SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

Subsecretaría de Educación Superior
Tecnológico Nacional de México
Instituto Tecnológico de Orizaba

Instituto Tecnológico de Orizaba
División de Estudios de Posgrado e Investigación
Doctorado en Ciencias de la Ingeniería

“Desarrollo de una plataforma de software para la generación de código para aplicaciones multidispositivo a partir del procesamiento de imágenes”

TESIS

Que para obtener el grado de Doctor en:
Ciencias de la Ingeniería

PRESENTA:

M.S.C. Viviana Yarel Rosales Morales D04011109

DIRECTOR DE TESIS:

Dr. Giner Alor Hernández.

Orizaba, Veracruz, México. Mayo de 2017

Contenido

Lista de figuras

Lista de tablas

Resumen

El desarrollo automático de aplicaciones es un área de investigación importante dentro de la Ingeniería de Software en las ciencias de la computación. Actualmente, las propuestas de desarrollo automático de aplicaciones tienen ciertas desventajas tales como la complejidad en el proceso de desarrollo de aplicaciones, elevado tiempo de respuesta y tiempo empleado en el proyecto (en todas sus fases), entre otros. Además de todo lo anterior, las plataformas de software y el tipo de dispositivos móviles soportados son limitados. Para abordar estos problemas, es necesario desarrollar nuevos enfoques que permitan una rápida y más intuitiva generación de aplicaciones Web y móviles multiplataforma y multidispositivo.

Este trabajo de tesis doctoral propone una solución que consiste en el desarrollo e integración de un conjunto de métodos computacionales que permitan la generación automática de software a partir de técnicas de procesamiento de imágenes y reconocimiento de patrones de diseño de interfaz de usuario, usando un nuevo enfoque para el proceso de desarrollo de interfaces gráficas de usuario.

En este sentido, el impacto de este trabajo reside en 3 aspectos fundamentales: 1) el desarrollo de un conjunto de nuevos métodos computacionales para la generación de software, 2) proveer un nuevo enfoque en el reconocimiento elementos de interfaces gráficas mediante su representación en patrones de diseño de interfaces de usuario con una aplicación en el área de la ingeniería de software, 3) la aplicación de la generación de este conjunto de métodos computacionales en el área del desarrollo de software multidispositivo y multiplataforma a través de una herramienta generadora de código.

Abstract

Automatic application development is an important research area of software engineering. However, current automatic application development initiatives present certain disadvantages, such as a complex and time-consuming development process and high response time; moreover, they support a limited number of software platforms and mobile devices. To address these issues, it is necessary to develop new approaches to a faster and more intuitive generation of multi-device and multi-platform Web and mobile applications. From this perspective, this doctoral dissertation proposes a set of computational methods for automatic software generation from image processing techniques and recognition of user interface design patterns (UIDPs), thus exploiting a new approach to the process of graphical user interface development. The impact of this work can be perceived through its three major contributions: 1) it proposes a set of new computational methods for software generation, 2) it offers a new approach to the recognition of graphical interface elements through their UIDPs representations with an application, 3) it presents a solution to develop multi-device and multi-platform software through a code-generating tool.

Introducción

Dentro de la ingeniería de software existe un área de investigación muy importante que es el desarrollo automático de aplicaciones o de software. A pesar de la gran variedad de propuestas para el desarrollo automático de aplicaciones Web y móviles, todas ellas tienen ciertas desventajas, como la complejidad en el proceso de desarrollo de aplicaciones, el tiempo de respuesta y el tiempo empleado en el proyecto (diseño, implementación, etc.), entre otros. Además, el desarrollo de aplicaciones multiplataforma es una tarea compleja. Para abordar estas cuestiones, es necesario desarrollar un nuevo enfoque que permita una rápida generación de aplicaciones Web y móviles multiplataforma y reduzca la brecha de aprendizaje de un programador. En otras palabras, se requiere una herramienta eficiente y fácil de usar para la generación de aplicaciones para dispositivos móviles multiplataforma adecuada para programadores principiantes.

La principal contribución de este trabajo es, por lo tanto, un enfoque nuevo, ágil e intuitivo para desarrollar aplicaciones para dispositivos móviles multiplataforma que cumplan los criterios antes mencionados para un desarrollo eficaz de aplicaciones web y móviles multiplataforma. Este enfoque está integrado en la herramienta ImagIng y es particularmente adecuado para usuarios no experimentados. Este enfoque se basa en técnicas de procesamiento de imágenes para reconocer patrones de diseño de interfaz de usuario (PDIUs) dentro de interfaces de usuario, y a partir de estos patrones generar el código fuente para aplicaciones multiplataforma y multidispositivo.

La herramienta ImagIng es capaz de desarrollar aplicaciones multiplataforma para Web, Android™, Firefox® OS, iOS y Windows Phone® a partir de una imagen de entrada proporcionada por el usuario. La imagen debe cumplir con ciertas características y debe representar la interfaz a generar. Las imágenes pueden ser ADVs (*Abstract Data View*), o dibujos a mano alzada que representan el PDIU que se generará, el cual, a través de la herramienta ImagIn, será codificado en un lenguaje apropiado dependiendo de la plataforma elegida para el despliegue.

Este documento está estructurado en cinco capítulos, estos describen los aspectos fundamentales de la tesis, los hallazgos y los resultados. El capítulo 1 presenta los antecedentes del proyecto, se presenta la revisión de los fundamentos teóricos que sirven como base conceptual del proyecto de tesis doctoral y que permiten comprender el contenido de la misma. Además se presentan también el objetivo general y los específicos del proyecto de investigación, así como la hipótesis, el planteamiento del problema y la justificación.

Por su parte el capítulo 2 presenta una revisión del estado del arte, en la cual se discuten brevemente algunos de los trabajos más relevantes y que se encuentran relacionados con el presente trabajo de tesis. El estado del arte se encuentra dividido en dos secciones: 1) Enfoques de desarrollo de software, y 2) Herramientas para desarrollo de aplicaciones.

El capítulo 3 describe la metodología utilizada para el desarrollo de este trabajo de tesis, así como los resultados de la aplicación de esta metodología, incluyendo la arquitectura de la herramienta, el proceso de generación de aplicaciones, el flujo de trabajo, por mencionar algunos.

En el capítulo 4 se presentan los resultados del proyecto de tesis, que incluyen: 3 casos de estudio, 1) se muestra la generación de código para múltiples plataformas a partir de una imagen ideal, 2) se muestra la generación de código para una aplicación Web a partir de una imagen no ideal, y 3) se muestra la generación de código para aplicaciones para dispositivos móviles a partir de imágenes no ideales. Además también se presenta la evaluación a la herramienta ImagIng.

Finalmente, en el capítulo 5 se presentan las conclusiones y el trabajo a futuro.

Capítulo 1: Antecedentes

En este capítulo se presentan los antecedentes de investigación de la tesis doctoral, dentro de los que se encuentra un conjunto de conceptos relacionados con el tema de tesis, además se describen los objetivos de la investigación, la hipótesis, el planteamiento del problema, la justificación y las aportaciones.

1.1.Marco teórico

A continuación se presentan algunos conceptos relacionados con el tema de tesis los cuales son esenciales para un mejor entendimiento a cerca de lo planteado en esta tesis tales como los algoritmos para el procesamiento de imágenes además de algunas tecnologías y enfoques de desarrollo de software.

1.1.1. Ingeniería de software

Según la definición del IEEE, "software es la suma total de los programas de computadora, procedimientos, reglas, la documentación asociada y los datos que pertenecen a un sistema de cómputo" y "un producto de software es un producto diseñado para un usuario". En este contexto, la Ingeniería de Software (SE del inglés "*Software Engineering*") es un enfoque sistemático, disciplinado y cuantificable del desarrollo, operación, mantenimiento y retiro del software (*IEEE Standard Glossary of Software Engineering Terminology*, 1990).

Ingeniería de software es la aplicación práctica del conocimiento científico al diseño y construcción de programas de computadora y a la documentación asociada requerida para desarrollar, operar y mantenerlos. Se conoce también como desarrollo de software o producción de software (Boehm, 1976).

La ingeniería de software trata del establecimiento de los principios y métodos de la ingeniería a fin de obtener software de modo rentable, que sea fiable y trabaje en máquinas reales (Bauer, 1975).

1.1.2. Enfoques de desarrollo de software

La ingeniería de software es un área importante dentro del desarrollo de software, además existen diversos enfoques y metodologías para el desarrollo de software, a continuación se describen los más relevantes.

