la herramienta generadora incluía únicamente generación de código para la plataforma web, los archivos generados eran creados todos en el directorio raíz src-gen creado por Xtext. En un archivo llamado properties.js se configuraba la fuente de datos genérica para todos los gráficos, eso generaba errores en la ejecución de algunos gráficos, esta versión de la herramienta generaba únicamente archivos HTML y el código JavaScript se encontraba declarado dentro del mismo archivo haciendo difícil la comprensión del código generado.

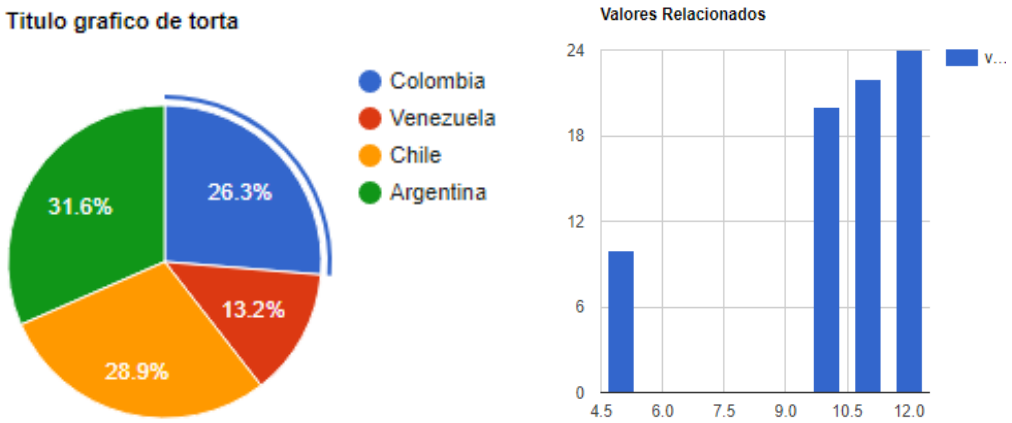
Los Dashboard declarados por el usuario no eran generados en esta versión de la herramienta, en la figura 19 se puede apreciar la estructura del proyecto y de los ficheros generados del prototipo 1.

Figura 19. Estructura proyecto y ficheros generados Prototipo 1.



En la figura 20 se puede observar un gráfico de Torta y un gráfico de barras generados con el prototipo 1.

Figura 20. Gráficos generados con el prototipo 1.



El código fuente completo de la herramienta generadora para el prototipo 1 puede ser consultado en el Anexo B. Código Fuente Herramienta Generadora Prototipo 1.

7.2 PROTOTIPO FINAL

Luego de una nueva iteración de acuerdo con las fases de desarrollo del proyecto, se implementó el prototipo final que corrigió los errores de diseño tanto del DSL como de la herramienta generadora que fueron identificados con el prototipo 1, e incluyo todas las nociones y características necesarias para cumplir con los objetivos del proyecto.

7.2.1 Actualización del lenguaje de dominio específico. La última versión del DSL permite describir todas las visualizaciones seleccionadas en el diseño con una sintaxis sencilla, adicionalmente corrige errores como la posibilidad de establecer el mismo nombre para elementos diferentes, la versión final del lenguaje incluye la noción de elementos de configuración que permiten establecer parámetros como la plataforma destino para la generación de código, en la figura 21 podemos apreciar

la sintaxis para seleccionar web como una plataforma de generación de código y la descripción de un gráfico de torta y su respectivo conjunto de datos:

Figura 21. Código versión final DSL.

```
CodeGeneratedTarget.Web

PieChart graficoTorta{
	HurtosTorta
	Title="Hurtos en Colombia según el día de la semana - 2014"
	Legend.False
	Hole.True
	CellSize.2x1

}

Tuple1 HurtosTorta{
	URL="https://www.datos.gov.co/resource/mjkh-xfbi.json?$where=dia!=%22-
%22&$select=dia,count(_2014)&$group=dia"
	String dia
	Number count_2014
}
```

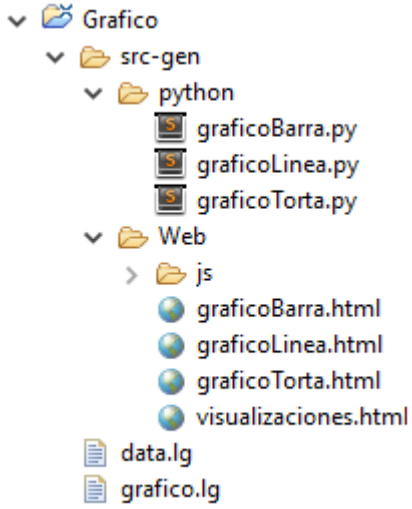
Respecto a las características del prototipo anterior cabe resaltar los siguientes cambios y actualizaciones de esta última versión en la gramática del lenguaje:

- Cada conjunto de datos requiere que se declare la URL donde se encuentra el servicio web que provee los datos.
- Las variables se declaran de acuerdo con el tipo de dato y el nombre de la variable.
- Existen dos características de configuración global: Plataforma objetivo y activar o desactivar filtros simples para el tratamiento de datos de entrada.
- Se incluyen más características visuales para los gráficos.
- La declaración de las características de un gráfico es mucho más sencilla.
- La asignación de un valor establecido por el usuario a una característica de los gráficos se hace mediante el operador "="
- La asignación de valores establecidos por el lenguaje a una característica de los gráficos se hace mediante el operador "."

Puede consultar la gramática completa del prototipo final del DSL en el Anexo C. Gramática formal prototipo final DSL.