MDA: Model-Driven Architecture, es un enfoque para el desarrollo de software que está habilitado por las especificaciones OMG (Object Management Group) existentes, tales como el Lenguaje de Modelado Unificado (UML), el *Meta Object Facility* (MOF) y el *Common Warehouse Metamodel* (CWM). Un aspecto fundamental de MDA es su capacidad para abordar el ciclo completo de desarrollo, que abarca el análisis y el diseño, la programación, las pruebas, el ensamblaje de componentes, así como el despliegue y el mantenimiento (Truyen, 2006).

MDD: Model-Driven Development, es una alternativa a la ingeniería *round-trip*. Ingeniería *round-trip* es el concepto de ser capaz de hacer cualquier tipo de cambio a un modelo, así como al código generado a partir de ese modelo. Los cambios siempre se propagan bidireccionalmente y ambos artefactos son siempre consistentes (Völter et al., 2013).

MDE: Model-Driven Engineering, es una metodología de desarrollo de software que se centra en la creación y explotación de modelos de dominio, que son modelos conceptuales de todos los temas relacionados con un problema específico. Por lo tanto, destaca y apunta a representaciones abstractas del conocimiento y las actividades que rigen un dominio de aplicación particular (Schmidt, 2006).

Para comprender mejor como se relacionan MDA, MDD y MDE, se muestra la figura 1.1, en la cual se observa claramente la interrelación entre los enfoques mencionados.

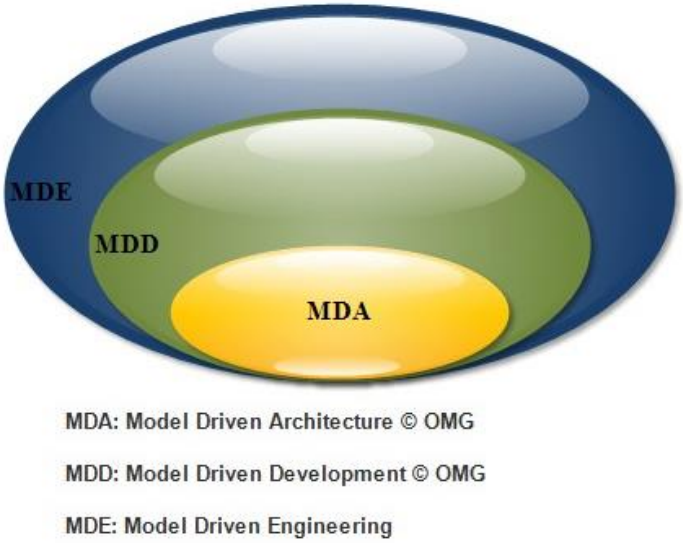


Figura 1.1. Representación gráfica de los enfoques de desarrollo de software MDA, MDD y MDE

FDD: Feature-Driven Development, es un proceso de desarrollo de software iterativo e incremental. Es uno de varios métodos ágiles para desarrollar software. FDD combina una serie de mejores prácticas reconocidas por la industria en un todo cohesivo. Todas estas prácticas se basan en una perspectiva de funcionalidad (*feature*) valorada por el cliente (Coad, Lefebvre and De Luca, 1999; Palmer and Felsing, 2002).

En la figura 1.2 se muestra el ciclo de vida de FDD.

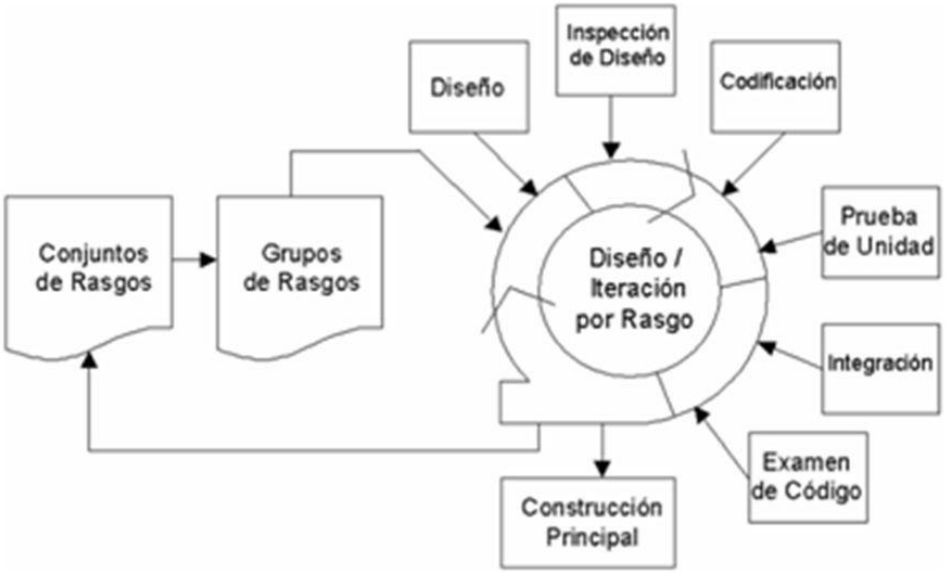


Figura 1.2. Ciclo de vida de FDD

TDD: Test-Driven Development, es un proceso de desarrollo de software que se basa en la repetición de un ciclo de desarrollo muy corto: los requisitos se convierten en casos de prueba muy específicos, entonces el software se mejora para pasar las nuevas pruebas. Kent Beck (2012), a quien se atribuye haber desarrollado o "redescubierto" la técnica, declaró en 2003 que TDD alienta diseños simples e inspira confianza (Beck, 2003).
En la figura 1.3 se muestra el ciclo de vida de TDD.

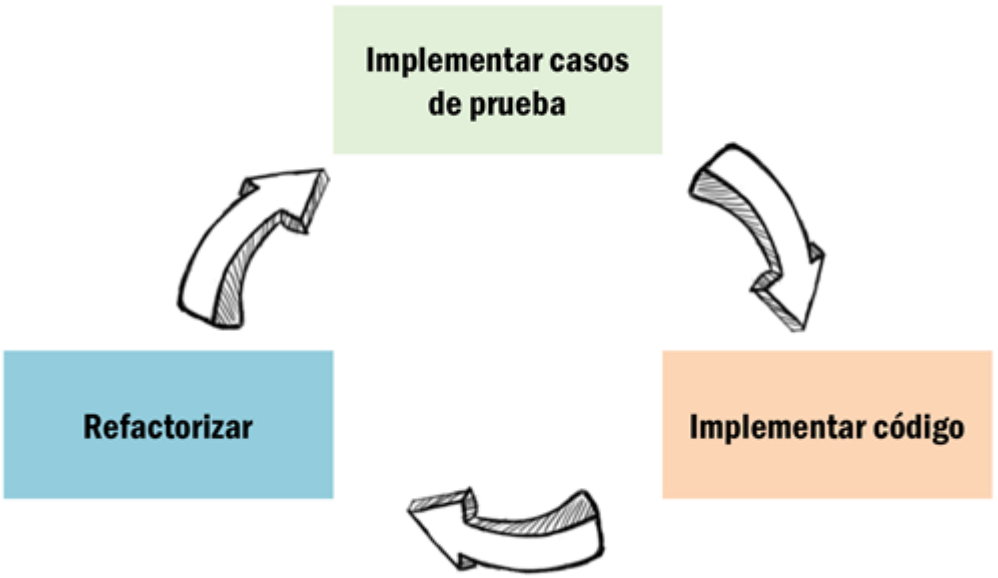


Figura 1.3. Ciclo de vida de TDD

PBD: Performance-Based Design, es un paradigma de diseño basado en el desempeño, en lugar de los paradigmas prevalecientes basados en procesos. Sugiere que la fuerza impulsora detrás de cualquier actividad de diseño es el deseo de lograr una solución cualitativa para una combinación particular de forma y función en un contexto específico (Kalay, 1999).

PPRD: Phases Process for RIAs Development, es un proceso para construir RIAs que se centra sólo en las actividades principales para transformar los requerimientos del usuario en un producto de software (Colombo-Mendoza, Alor-Hernández, Rodríguez-González and Colomo-Palacios, 2013; Alor-Hernández, Rosales-Morales, and Colombo-Mendoza, 2014).
En la figura 1.4 se muestran las fases de PPRD.