7.2.2 Actualización herramienta generadora. En el prototipo final se genera código para las plataformas web y Python, adicionalmente se incluyen unos filtros básicos para el tratamiento de los datos de entrada, dichos filtros organizan los datos y eliminan repeticiones en los registros, esto permite utilizar la herramienta con un mayor número de entradas corrigiendo algunos errores de compatibilidad entre algunos conjuntos de datos y gráficos, adicionalmente soporta todas las características visuales nuevas definidas en esta versión del DSL, también se generan tableros de visualización para la plataforma Web únicamente, y estructura los ficheros generados de acuerdo a las especificaciones del diseño de la herramienta como se muestra en la figura 22:

Figura 22. Estructura proyecto y ficheros generados prototipo final.



En la figura 23 y 24 se puede observar un gráfico de torta y un gráfico de barras respectivamente generados para la plataforma web con el prototipo final del framework.

Figura 23. Gráfico de torta para plataforma web, prototipo final.

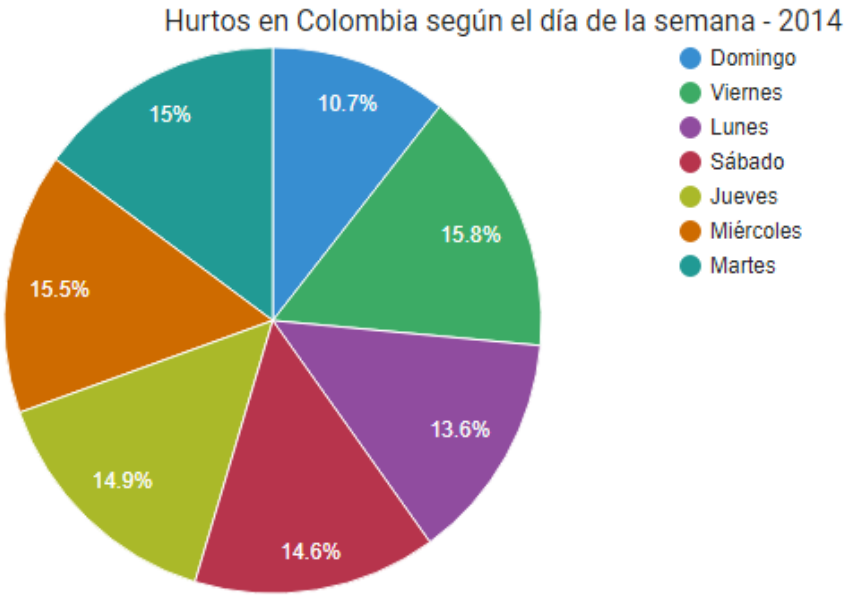
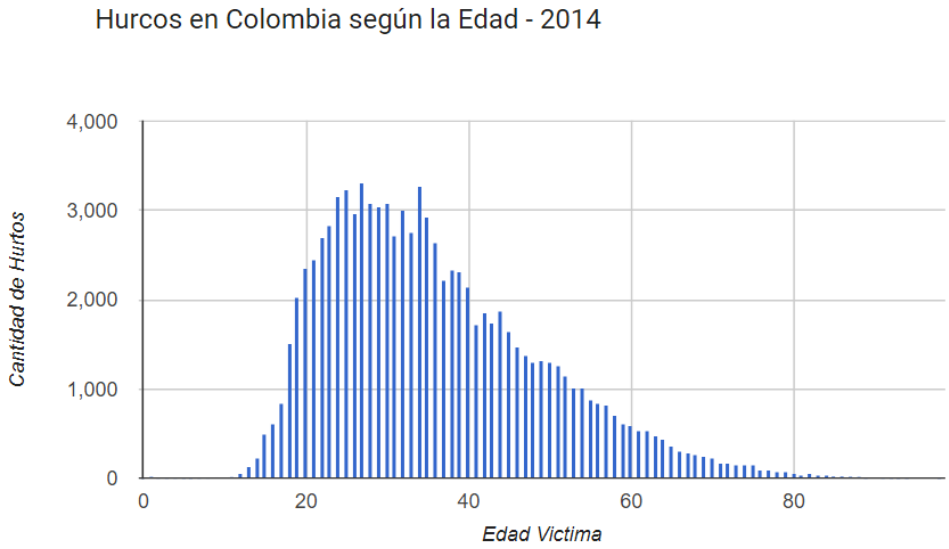
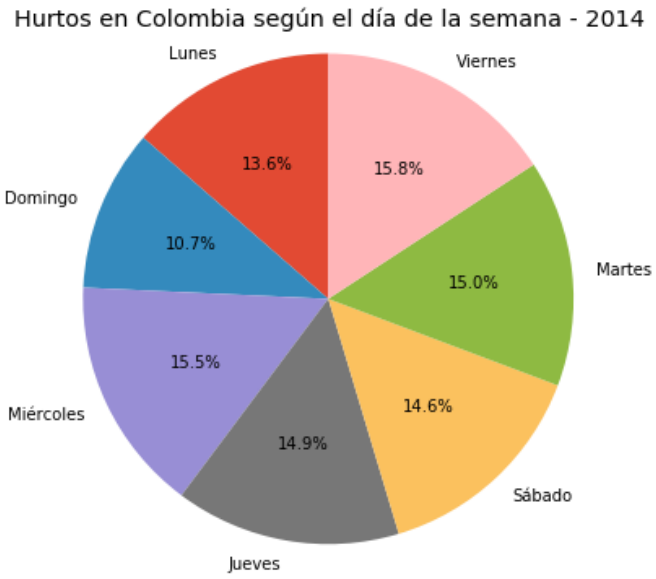


Figura 24. Gráfico de barras para plataforma web, prototipo final.