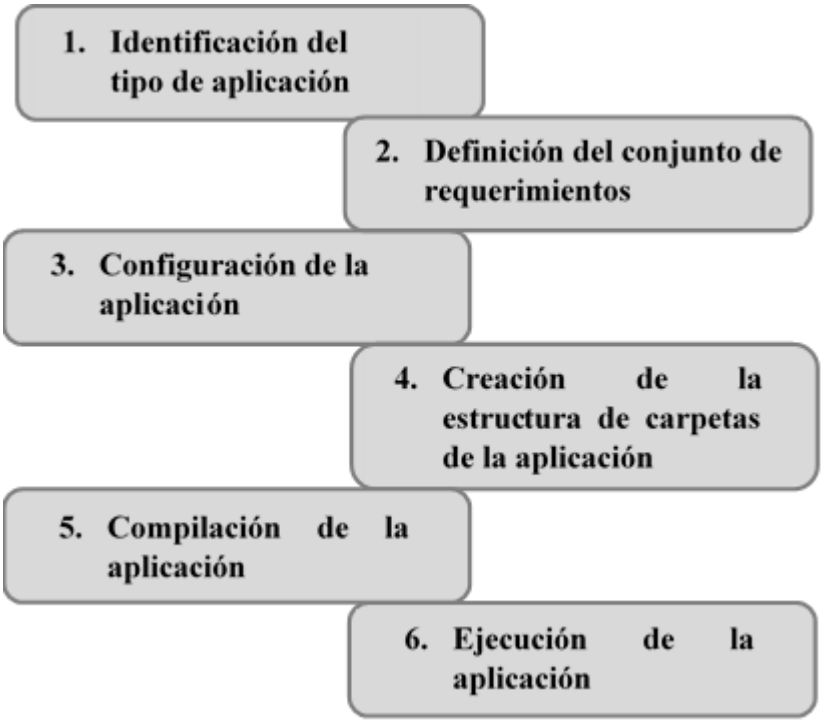


Figura 1.4. Fases de PPRD

D3: Design-Driven Development, es un proceso ágil para crear requisitos innovadores para construir mejores soluciones. Este proceso trabaja en estrecha colaboración con *SCRUM* y *Extreme Programming* (XP) para gestionar e implementar estos requisitos. Finalmente, se trabaja con procesos no ágiles como RUP.

LSD: Lean-Software Development, es una traducción de manufactura *lean* y principios y prácticas *lean* de TI (tecnologías de información) al dominio de desarrollo de software. Adaptado del sistema de producción de Toyota, una subcultura pro-lean está emergiendo de dentro de la comunidad ágil (Yasuhiro Monden, 1998).

DDD: Domain-Driven Design, es un enfoque para el desarrollo de software con necesidades complejas mediante una profunda conexión entre la implementación y los conceptos del modelo y núcleo del negocio, provee una estructura de prácticas y terminologías para tomar decisiones de diseño que enfoquen y aceleren el manejo de dominios complejos en los proyectos de software (Evans, 2004).

BDD: Behavior-Driven Development, es un proceso de desarrollo de software que surgió a partir del desarrollo guiado por pruebas (TDD: *Test-Driven Development*). El desarrollo guiado por el comportamiento combina las técnicas generales y los principios de TDD, junto con ideas del diseño guiado por el dominio y el análisis y diseño orientado a objetos para proveer al desarrollo de software y a los equipos de administración, con herramientas compartidas y un proceso compartido de colaboración en el desarrollo de software (Solis and Wang, 2011).

RDD: Responsibility-Driven Design, es una técnica de diseño en programación orientada a objetos, que mejora la encapsulación mediante el uso del modelo cliente-servidor. Fue propuesta por Rebecca Wirfs-Brock y Brian Wilkerson (Wirfs-Brock and Wilkerson, 1989; Wirfs-Brock and McKean, 2003).

1.1.3. Tipos de herramientas de desarrollo de software

Una herramienta de desarrollo de software es un programa informático que se usa por un programador para diseñar, desarrollar, depurar, gestionar o mantener un sistema de software, es decir cumplir con el ciclo de vida del desarrollo de software. Existen diversas herramientas para cumplir con cada una de las actividades del desarrollo de software, por ejemplo para la fase de diseño, se tienen herramientas para modelado UML como Visual Paradigm o MagicDraw™ UML, para llevar a cabo pruebas en el software se tiene Selenium o Testlink por mencionar algunos. Sin embargo también existen herramientas que cumplen con más de una fase o actividad en el desarrollo de software, estas herramientas suelen clasificarse como IDEs y CASE, a continuación se explica cada una de estas.

IDE, un Entorno de Desarrollo Integrado (IDE: *Integrated Development Environment*) es una aplicación visual que sirve para el desarrollo de aplicaciones a partir de componentes. Por lo general todas ellas cuentan con los siguientes elementos:

- una o más ‘paletas’ para mostrar como iconos los componentes disponibles;
- un ‘lienzo’ o ‘contenedor’ en el cual se colocan los componentes y se interconectan entre sí;
- editores específicos para configurar y especializar los componentes;

- visores (browsers) para localizar componentes de acuerdo a ciertos criterios de búsqueda;
- directorios de componentes;
- acceso a editores, interpretes, compiladores y depuradores para desarrollar nuevos componentes;
- y finalmente, acceso a algunas herramientas de control y gestión de proyectos y CSCW, esenciales para grandes proyectos software.

Ejemplos de IDEs son Visual Studio® de Microsoft®, VisualAge de IBM® o VisualCafe™ de Symantec™, complementados con lenguajes de configuración como VBScript y JavaScript (Fuentes, Troya & Vallecillo, n.d.).

CASE, las herramientas CASE (*Computer Aided Software Engineering*, Ingeniería Asistida por Computadora) son diversas aplicaciones de software destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero. Estas herramientas ayudan en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso del diseño del proyecto, cálculo de costos, implementación de parte del código con respecto al diseño dado, compilación automática, documentación o detección de errores entre otras (Sommerville and Sawyer 1997; Sommerville, 2004).

1.1.4. Generación automática de código

La generación automática de código y las herramientas de generación automática de código, sirven para agilizar el desarrollo de sistemas de software e incrementar su confiabilidad, este un concepto bien conocido en el área de la computación (Bell, 1998; Herrington, 2003). Gran parte del esfuerzo que se invierte en la investigación en el área de la generación automática de código, está orientada a la generación de código en lenguajes de alto nivel a partir de modelos abstractos de sistemas de software (Rincón, Hidrobo, & Aguilar, 2010).

1.1.5. Desarrollo de aplicaciones para dispositivos móviles

Con el auge de los teléfonos inteligentes (*smartphones*) y otros dispositivos móviles el mercado se ha dividido, los dispositivos tales como computadoras, tablets, teléfonos

inteligentes y televisiones inteligentes tienen un comportamiento similar en cuanto a la adaptación de las aplicaciones. Sin embargo difieren en cuanto a la resolución, la interacción con el usuario y en ocasiones la forma de presentar las aplicaciones. Por otra parte las plataformas de software son también importantes de mencionar ya que de esto dependerá la forma de desplegar las aplicaciones, cabe mencionar que con el tiempo surgen nuevas plataformas o sistemas operativos, sin embargo hasta el momento Android™ y iOS tienen mayor precedencia en el mercado. Algunas de las plataformas para aplicaciones para dispositivos móviles más importantes al momento son:

Android™: es un sistema operativo basado en el núcleo Linux. Se diseñó principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes, *tablets* o tabléfonos; y también para relojes inteligentes, televisores y automóviles, se desarrolló por Google Inc.

Firefox® OS: es un sistema operativo para dispositivos móviles, basado en HTML5 con núcleo Linux, de código abierto para varias plataformas. Se desarrolló por Mozilla Corporation bajo el apoyo de otras empresas y una gran comunidad de voluntarios de todo el mundo. El sistema operativo está diseñado para permitir a las aplicaciones basadas en HTML5 comunicarse directamente con el hardware del dispositivo usando JavaScript y Open Web APIs.

iOS: es un sistema operativo para dispositivos móviles de la multinacional Apple Inc. Originalmente desarrollado para el iPhone (iPhone OS), después se usó en dispositivos como el iPod touch y el iPad.

Windows Phone®: es un sistema operativo para dispositivos móviles desarrollado por Microsoft®, como sucesor de Windows Mobile.

Actualmente la mayoría de las empresas que se dedican al desarrollo de aplicaciones para dispositivos móviles intentan abarcar el mayor mercado posible y abarcar múltiples dispositivos y plataformas, de ahí que las aplicaciones tengan estas funcionalidades múltiples, y hayan surgido términos como multidispositivo y multiplataforma.

Aplicaciones Multidispositivo: las aplicaciones multidispositivo como su nombre lo indica, permiten el desarrollo en un lenguaje o conjunto de lenguajes y su posterior ejecución en diferentes dispositivos de hardware.

Aplicaciones Multiplataforma: es un atributo conferido a las aplicaciones o programas informáticos que significa su despliegue o ejecución en diversos entornos o sistemas operativos.

1.1.6. Tipo de aplicaciones para dispositivos móviles

La tecnología avanza rápidamente y cada vez es más popular el uso de aplicaciones sobre todo móviles para casi cualquier tarea que requiera desarrollarse, por otro lado es también importante abarcar la mayor cuota de mercado en el desarrollo de aplicaciones, por tal motivo las empresas desarrollan aplicaciones que se utilizan en múltiples dispositivos y en múltiples plataformas, y la elección principal es saber el tipo de aplicación que se desea desarrollar de tres tipos diferentes, aplicaciones nativas, híbridas o basadas en Web.

Aplicaciones Nativas: la aplicación nativa está desarrollada y optimizada específicamente para el sistema operativo determinado y la plataforma de desarrollo del fabricante (Android™, iOS, entre otras). Este tipo de aplicaciones se adapta al 100% con las funcionalidades y características del dispositivo obteniendo así una mejor experiencia de uso (Raona.com, 2017).

Aplicaciones Híbridas: este tipo de aplicación aprovecha al máximo la versatilidad de un desarrollo Web y tiene la capacidad de adaptación al dispositivo como una aplicación nativa. Además, comporta un menor coste que una aplicación nativa y una mejor experiencia de uso que una aplicación Web (Raona.com, 2017).

Aplicaciones basadas en Web: la aplicación Web es la opción más sencilla y económica de desarrollar aplicaciones, puesto que al desarrollar una única aplicación se reducen al máximo los costos de desarrollo. Asimismo, en este tipo de aplicaciones, se utiliza el "diseño Web responsivo", creando así una única aplicación adaptada para todo tipo de dispositivos (Raona.com, 2017).

1.1.7. Procesamiento Digital de Imágenes

El procesamiento digital de imágenes es el conjunto de técnicas que se aplican a las imágenes digitales con el objetivo de mejorar su calidad o facilitar la búsqueda de información inmersa en ellas. En este sentido el tratamiento digital de imágenes contempla el procesamiento y el análisis de imágenes (Gonzalez and Woods, 2006). Este procesamiento se refiere a la realización de transformaciones y a la restauración y mejoramiento de las imágenes. El análisis consiste en la extracción de propiedades y características de las imágenes, así como la clasificación e identificación y el reconocimiento de patrones (De la Rosa, 2007).

MATLAB®, Image Processing Toolbox™ (IPT)

MATLAB® es un lenguaje de alto nivel y un entorno interactivo para el cálculo numérico, la visualización y la programación. Mediante MATLAB®, es posible analizar datos, desarrollar algoritmos y desarrollar modelos o aplicaciones. El lenguaje, las herramientas y las funciones matemáticas incorporadas permiten explorar diversos enfoques y llegar a una solución antes que con hojas de cálculo o lenguajes de programación tradicionales, como son C/C++ o Java™.

Image Processing Toolbox™ es un conjunto de herramientas que se integran a MATLAB®, estas herramientas ofrecen un conjunto completo de algoritmos de referencia estándar, funciones y aplicaciones para el procesamiento de imágenes, análisis, visualización y desarrollo de algoritmos. Realiza análisis de imágenes, segmentación de imágenes, mejora de imagen, reducción de ruido, transformaciones geométricas, y registro de imágenes, entre otros. Muchas de las funciones de herramienta de MATLAB® soportan procesadores multi-núcleo, GPU, y la generación de código C (mathworks, 2016; Cuevas, Zaldivar & Rojas, 2003).

1.1.8. Algoritmos de procesamiento de imágenes

Dentro de los algoritmos para procesamiento de imágenes, existen dos clases de transformaciones de imágenes:

1. **Transformaciones radiométricas**, en las cuales los valores de nivel de gris de los pixeles se alteran sin modificar la geometría de la imagen (contrastes, filtrados,

clasificación, texturas, cocientes). Aquí se distinguen dos grupos de algoritmos de procesamiento de imágenes: puntual y espacial.

- **Procesamiento Puntual:** las operaciones sobre un pixel de la imagen se realizan sin tener en cuenta los pixeles vecinos (ensanche de contraste, umbralización, pseudocolor, operaciones algebraicas entre imágenes).
- **Procesamiento espacial:** la operación para un pixel de la imagen de salida tiene en cuenta tanto el pixel correspondiente en la imagen de entrada como una cantidad arbitraria de vecinos de éste (operaciones de filtrado, gradientes, realce de bordes).

2. **Transformaciones geométricas**, en las que se altera la geometría de la imagen, es decir, la ubicación de los pixeles dentro de esta.

A continuación se presenta una pequeña descripción de algunos algoritmos empleados para el procesamiento de imágenes, es importante mencionar que estos algoritmos son de manera general puesto que existen distintos aplicados para cada uno.

Filtrado: filtro de mediana

Un filtro de mediana es un filtro digital no lineal, que es capaz de preservar los cambios de señal agudos y es muy eficaz en la eliminación de ruido de impulso. Este algoritmo sustituye el valor de un pixel por el valor de la media de los pixeles vecinos (Gonzalez and Woods, 2006). Es capaz de mejorar ciertas características de una imagen que posibiliten efectuar operaciones del procesado sobre ella.

Operaciones morfológicas

Las operaciones morfológicas se utilizan en binario y las imágenes en escala de grises, y es útil en muchas áreas de procesamiento de imágenes, tales como esqueletización, restauración y análisis de la textura. Un operador morfológico utiliza un elemento de estructuración para procesar una imagen.

Operaciones de convolución

La convolución es una operación matemática simple que es fundamental para muchos operadores comunes de procesamiento de imágenes. Convolución es una forma de multiplicar juntos dos conjuntos de números de diferentes tamaños para producir una tercera

matriz de números. La Convolución pertenece a una clase de algoritmos denominados filtros espaciales.

Detección de bordes

La detección de bordes se utiliza ampliamente en la segmentación de la imagen cuando se quiere dividir la imagen en las zonas correspondientes a los diferentes objetos, llamadas zonas de interés.

Momentos invariantes de Hu: Parámetros estadísticos invariantes a la traslación, tamaño del objeto de análisis y rotación de acuerdo con (Gutiérrez, Díaz and Torres, 2014). Los momentos invariantes son uno de los descriptores de forma más usados desde que se introdujeron en los 60´s por Hu (Srihari and Franke, 2008).

1.1.9. Red Neuronal Artificial

Las redes neuronales artificiales son modelos matemáticos que modelan el comportamiento del cerebro humano. En el trabajo presentado por López y Fernández (2008) se mencionó que, las redes neuronales artificiales son sistemas complejos que procesan información, y tienen la capacidad de responder dinámicamente.

Existen diferentes modelos de redes neuronales artificiales algunos de ellos y sus características principales, de acuerdo con (López and Fernández, 2008) son los siguientes:

- **Perceptrón simple:** red unidimensional, con dos capas de neuronas es decir; N celdas de entrada y M neuronas de salida. Modelo útil en tareas de clasificación como para la representación de funciones booleanas, permite discriminar entre dos clases linealmente separables.
- **Adaline (ADAptive LINear Element) y Madaline (Multiple ADALINE):** modelos muy similares al Percetrón simple, Adaline utiliza funciones de transferencia lineales y Madaline permite conectarse de forma sucesiva, en ambos casos el mecanismo de aprendizaje es la regla de Widrow-Hoff.
- **Perceptrón Multicapa:** modelo con propagación hacia adelante, se caracteriza por su organización en capas de celdas disjuntas, esto es, ninguna salida neuronal constituye una entrada para las neuronas de la misma capa o de capas previas. De acuerdo con (Díez, Gómez and de Abajo Martínez, 2001), la arquitectura del

perceptrón multicapa tiene al menos tres capas, una de entrada, una de salida y una o más intermedias. Es el modelo más utilizado en implementaciones comerciales.

1.1.10. Patrones de Diseño de Interfaz de Usuario

Los patrones de diseño de interfaz de usuario son modelos que permiten resolver problemas de diseño con características comunes. En la actualidad los Sistemas Operativos para dispositivos móviles más populares son Android™, iOS, BlackBerry y Windows Phone®, por mencionar algunos. Cada sistema operativo tiene su propia identidad que se refleja en la apariencia y comportamiento de cada uno de los elementos que componen su interfaz. Sin embargo, todos comparten algunos puntos de vista fundamentales que se manifiestan en el diseño de sus interfaces, como lo es la navegación, los cuadros de diálogo, notificaciones, entre otros que todos los sistemas operativos contienen, pero cada uno de ellos los representa en diferente lugar. Se identificaron diferentes categorías de patrones de diseño de interfaz de usuario propuestos por diversos autores, los planteados por Theresa Neil (Neil, 2014), Jennifer Tidwell (Tidwell, 2010), Mari Sheibley (Sheibley, 2013), Anders Toxboe (Toxboe, 2016) y UNITiD (UNITiD, 2016). En la tabla 1.1 se presentan las categorías que cada autor asigna a sus patrones de diseño de interfaz de usuario.