Las características definidas en el lenguaje también permiten generar los gráficos para la plataforma Python como lo muestra el grafico de torta de la figura 25.

Figura 25. Gráfico de torta para plataforma Python, prototipo final.



Puede encontrar el código fuente de la última versión de la herramienta generadora para el prototipo final en el Anexo D. Código fuente herramienta generadora prototipo Final.

8. PRUEBAS

En esta sección se van a presentar las pruebas del proyecto, en la primera parte se presentan las pruebas de validación realizadas con el fin de comprobar el correcto funcionamiento y adaptación del framework, la segunda parte se enfoca en las pruebas de usuario diseñadas para validar la reducción en la complejidad de las tareas propuestas y la optimización en el tiempo de desarrollo de las aplicaciones orientadas a la visualización de datos.

8.1 PRUEBAS DE VALIDACIÓN

Con el fin de probar el funcionamiento del framework se diseñaron tres pruebas

8.1.1 Validación de DSL. La primera prueba se enfocó en validar el comportamiento del DSL para realizar las descripciones de las visualizaciones, que incluye la descripción de los gráficos y los conjuntos de datos, los resultados de esta prueba para cada prototipo se encuentran a continuación:

Tabla 2: Resultados primera prueba de validación DSL para prototipo 1.

Elemento	**Existe una descripción valida.**	**Manipulación de Contenido**	**Manipulación de propiedades visuales.**	**Correcta relación tipo dato - variable**
Gráfico de torta.	Si	Si	No	N/A
Gráfico de barras.	Si	Si	No	N/A
Gráfico de línea.	Si	Si	No	N/A
Mapa	Si	Si	No	N/A
Tabla	No	No	No	N/A

Dashboard	Si	Si	No	N/A
Tupla 1	Si	Si	N/A	No
Tupla 2	Si	Si	N/A	No
Tupla 3	Si	Si	N/A	No
Tupla 4	No	No	N/A	No

Tabla 3: Resultados primera prueba de validación DSL para prototipo Final.

Elemento	**Existe una descripción valida.**	**Manipulación de Contenido**	**Manipulación de propiedades visuales.**	**Correcta relación tipo dato - variable**
Gráfico de torta.	Si	Si	Si	N/A
Gráfico de barras.	Si	Si	Si	N/A
Gráfico de línea.	Si	Si	Si	N/A
Mapa	Si	Si	Si	N/A
Tabla	Si	Si	Si	N/A
Dashboard	Si	Si	Si	N/A
Tupla 1	Si	Si	N/A	Si
Tupla 2	Si	Si	N/A	Si
Tupla 3	Si	Si	N/A	Si
Tupla 4	Si	Si	N/A	Si

8.1.2 Validación herramienta generadora (CASE). Las pruebas de validación de la herramienta generadora para cada uno de los prototipos se basaron en la correcta generación de código base, la generación para las plataformas seleccionadas, la implementación de las características pertenecientes a cada plataforma y la correcta ejecución y visualización de las características definidas en el código generado.

Puede consultar el listado completo de características de los gráficos en el Anexo E. Documentación, los resultados de la prueba de validación para cada prototipo se encuentran a continuación.

Tabla 4: Resultados prueba de validación herramienta generadora prototipo 1.

Elemento	**Generación**		**Ejecución/ Visualización**		**Característica Visual**					
	web.	**Python**	**web.**	**Python**	**1**	**2**	**3**	**4**	**5**	**6**
Gráfico de torta.	Si	No	Si	No	Si	No	No	No	N/A	N/A
Gráfico de barras.	Si	No	Si	No	Si	No	No	No	No	No
Gráfico de línea.	Si	No	Si	No	Si	No	No	No	No	No
Mapa	Si	No	Si	No	Si	No	No	No	No	N/A
Tabla	No	No	No	No	No	No	N/A	N/A	N/A	N/A
Dashboard	No	N/A	No	N/A	No	N/A	N/A	N/A	N/A	N/A
Tupla 1	Si	No	Si	No	N/A	N/A	N/A	N/A	N/A	N/A
Tupla 2	Si	No	Si	No	N/A	N/A	N/A	N/A	N/A	N/A
Tupla 3	Si	No	Si	No	N/A	N/A	N/A	N/A	N/A	N/A
Tupla 4	No	No	No	No	N/A	N/A	N/A	N/A	N/A	N/A

Tabla 5: Resultados prueba de validación herramienta generadora prototipo Final.

Elemento	Generación		Ejecución/ Visualización		Característica Visual					
	web.	**Python**	**web.**	**Python**	**1**	**2**	**3**	**4**	**5**	**6**
Gráfico de torta.	Si	Si	Si	Si	Si	Si	Si	Si	N/A	N/A
Gráfico de barras.	Si	Si	Si	Si	Si	Si	Si	Si	Si	Si
Gráfico de línea.	Si	Si	Si	Si	Si	Si	Si	Si	Si	Si
Mapa	Si	Si	Si	Si	Si	Si	Si	Si	Si	N/A
Tabla	Si	Si	Si	Si	Si	Si	N/A	N/A	N/A	N/A
Dashboard	Si	N/A	Si	N/A	Si	N/A	N/A	N/A	N/A	N/A
Tupla 1	Si	Si	Si	Si	N/A	N/A	N/A	N/A	N/A	N/A
Tupla 2	Si	Si	Si	Si	N/A	N/A	N/A	N/A	N/A	N/A
Tupla 3	Si	Si	Si	Si	N/A	N/A	N/A	N/A	N/A	N/A
Tupla 4	Si	Si	Si	Si	N/A	N/A	N/A	N/A	N/A	N/A

8.1.3 Validación conexión con fuentes de datos reales. Las pruebas de validación de conexión buscaron medir la capacidad de trabajo entre las fuentes de datos reales y la herramienta, para esta tarea seleccionamos dos plataformas de fuentes de datos:

- Datos abiertos Colombia. [9]
- NYC Open data. [14]

Estas plataformas cuentan con servicios web que dan acceso a un gran número de datos libres que permiten hacer las pruebas, para las cuales se seleccionaron

aproximadamente 20 conjuntos de datos y se estableció un porcentaje de acuerdo con el número de veces que fue posible realizar las acciones descritas, los resultados se encuentran en la Tabla 6:

Tabla 6: Resultados pruebas de conexión con datos reales.

Fuente de datos	**Describir el conjunto de datos en el DSL**	**Establecer conexión con el Servicio web**	**Acceder a los datos desde el código generado.**	**Visualizar correctamente los datos obtenidos**
Datos Abiertos Colombia	100%	100%	80%	80%
NYC Open Data.	100%	100%	80%	80%

La descripción en el DSL se realizó satisfactoriamente para todos los conjuntos de datos seleccionados e igualmente se pudo establecer conexión con el servicio web, en algunos casos el servicio web retorno conjuntos de datos que no respetaban el formato estándar o que tenía registros con diferente número de variables por registro, esto implico que en aproximadamente el 20% de los casos de las pruebas hechas, el código generado no pudo acceder a los datos o los datos no fueron visualizados de forma correcta, en estos casos haciendo uso de la API proporcionada por las fuentes de datos se pudo realizaron un tratamiento previo de los datos de lado del servidor de forma que pudieran ser accedidos por el código generado.

8.2 PRUEBAS DE USUARIO

Uno de los objetivos del proyecto era validar que el uso framework representaba una reducción en la complejidad y el tiempo de desarrollo de aplicaciones orientadas a la visualización de datos, es por esto que fue necesario desarrollar una serie de pruebas de usuario para verificar si la herramienta cumplía con el objetivo propuesto.

Trabajar con expertos en visualizaciones e ingenieros de software para validar el objetivo del proyecto no fue posible debido a que las personas con estos roles tienen una agenda relativamente apretada y conseguir una cita para realizar las pruebas era tedioso y sumamente demorado, además de no contar con un número suficiente de contactos con estos perfiles para realizar más de dos pruebas, por lo que se seleccionó estudiantes o profesionales del área de sistemas como usuarios ya que es un perfil mucho más accesible.

Las pruebas planteadas están divididas en dos fases, cada una de las cuales tiene como objetivo obtener los mismos resultados mediante marcos de trabajo distintos.

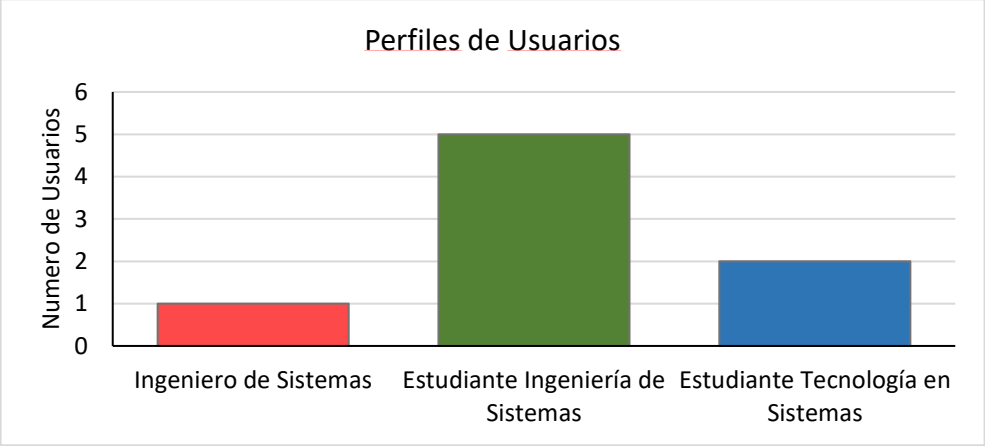
- **Fase 1:** en esta fase el usuario busca desarrollar el código para un conjunto de cuatro visualizaciones, dos de ellas tanto en plataforma web como en Python y las últimas dos visualizaciones únicamente en la plataforma web, para realizar esta fase el usuario tiene un máximo de 60 minutos.
- **Fase 2:** en la segunda fase el usuario describe el conjunto de visualizaciones de la fase anterior, haciendo uso del lenguaje de dominio específico y genera el código base de las visualizaciones, el usuario tiene un máximo de 60 minutos para realizar esta tarea.

Teniendo en cuenta que cada fase requería de una introducción previa al usuario, que se realizó un perfil inicial, y al finalizar la prueba el usuario realizo un cuestionario para evaluar su experiencia en las dos fases, las pruebas de usuario tuvieron una duración aproximada de dos horas y media cada una, y se realizaron 8 pruebas de usuario en aproximadamente 20 horas, el documento completo de las pruebas de usuario se encuentra en el Anexo F. pruebas de usuario.