Tabla 1.1. Categorías de patrones de diseño de interfaz de usuario presentado por autor

Autor	Plataforma	Categorías	Patrones identificados
Theresa Neil	Móvil	Navegación (10); Formularios (7); Tablas y listas (8); Búsqueda, ordenamiento y filtración (14); Herramientas (7); Gráficos (8); Llamadas (8); Retroalimentación y ofrecimiento (5); Ayuda (3)	70
Jennifer Tidwell	Móvil y Web	Acciones del usuario (14); Organización del contenido (10); Navegación, indicadores y señalización (13); Organización de la página (13); Listas (12); Acciones y comandos (11); Árboles y gráficas (11); Formularios y controles (11); Medios sociales (12); Diseño móvil (11); Diseño visual (7)	125
Mari Sheibley	Móvil	Patrones de interfaz de usuario (22)	22
Anders Toxboe	Móvil y Web	Obtención de entrada (28); Navegación (25); Incorporación (9); Manejo de datos (11);	89

Autor	Plataforma	Categorías	Patrones identificados
		Social (11); Diverso (5)	
UNITiD	Móvil	Manejo de datos (16); Obtener entradas (8); Navegación (30); Notificaciones (7); Personalizar (6); Interacciones de pantalla (6); Social (4)	77

A continuación se presenta la lista de patrones de diseño aplicados en el presente proyecto:

Splashscreen: este patrón permite mostrar al usuario una pantalla personalizada al iniciar una aplicación (Neil, 2014). En la figura 1.4 se muestra un ejemplo de este PDIU.

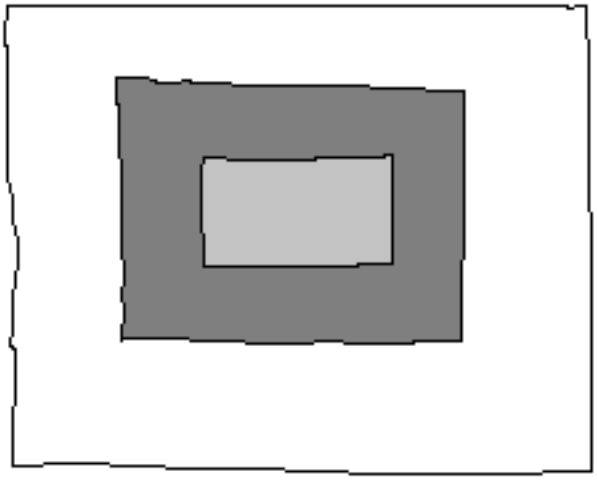


Figura 1.5. Interfaz gráfica de usuario que representa al PDIU Splashscreen

Login: patrón que permite controlar el acceso de usuarios a un área restringida (Neil, 2014). En la figura 1.5 se muestra un ejemplo de este PDIU.

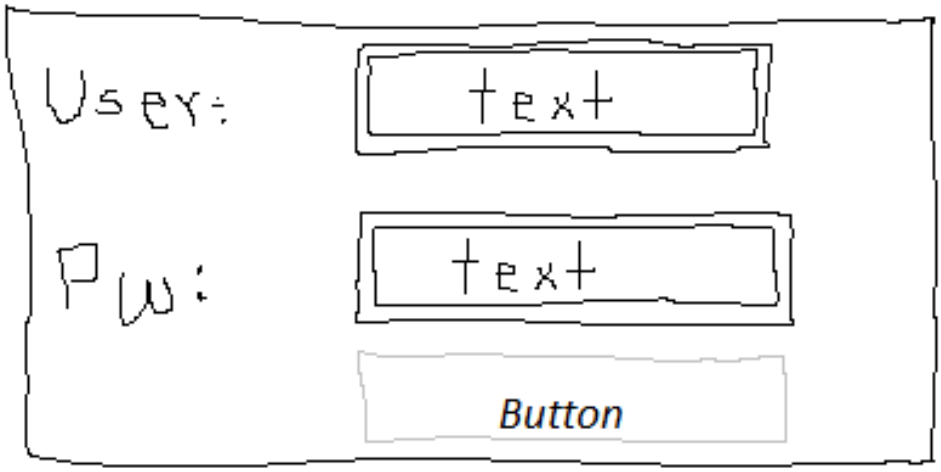


Figura 1.6. Interfaz gráfica de usuario que representa al PDIU Login

Video: Es un patrón de diseño ideal para incorporar contenido multimedia en una aplicación (Neil, 2014). En la figura 1.6 se muestra un ejemplo de este PDIU.

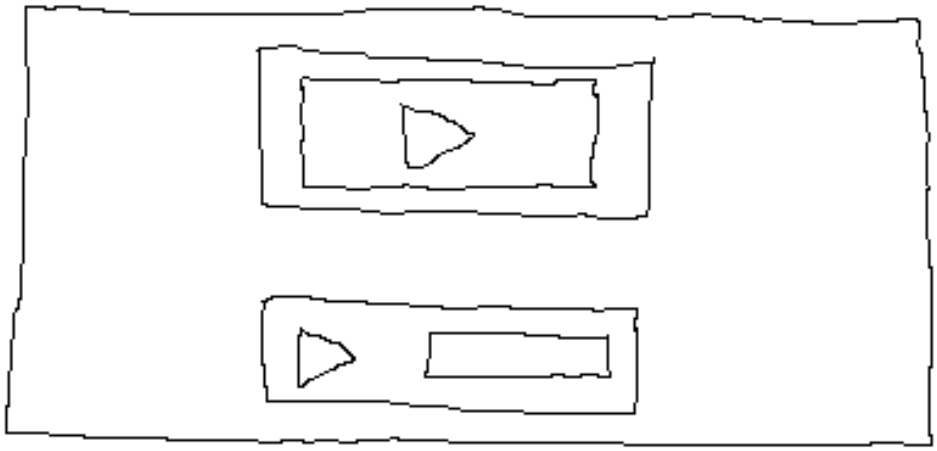


Figura 1.7. Interfaz gráfica de usuario que representa al PDIU Video

Dashboard: Permite al usuario visualizar de forma rápida el contenido más importante de una aplicación, sin la necesidad de navegar en otra pantalla (UI-Patterns, 2016). En la figura 1.7 se muestra un ejemplo de este PDIU.

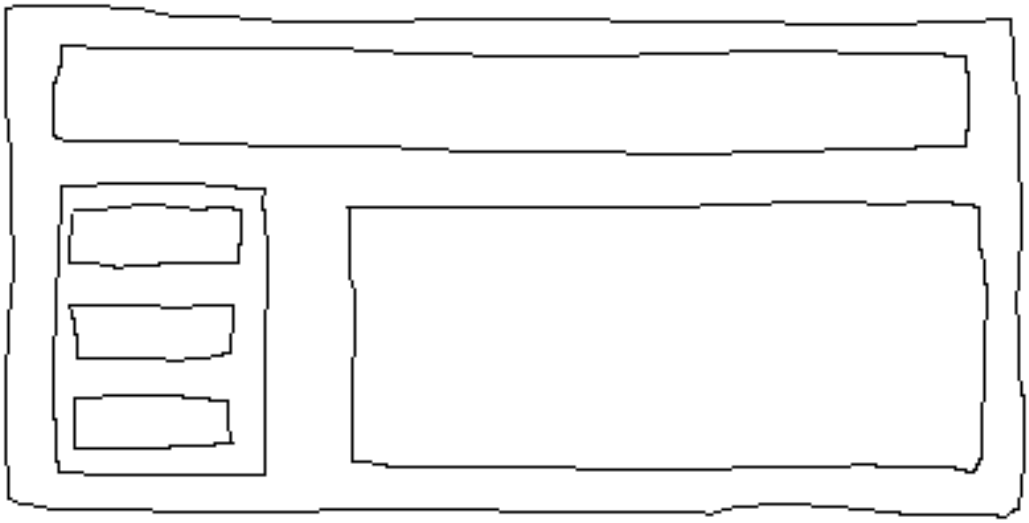


Figura 1.8. Interfaz gráfica de usuario que representa al PDIU Dashboard

Carrousel: Este patrón se utiliza para navegar rápidamente por un conjunto de páginas Web o imágenes utilizando el gesto de desplazamiento (Neil, 2014; UI-Patterns, 2016). En la figura 1.8 se muestra un ejemplo de este PDIU.