8.2.1 Análisis resultados pruebas de Usuario. En este apartado presentaremos los resultados obtenidos con la aplicación de la prueba de usuario.

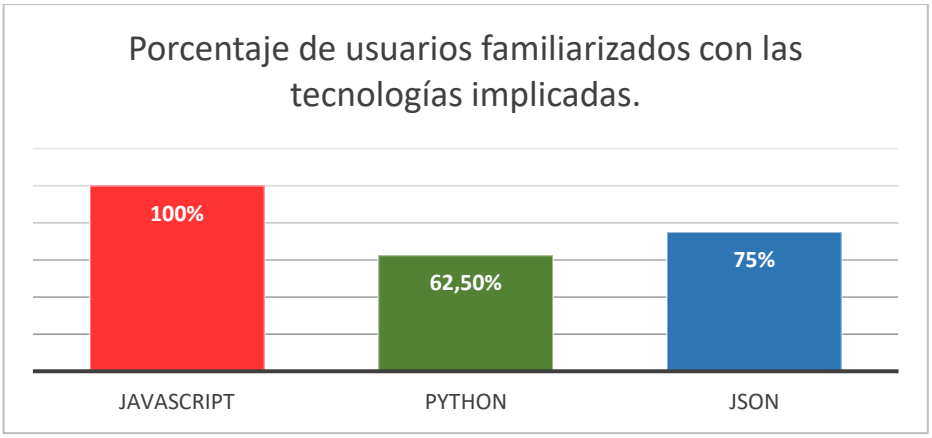
8.2.1.1 Perfiles de Usuarios. Los usuarios que realizaron la prueba en su mayoría eran estudiantes de carreras técnicas o profesionales relacionadas con el área de sistemas, en la figura 26 se puede apreciar los perfiles de los usuarios que realizaron la prueba.

Figura 26. Gráfico perfiles de usuario.



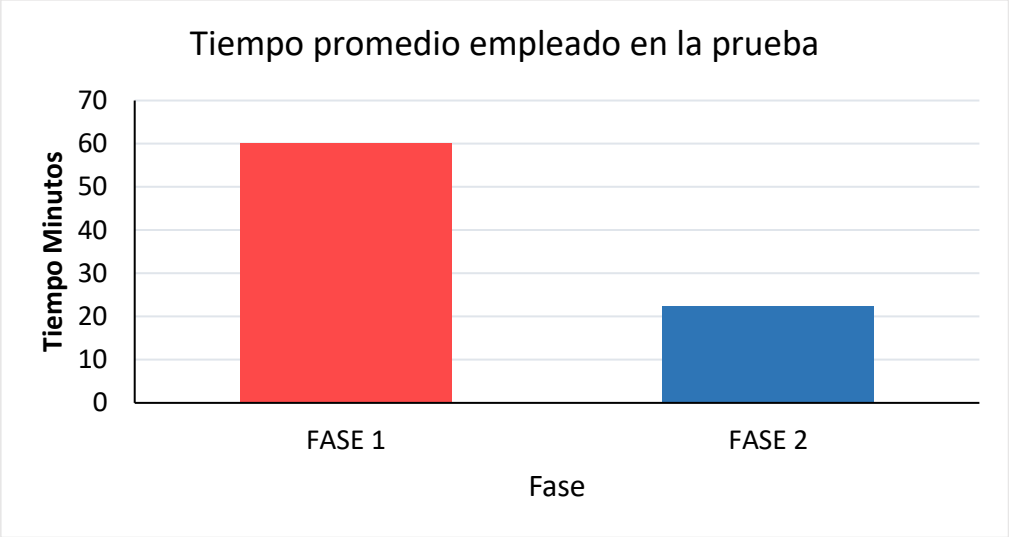
La figura 27 representa el porcentaje de usuarios de la prueba que habían trabajado previamente con los lenguajes JavaScript, Python y que conocían el formato JSON.

Figura 27. Porcentaje de usuarios familiarizado con el uso de tecnologías implicadas en la prueba.



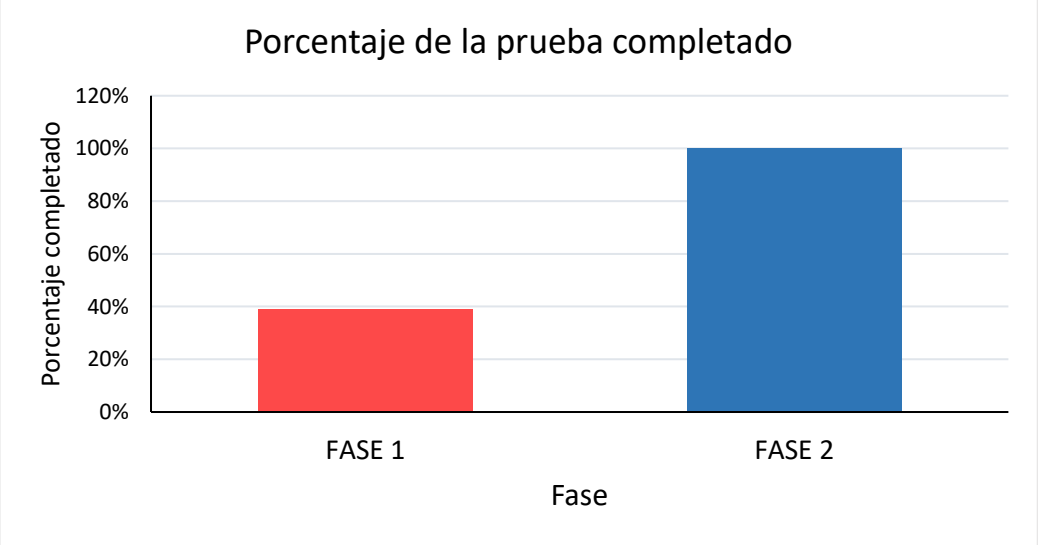
8.2.1.2 Tiempo promedio empleado en cada fase de la prueba. La figura 28 muestra el tiempo promedio empleado por los usuarios durante cada fase de la prueba.