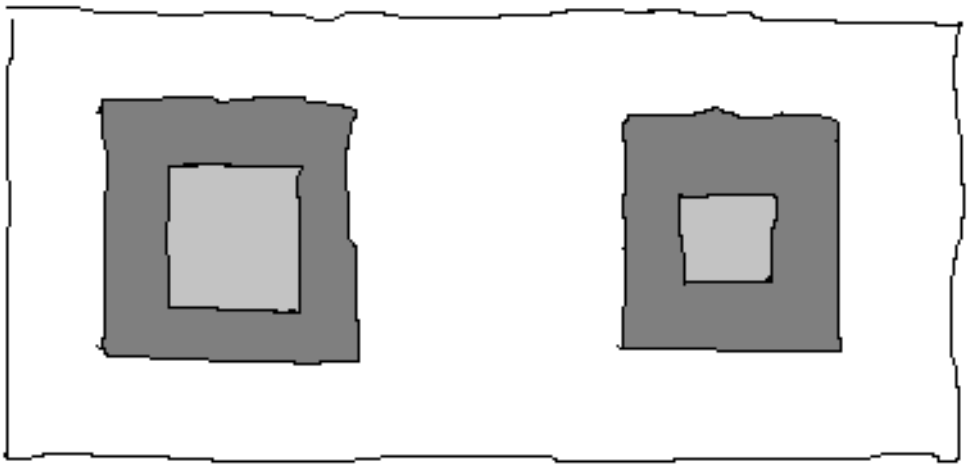


Figura 1.9. Interfaz gráfica de usuario que representa al PDIU Carousel

1.2.Planteamiento del problema

Dentro de la ingeniería de software existe un área muy importante que es la generación automática de software sin embargo las herramientas y lo procesos empleados en esta área aun dejan mucho que desear, es decir los procesos son muy similares entre sí, y requieren de

un amplio conocimiento en diseño, modelado y desarrollo de software lo cual al incrementar la curva de aprendizaje deja rezagados a muchos desarrolladores.

La problemática más relevante es el tiempo empleado en el desarrollo de sistemas de software y el costo que esto implica, es decir, mientras más tiempo tome a un desarrollador o a un grupo de desarrolladores concluir un trabajo de desarrollo de software más costos será este.

Las propuestas de soluciones existentes en la actualidad presentan características muy similares, como modelado UML, diseño de modelos propios o dictado de requerimientos lo cual no reduce en gran medida el tiempo empleado al menos en las primeras fases del ciclo de vida del desarrollo de software.

El desarrollo de sistemas de software que proporcionen las funcionalidades necesarias para ofrecer soluciones efectivas durante el desarrollo automático de otros sistemas de software no es una tarea que tenga una solución trivial ni sencilla. Las aplicaciones para la generación automática de código existentes proporcionan frecuentemente mecanismos y características de desarrollo muy similares entre sí como se mencionó anteriormente, mismas que son costosas en términos de tiempo y dinero para los desarrolladores.

Por lo cual es necesario desarrollar nuevas propuestas de solución para la generación automática de software que sean efectivas en la reducción de tiempo/esfuerzo y por su puesto en costos.

En este contexto una solución es desarrollar una herramienta que proporcione la funcionalidad de generar el código fuente necesario para el desarrollo de aplicaciones multidispositivo a partir del uso de técnicas para el procesamiento de imágenes y que permitan identificar determinados elementos contenidos en una imagen digitalizada (previamente delimitada), con esto se pretende reducir costos y mejorar los procesos de desarrollo automático de software.

1.3.Objetivos

1.3.1. Objetivo General

Desarrollar un conjunto de nuevos métodos computacionales que permita la generación automática de software multidispositivo y multiplataforma, a través de técnicas de reconocimiento de imágenes y de patrones de diseño de interfaces de usuario.

1.3.2. Objetivos Específicos

- Estudiar, analizar y utilizar las técnicas de análisis y procesamiento de imágenes para la identificación de los elementos contenidos dentro de las imágenes
- Estudiar, analizar y utilizar las técnicas de reconocimiento de patrones para identificar los patrones de diseño de interfaces en las imágenes
- Delimitar el tipo, calidad y contenido de las imágenes a utilizar, con la finalidad de hacer un reconocimiento de imágenes más eficiente
- Investigar, estudiar, analizar y utilizar herramientas y técnicas de programación que soporten desarrollo de aplicaciones multidispositivo, para desarrollar aplicaciones con esta funcionalidad
- Definir y realizar la integración de la herramienta desarrollada, para poner en práctica los casos de prueba
- Definir los casos de estudio para aplicación de la herramienta desarrollada, con el objetivo de realizar la evaluación de la herramienta y comprobar su funcionalidad
- Determinar los tipos de lenguajes de programación soportados para la generación de código fuente, con la finalidad de realizar generación de aplicaciones multiplataforma
- Desarrollar una arquitectura para la herramienta de generación de código, con la finalidad de establecer la funcionalidad completa de la herramienta
- Definir los niveles de prestación de la herramienta desarrollada, con la finalidad de realizar una generación de código más eficiente y reducir los posibles errores
- Desarrollar técnicas para la interpretación y reconocimiento de imágenes para la transformación a componentes de una aplicación de software y que posteriormente generará el código fuente de la aplicación multidispositivo y multiplataforma, con los elementos de la imagen
- Validar el funcionamiento de la herramienta a través de la evaluación de la misma con respecto a otras herramientas similares

1.4.Hipótesis

Todo lo escrito en el aparatado del planteamiento del problema tiene la finalidad de plantear formalmente una hipótesis válida tal como la que se presenta a continuación:

El desarrollo de un nuevo enfoque de proceso de desarrollo de software basado en el uso de técnicas de inteligencia artificial para la generación de código a partir del procesamiento de imágenes, redes neuronales y reconocimiento de patrones de diseño de interfaces de usuario permitirá ofrecer mecanismos para agilizar y facilitar el desarrollo de aplicaciones multidispositivo y multiplataforma en la Ingeniería de Software.

Lo anterior permitirá establecer y representar cualquier aplicación de software como un conjunto de patrones de diseño de interfaces de usuario.

Esto permitirá generar nuevo conocimiento en las ciencias computacionales, proporcionando un avance muy significativo en la Ingeniería de Software, siendo generador de un nuevo paradigma que abre puertas a nuevos enfoques y formas de analizar, diseñar y desarrollar software.

1.5.Justificación

La motivación para la realización de este trabajo surge en un principio de la necesidad de transformar las funcionalidades de los marcos de trabajo para la Web, tradicionalmente utilizados por los desarrolladores, programadores y todos aquellos profesionales del desarrollo de software. El uso de una aplicación que proporcione la funcionalidad de generar el código fuente necesario para el desarrollo de aplicaciones multidispositivo a partir del uso de técnicas para el procesamiento de imágenes y que permita identificar determinados elementos contenidos en una imagen digitalizada (previamente delimitada ya sea una interfaz gráfica de usuario o un ADV, Abstract Data View, por mencionar algunos ejemplos), asegura un avance importante en la manera de desarrollar sistemas computacionales basados en la Web.

Al revisar los marcos de trabajo existentes para el desarrollo de aplicaciones para dispositivos móviles multidispositivo, se observa claramente que sus características están limitadas, es decir algunos solo proveen las bibliotecas de funciones para el desarrollo y los más completos generan código que se centra en transformar lo que escribe el desarrollador en un esquema que se implementa en diferentes dispositivos móviles y con diferentes sistemas operativos. Sin embargo no ofrecen algo innovador, de ahí surge lo novedoso de esta propuesta ya que de manera oficial y hasta el momento, se desconoce la existencia de algún marco de trabajo

de aplicaciones multidispositivo que proporcione toda la funcionalidad que esta propuesta plantea, tal como la generación de código a partir de una imagen digitalizada, solo por mencionar una de las funcionalidades más importantes, lo que conlleva a la obtención de los siguientes beneficios:

- Agilizar y facilitar en los marcos de trabajo existentes el desarrollo de aplicaciones multidispositivo.
- Generar únicamente el código fuente de los componentes que se consideren necesario y que previamente se identificaron dentro de alguna imagen debidamente delimitada.
- Generar el código fuente de una interfaz gráfica de usuario completa a partir de una imagen debidamente delimitada.