Figura 28. Tiempo promedio empleado por fase en las pruebas de usuario.



8.2.1.3 Porcentaje promedio completado de la prueba. La figura 29 muestra el porcentaje promedio de la prueba realizado por los usuarios durante cada fase.

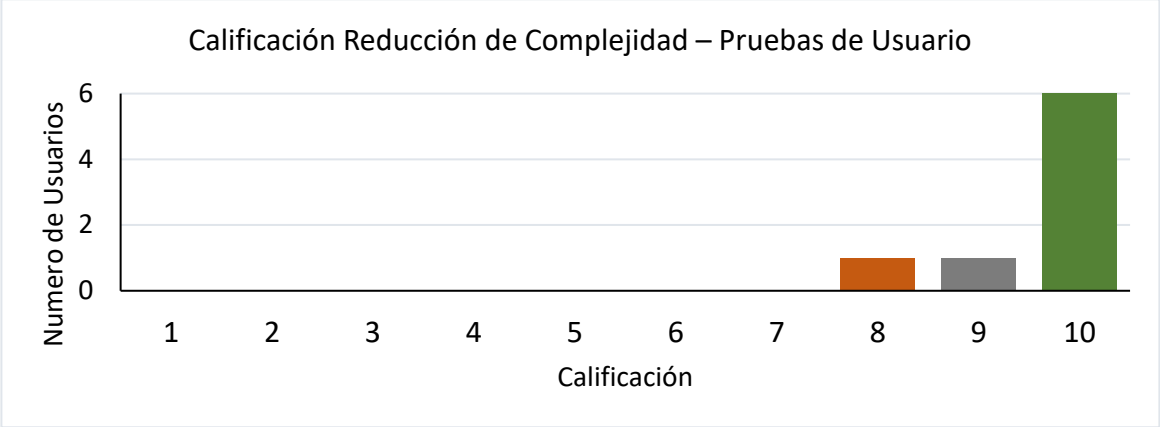
Figura 29. Porcentaje de la prueba completado por fases.



8.2.1.4 Resultado formulario experiencia de usuario. A continuación, se presentan los resultados del formulario en el cual el usuario calificó su experiencia con la herramienta.

Disminución de complejidad: La figura 30 muestra la respuesta a la pregunta ¿Qué tanto disminuyo la complejidad de la tarea haciendo el uso del framework? asignando una calificación de 1 a 10, donde 1 es la calificación mínima.

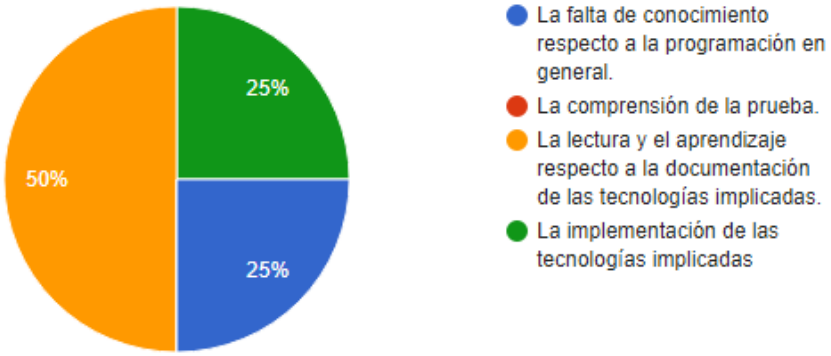
Figura 30. Reducción de complejidad de tarea con el uso del framework.



Uso de la herramienta: 100% de los usuarios respondieron que usarían la herramienta para generar el código base y luego adaptarlo a sus necesidades antes que realizar dicho código de la forma tradicional o mediante el uso de otras herramientas.

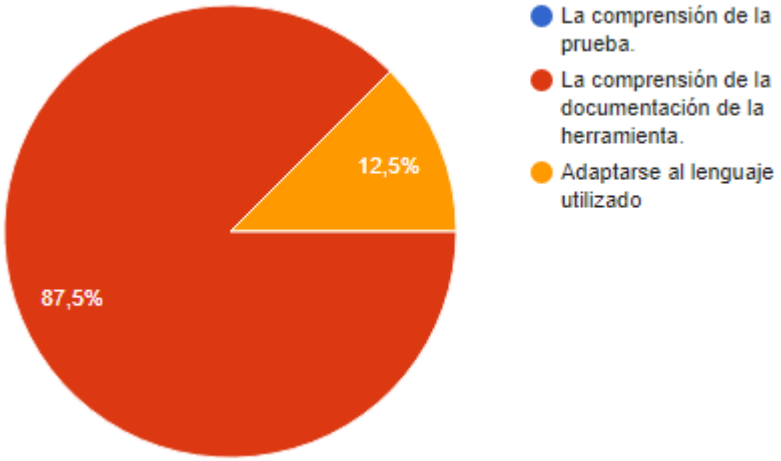
Retraso primera fase: Las dificultades en la primera fase se ven en la figura 31.

Figura 31. Dificultades en la realización de la fase 1.