1.6.Aportaciones

El presente trabajo de tesis presenta diversas aportaciones las cuales se resumen en 2 principales puntos, los cuales se presentan a continuación:

- Una arquitectura de software para el desarrollo de un sistema de generación automática de aplicaciones multidispositivo
- Un nuevo proceso para la generación de aplicaciones multidispositivo basado en el reconocimiento de elementos de interfaces de usuario a través de técnicas de procesamiento de imágenes

Capítulo 2: Estado del arte

Para comprender mejor el estado del arte, éste se ha dividido en dos subsecciones, la primera en torno a los enfoques de desarrollo de software y la segunda en torno a las herramientas para el desarrollo de aplicaciones.

2.1. Enfoques de desarrollo de software

Existen varios enfoques de desarrollo de software para la generación automática de código tales como MDA, MDD, MDE, FDD, TDD, por mencionar algunos, en esta sección se presentan los trabajos más relevantes que abordan algunos de estos enfoques, los trabajos fueron seleccionados por temática, es decir por el enfoque de desarrollo que utilizaron, así como también se seleccionaron los trabajos más recientes y más citados.

En cuanto a los enfoques MDA, MDD y MDE, los autores Dunkel y Bruns (2007) propusieron el proyecto BAMOS, un enfoque de software diseñado e implementado para el desarrollo genérico y flexible de aplicaciones para dispositivos móviles. La arquitectura del proyecto BAMOS se basó en la descripción declarativa de los servicios disponibles. Es un enfoque basado en modelos para generar el código fuente casi completo de servicios móviles. Mediante la aplicación de un modelo de desarrollo impulsado por el enfoque, un nuevo servicio es convenientemente modelado con una herramienta de modelado gráfico y los modelos gráficos se utilizan para generar las descripciones XML correspondientes de la interfaz de usuario móvil y la especificación de flujo de trabajo. Para utilizar dicho servicio, no se implementa ningún código fuente específico en el dispositivo móvil.

Del mismo modo, Balagtas-Fernández y Hussmann (2008) argumentaron que el objetivo en la generación de aplicaciones para dispositivos móviles era simplificar el proceso de generación mediante el desarrollo de un modelo independiente de plataforma de alto nivel de una aplicación y transformar automáticamente este modelo en código específico de plataforma. El método de investigación que utilizaron fue una combinación del enfoque de desarrollo impulsado por el modelo (MDD) en el desarrollo de software y la aplicación de técnicas en el campo de la interacción hombre-computadora (HCI), particularmente en el

diseño del sistema centrado en el usuario. Con lo cual propusieron el desarrollo de un lenguaje de modelado gráfico específico para las aplicaciones para dispositivos móviles y el desarrollo de un algoritmo genérico para la conversión de este modelo gráfico en código. Sin embargo, su objetivo principal fue el diseño del modelo gráfico y las técnicas de interacción que permiten a los usuarios no especializados desarrollar aplicaciones para dispositivos móviles especializadas.

Desde una perspectiva similar, Heitkötter, Majchrzak y Kuchen (2013) presentaron MD2, un enfoque basado en modelos y multiplataforma para el desarrollo de aplicaciones, con el que los desarrolladores especifican una aplicación en un lenguaje de alto nivel (dominio específico) diseñado para describir sucintamente las aplicaciones empresariales. De este modelo, se generan automáticamente las aplicaciones nativas para Android™ e iOS. MD2 se desarrolló en estrecha cooperación con la industria y proporciona medios para desarrollar aplicaciones basadas en datos con una apariencia nativa. Las aplicaciones acceden al hardware del dispositivo e interactúan con servidores remotos.

Desde otra perspectiva, Núñez-Valdez et al. (2016) propuso un enfoque para el desarrollo ágil de videojuegos multiplataforma mediante el uso de modelos de abstracción de alto nivel. Este enfoque ofrece una solución viable soportada por una herramienta que permite el desarrollo de videojuegos de una manera sencilla y ágil. Para aumentar la competitividad, los autores utilizaron un enfoque de ingeniería impulsada por modelos (MDE) para desarrollar videojuegos multiplataforma de una manera rápida y fácil. MDE ayuda a reducir el número de errores, la cantidad de tiempo requerido, costos de desarrollo y favorece un incremento de la productividad.

También, Vaupel et al. (2016) presentaron un lenguaje de modelado y una infraestructura para el MDD de aplicaciones nativas en Android™ e iOS. Este enfoque permite un desarrollo de aplicaciones flexibles en diferentes niveles de abstracción: modelado compacto de elementos estándar de la aplicación, como la gestión de datos estándar y el modelado cada vez más detallado de elementos individuales para cubrir, por ejemplo, un comportamiento específico. Además, se admite un tipo de modelado de variabilidad que permite desarrollar

aplicaciones para dispositivos móviles con variantes. Se demostró también que el enfoque MDD se utilizó en varias aplicaciones, incluyendo una aplicación de conferencia, una guía de museo con funcionalidad de realidad aumentada y un SmartPlug.

Vaquero-Melchor et al. (2017) mencionaron que tradicionalmente, el modelado utilizando DSL (*Domain-Specific Languages*) tiene soporte por computadoras de escritorio en entornos estáticos que descuidan la información contextual circundante. En cambio, los autores afirman que las DSL también son muy útiles en un entorno dinámico donde se benefician de la movilidad y el contexto. Por lo tanto, afirman que identificaron varios escenarios donde el modelado utilizando dispositivos móviles es útil.

Cassani et al. (2017) presentaron una metodología de diseño y una plataforma de acompañamiento para el diseño y desarrollo rápido de Context-Aware Mobile mashUpS (CAMUS). El enfoque se caracteriza por el papel dado al contexto como una dimensión de modelado de primera clase utilizada para apoyar: 1) la identificación de los recursos más adecuados que satisfacen las necesidades situacionales de los usuarios y 2) la consecuente adaptación en tiempo de ejecución de los datos y funciones proporcionados. Gracias a la adopción de técnicas de ingeniería impulsada por modelos (MDE), estos modelos impulsan la ejecución flexible de la aplicación final en los dispositivos móviles de destino.

Bajo el enfoque FDD, Firdaus, Ghani y Yasin (2013) explicaron que los procesos ágiles, como Feature Driven Development (FDD), Scrum y Extreme Programming (XP), son criticados por no proporcionar un marco adecuado para la construcción de software seguro. Por lo tanto se propusieron investigar si el FDD soporta el cambio de requisitos y la seguridad del software en conjunto. Finalmente encontraron que FDD tenía el potencial para adaptarse con el ciclo de vida de desarrollo seguro.

Estrada Cota et al. (2016) reportaron el desarrollo de un sistema Web que agiliza el proceso de solicitudes de mantenimiento eléctrico en cualquier momento y desde cualquier lugar; a su vez ordena, agiliza y transparenta las asignaciones de trabajo, solicitudes de material y monitoreo de la solicitud del trabajo eléctrico, lo cual brinda un mejor servicio dentro de la

institución. Este sistema Web basado en FDD es capaz de concentrar en un solo lugar la información que permite generar informes oportunos que apoyen el proceso de toma de decisiones en mejora del proceso de solicitudes de mantenimiento eléctrico en una institución.

En lo que respecta al enfoque TDD, Kim, Choi y Yoon (2009) propusieron los métodos de pruebas de rendimiento basados en TDD con respecto a factores no funcionales, así como la funcionalidad del software durante el proceso de desarrollo de software realizando pruebas de rendimiento a la etapa de desarrollo y también introdujeron una herramienta que ayuda a las pruebas de rendimiento en la fase de desarrollo de software. La importancia de las pruebas se enfatiza en TDD y el marco de pruebas automatizado es compatible con un desarrollo de software eficiente con pruebas unitarias.

Del mismo modo, Amalfitano, Fasolino y Tramontana (2011) abordaron el problema de las pruebas automáticas de aplicaciones para dispositivos móviles desarrolladas para la plataforma Google Android™, y presentaron una técnica para pruebas rápidas y pruebas de regresión de aplicaciones Android™. La técnica se basó en un *crawler* que construye automáticamente un modelo de la GUI de la aplicación y obtiene casos de prueba que se ejecutan automáticamente. La técnica está soportada por una herramienta para *crawling* la aplicación y generar los casos de prueba.

Con respecto al enfoque PPRD, Colombo-Mendoza, et al. (2013) presentaron el enfoque llamado PPRD (por sus siglas en inglés Phases Process for RIAs Development), además propusieron una herramienta visual que implementa un enfoque basado en patrones GUI para la generación de código de RIAs para múltiples dispositivos. Esta herramienta visual llamada AlexandRIA es un generador de código fuente nativo para el Desarrollo de Aplicaciones Rápidas (RAD), que permite generar automáticamente código basado en un conjunto de preferencias seleccionadas a través de un asistente.