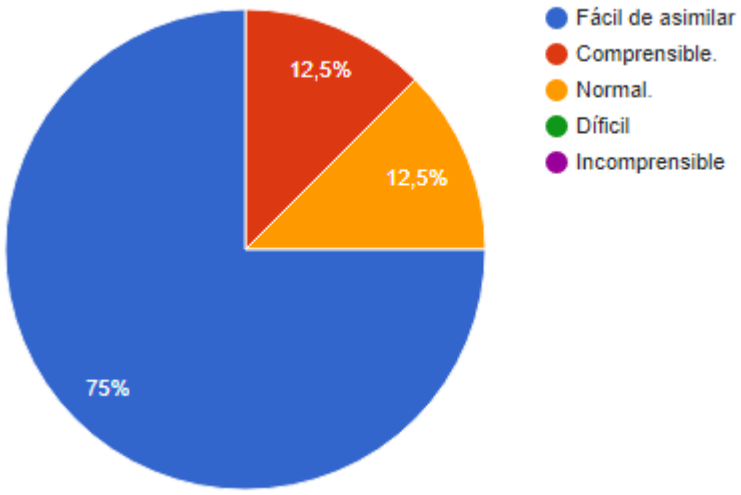
Retraso segunda fase: las razones por las cuales los usuarios tuvieron dificultades en la segunda fase se muestran en la figura 32.

Figura 32. Dificultades en la realización de la fase 2.



Calificación comprensión de la herramienta: la figura 33 muestra la calificación asignada por los usuarios a la facilidad de comprensión del lenguaje y de la herramienta en general:

Figura 33. Calificación comprensión de la herramienta.



9. CONCLUSIONES

- El incremento de dispositivos conectados a internet y el aumento del volumen de datos generados en el mundo ha creado una demanda progresiva de personas capaces de interpretar y transformar los datos en información, surgiendo nuevos roles como expertos en visualizaciones de datos, que trabajan de la mano con ingenieros de software para dar respuesta a una creciente demanda de aplicaciones orientadas a la visualización de datos, en consecuencia el framework desarrollado brinda una plataforma para facilitar la interacción entre estos roles reduciendo la complejidad y el tiempo de desarrollo de las aplicaciones e integrando diferentes plataformas y herramientas en el proceso.
- La metodología de trabajo permitió identificar e implementar las características del framework de forma que los objetivos planteados en el proyecto se cumplieron.
- El lenguaje diseñado permite describir las visualizaciones seleccionadas con un alto nivel de abstracción, por lo tanto, cumplió con las expectativas del proyecto.
- La herramienta CASE interpreta un modelo de las descripciones realizadas por el usuario en el DSL y genera código para diferentes plataformas, por lo tanto, cumplió con los requerimientos del proyecto además de realizar la integración de diferentes tecnologías web como Google Chart, JQuery y tecnologías Python como Matplotlib lo cual facilita el trabajo del desarrollador.
- El prototipo final del framework cumplió con las expectativas e incluso agrego más funcionalidades de las pensadas en el diseño original, además de generar satisfactoriamente código base para dos plataformas objetivo.
- Los resultados de las pruebas de validación fueron satisfactorios, reflejando que el prototipo final del framework cumple con las especificaciones del diseño realizado.

- Los resultados de las pruebas de usuario validan la capacidad del framework para reducir la complejidad y el tiempo en el desarrollo de aplicaciones orientadas a la visualización de datos, permitiendo llevar a cabo un mayor porcentaje del desarrollo en menos tiempo.
- Este proyecto refleja la importancia de asignaturas como autómatas y lenguajes formales y la asignatura electiva de compiladores, que permiten complementar la formación de profesionales del área de sistemas para comprender mejor el funcionamiento de los lenguajes de programación.
- La integración de profesionales del área de sistemas con los profesionales de otras áreas es una tarea necesaria para casi cualquier desarrollo ya que cualquier sistema creado tiene un enfoque que requiere de la asesoría de un experto en el campo. El desarrollo y uso de herramientas como las planteadas en el proyecto actual mejora la interacción entre ingenieros de sistemas y expertos en campos específicos.
- Las visualizaciones seleccionadas para el desarrollo del proyecto son las más usadas y clásicas, aunque de acuerdo con la complejidad de un conjunto de datos se requiere de visualizaciones más complejas y completas para revelar la información deseada.

10. RECOMENDACIONES

- Al trabaja en el diseño de lenguajes de dominio específico es muy importante tener claro los contenidos de la asignatura autómatas y lenguajes formales y en especial entender el concepto de gramáticas libres de contexto, ya que esto permite realizar un mejor diseño e implementación del lenguaje.
- Xtext es una excelente herramienta para diseño de lenguajes que se adapta no solo al entorno de Eclipse, también se puede utilizar en entornos como Android Studio y existe la opción de crear un editor web para nuestro lenguaje.
- Debido al alcance del proyecto el DSL fue diseñado para cumplir un objetivo específico, aunque el lenguaje diseñado podría evolucionar para un mayor número de necesidades y proporcionar más control sobre el resultado a los usuarios del framework sin abandonar su dominio, lo cual podría convertirse en parte o enfoque de la continuación del proyecto de investigación actual.
- La herramienta CASE programada para la generación de código es una herramienta muy básica pero que cumple con su función, valdría la pena trabajar para incluir más funcionalidades y módulos de generación de código como se explica en el apartado de extensibilidad en una futura continuación de la investigación.

REFERENCIAS BIBLIOGRAFICAS

1. DIARIO LA EXPASIÓN - ESPAÑA. Internet en un minúto [En línea]. (Recuperado 14 de Noviembre de 2016) Disponible en http://www.expansion.com/economia-digital/2016/04/21/5718a79a22601d8d028b4647.html
2. BASS, Len. CLEMENTS, Paul. KAZMAN, Rick. Software architecture in practice. 3rd ed. Boston: Addison-Wesley, 2012. p.18. ISBN - 13: 978-0321815736
3. RODENAS, Ferran. QUERALT, Anna. Metodologies de Desenvolupament Dirigides per Models [En línea]. (Recuperado 6 de Mayo de 2017) Disponible en: http://cv.uoc.edu/webapps/dspace_rei/bitstream/10609/547/1/42181tfc.pdf.
4. BRAMBILLA, Marco. CABOT, Jordi. WIMMER, Manuel. Model-driven Software Engineering in Practice - 2nd ed. Morgan & Claypool publishers. 2017. p. 85. ISBN: 9781627057080
5. BETTINI, Lorenzo. Implementing Domain-Specific Languages with Xtext and Xtend, Practice - 2nd ed, 2016. ISBN 978-1-78646-496-5
6. THE ECLIPSE FOUNDATION. Eclipse Modeling Framework [En línea]. (Recuperado 15 de Junio de 2016) Disponible en: http://www.eclipse.org/modeling/emf/.
7. PASSERINI, Nicolás. FERNANDES, Javier. LORENZANO, Esteban.Programación Avanzada con Objetos [En línea]. (Recuperado 18 de Junio de 2016) Disponible en: https://sites.google.com/site/programacionhm/conceptos/dsls/domainspecificlanguage/dsl---xtext.
8. HAUSENBLAS, Michael. OpenData [En línea]. (Recuperado 25 de Junio de 2016) Disponible en: http://5stardata.info/en/.
9. VIVE DIGIGAL. Datos Abiertos Colombia [En línea]. (Recuperado 20 de Junio de 2016) Disponible en: https://www.datos.gov.co/.
10. SOCRATA, Plataforma de datos Abiertos. [En línea]. (Recuperado 20 de Junio de 2016) Disponible en: https://socrata.com/what-works-cities/.
11. PLOTLY, Plotly [En línea]. (Recuperado 15 de Mayo de 2017) Disponible en: https://plot.ly/.
12. TABLEAU SOFTWARE. Tab|eau. [En línea] (Recuperado 15 de Mayo de 2017) Disponible en: https://www.tableau.com/es-es
13. HOPCROFT, John. MOTWANI, Rajeev. ULLMAN, Jeffrey. Teoría de autómatas, lenguajes y computación. Madrid: PEARSON EDUCACIÓN S.A., 2007. p. 144 ISBN: 978-84-7829-088-8
14. CITY OF NEW YORK. NYC Open Data [En línea]. (Recuperado 20 de Junio de 2017) Disponible en: https://opendata.cityofnewyork.us/

BIBLIOGRAFIA

BASS, Len. CLEMENTS, Paul. KAZMAN, Rick. Software architecture in practice. 3rd ed. Boston: Addison-Wesley, 2012. p.18. ISBN - 13: 978-0321815736

BETTINI, Lorenzo. Implementing Domain-Specific Languages with Xtext and Xtend, Practice - 2nd ed, 2016. ISBN 978-1-78646-496-5

BRAMBILLA, Marco. CABOT, Jordi. WIMMER, Manuel.Model-driven Software Engineering in Practice - 2nd ed. Morgan & Claypool publishers. 2017. p. 85. ISBN: 9781627057080

CITY OF NEW YORK. NYC Open Data [En línea]. (Recuperado 20 de Junio de 2017) Disponible en: https://opendata.cityofnewyork.us/

DIARIO LA EXPASIÓN - ESPAÑA. Internet en un minúto [En línea]. (Recuperado 14 de Noviembre de 2016) Disponible en http://www.expansion.com/economia-digital/2016/04/21/5718a79a22601d8d028b4647.html

HAUSENBLAS, Michael. OpenData [En línea]. (Recuperado 25 de Junio de 2016) Disponible en: http://5stardata.info/en/.

HOPCROFT, John. MOTWANI, Rajeev. ULLMAN, Jeffrey. Teoría de autómatas, lenguajes y computación. Madrid: PEARSON EDUCACIÓN S.A., 2007. p. 144 ISBN: 978-84-7829-088-8

PLOTLY, Plotly [En línea]. (Recuperado 15 de Mayo de 2017) Disponible en: https://plot.ly/.

RODENAS, Ferran. QUERALT, Anna. Metodologies de Desenvolupament Dirigides per Models [En línea]. (Recuperado 6 de Mayo de 2017) Disponible en: http://cv.uoc.edu/webapps/dspace_rei/bitstream/10609/547/1/42181tfc.pdf.

SOCRATA, Plataforma de datos Abiertos. [En línea]. (Recuperado 20 de Junio de 2016) Disponible en: https://socrata.com/what-works-cities/.

TABLEAU SOFTWARE. Tab|eau. [En línea] (Recuperado 15 de Mayo de 2017) Disponible en: https://www.tableau.com/es-es

THE ECLIPSE FOUNDATION. Eclipse Modeling Framework [En línea]. (Recuperado 15 de Junio de 2016) Disponible en: http://www.eclipse.org/modeling/emf/.

VIVE DIGIGAL. Datos Abiertos Colombia [En línea]. (Recuperado 20 de Junio de 2016) Disponible en: https://www.datos.gov.co/.