Alor-Hernández, Rosales-Morales & Colombo-Mendoza (2014), describieron el uso del PPRD que es un proceso de desarrollo para desarrollar RIAs el cual se centra en las

principales actividades para transformar los requisitos de los usuarios en un producto de software, estos requisitos de software se transforman en componentes de las aplicaciones generadas, las aplicaciones son RIAs y se implementan en múltiples dispositivos.

En cuanto al enfoque basado en D3 para el desarrollo de aplicaciones, Enard et al. (2013) propusieron una metodología de desarrollo orientada al diseño para sistemas de computación resilientes. La computación resiliente se define como la capacidad de un sistema para mantenerse fiable cuando se enfrentan a los cambios. Para mitigar fallas en tiempo de ejecución, los sistemas fiables se aumentan con mecanismos de tolerancia a fallos como técnicas de replicación. Estos mecanismos tienen que ser sistemática y rigurosamente aplicados para garantizar la conformidad entre el comportamiento en tiempo de ejecución de la aplicación y sus requisitos de confiabilidad. El enfoque propuesto consiste en refinar el diseño con especificaciones dedicadas a las preocupaciones de fiabilidad. Este diseño se aprovecha para respaldar el desarrollo de la aplicación, garantizando al mismo tiempo la trazabilidad de los requisitos de fiabilidad a lo largo del ciclo de vida de la aplicación, incluida la adaptación en tiempo de ejecución.

Kabáč y Consel (2015) propusieron un enfoque de desarrollo basado en el diseño dedicado al dominio de la orquestación de masas de sensores. El desarrollador declara lo que hace una aplicación utilizando un lenguaje específico de dominio (DSL). El compilador procesa declaraciones específicas de dominio para generar un marco de programación personalizado que guía y apoya la fase de programación.

Mientras tanto, bajo el enfoque LSD, Vallon et al. (2015) propuso el modelo ALP-mobile, un proceso ágil y delgado para el desarrollo de aplicaciones para dispositivos móviles, en el que se combinan elementos de Scrum, Kanban y eXtreme Programming (XP). ALP-mobile se centra en proyectos de software para dispositivos móviles que se desarrollan para un cliente específico. ALP-mobile cubre todo el ciclo de vida del proyecto en el desarrollo de aplicaciones para dispositivos móviles incluyendo la definición, así como la implementación y prueba del producto, llevando la aplicación a su mercado deseado y proporcionando el mantenimiento adecuado del software para garantizar alta calidad.

Kupiainen, Mäntylä e Itkonen (2015) propusieron aumentar el conocimiento de las razones y efectos del uso de métricas en el desarrollo ágil industrial. Del mismo modo, sugirieron centrarse en las métricas que utilizan los equipos ágiles, en lugar de las utilizadas por investigadores de ingeniería de software. Además analizaron la influencia de las métricas utilizadas. Los resultados del estudio indicaron que las razones y los efectos del uso de métricas se centran en las siguientes áreas: planificación del sprint, seguimiento del progreso, medición de la calidad del software, solución de problemas de proceso de software y motivación de las personas. Además, se demostró que aunque los equipos ágiles usan muchas métricas sugeridas en la literatura ágil, también usan muchas métricas personalizadas. Finalmente, las métricas más influyentes en los estudios primarios fueron estimación de velocidad y esfuerzo.

En cuanto al enfoque DDD para el desarrollo de aplicaciones, James y Lalonde (2015) lo describieron como una vasta colección de reglas, términos, directrices y patrones. Domain-Driven Design es un marco de trabajo para diseñar y desarrollar soluciones de software en las que el diseño se basa en las entidades que conforman el dominio de negocio. DDD proporciona un conjunto de directrices que ayudan a alinear el diseño e implementación de software con las expectativas del cliente.

Mientras que Soares et al. (2015) propuso una herramienta de desarrollo en la que el desarrollador sólo modela los objetos de negocio, las asociaciones entre objetos y sus comportamientos usando patrones de dominio y patrones de diseño. El código se genera basándose en estos patrones de diseño y un marco de trabajo, que implementa los patrones arquitectónicos Naked Objects, el cual tiene la responsabilidad por la infraestructura.

Por último, en el enfoque BDD para el desarrollo de aplicaciones, Rocha Silva, Hak y Winckler (2016) propusieron un enfoque basado en el desarrollo impulsado por el comportamiento (BDD) para apoyar la evaluación automatizada de artefactos a lo largo del proceso de desarrollo de sistemas interactivos. El trabajo utiliza una ontología para

especificar pruebas que se ejecutan a través de múltiples artefactos compartiendo conceptos similares.

Del mismo modo, Rocha Silva, Hak y Winckler (2016) llevaron a cabo un caso de estudio de modelos de tareas, prototipos e interfaces de usuario finales para demostrar la viabilidad del enfoque basado en el desarrollo impulsado por el comportamiento (BDD) desde las primeras fases del proceso de diseño, proporcionando un aseguramiento continuo de la calidad de los requisitos y ayudando a los clientes y los equipos de desarrollo a identificar posibles problemas e incoherencias antes de comprometerse con la implementación del software.

Para resumir los enfoques de desarrollo de software antes mencionados y visualizar sus características de una manera fácil, se presenta a continuación la Tabla 2.1.

Tabla 2.1. Comparativa de los enfoques de desarrollo de software

Autores	Enfoque de desarrollo de software	Tipo de aplicación	Técnica
Dunkel y Bruns (2007)	MDA	Proyecto BAMOS, una aplicación para dispositivos móviles	Con una herramienta de modelado gráfico
Balagtas-Fernández y Hussmann (2008)	MDD	Aplicaciones para plataformas para dispositivos móviles	El método de investigación es una combinación del enfoque MDD en el desarrollo de software y la aplicación de técnicas en el campo de la interacción hombre-computadora (HCI)
Heitkötter, Majchrzak y Kuchen (2013)	MDD	Aplicaciones empresariales para Android™ e iOS	MD2, un enfoque para el desarrollo de aplicaciones multiplataforma basado en modelos
Núñez-Valdez et al. (2016)	MDE	Videojuegos multiplataforma	Modelos de abstracción de alto nivel
Vaupel et al. (2016)	MDD	Aplicaciones nativas en Android™ e iOS	El enfoque permite un desarrollo de aplicaciones flexible en diferentes niveles de

Autores	Enfoque de desarrollo de software	Tipo de aplicación	Técnica
			abstracción, por ejemplo, un comportamiento específico
Firdaus, Ghani y Yasin (2013)	FDD	Software seguro (Sitios Web)	FDD para el desarrollo de software seguro
Estrada Cota et al. (2016)	FDD	Sistema Web que agiliza el proceso de aplicación de mantenimiento eléctrico	El desarrollo de este sistema Web se implementó con FDD ya que se centra más en los resultados
Kim, Choi y Yoon (2009)	TDD	Aplicaciones para dispositivos móviles	Ejecutar pruebas en el proceso de desarrollo de todas las aplicaciones para dispositivos móviles y ayudar a detectar fallas
Amalfitano, Fasolino y Tramontana (2011)	TDD	Aplicaciones Android™	La técnica está soportada por una herramienta para rastrear la aplicación y generar los casos de prueba
Colombo-Mendoza, et al. (2013)	PPRD	RIAs	Proceso para el desarrollo de RIAs
Alor-Hernández, Rosales-Morales & Colombo-Mendoza (2014)	PPRD	RIAs	Desarrollo de RIAs basadas en los requisitos
Enard et al. (2013)	D3	Sistemas de cómputo resilientes	El enfoque consiste en refinar el diseño con especificaciones dedicadas a las preocupaciones de fiabilidad
Kabáč y Consel (2015)	D3	Orquestación de masas de sensores	Un enfoque de desarrollo de software que cubre todas las fases de una aplicación de orquestación de masa de sensores
Vallon et al. (2015)	LSD	Aplicaciones para dispositivos móviles	ALP-mobile, combina elementos de Scrum, Kanban y eXtreme Programming (XP)
Kupiainen, Mäntylä e Itkonen (2015)	LSD	Desarrollo de software ágil	El uso de métricas en el desarrollo de software ágil

Como principales conclusiones es importante tener en cuenta que estas propuestas no son muy intuitivas, y su uso en el desarrollo de aplicaciones para dispositivos móviles requiere

conocimiento previo de su funcionamiento. Además, dado que ninguno de estos enfoques de desarrollo de software está totalmente automatizado, los modelos de diseño y sus modificaciones eventuales requieren mucho tiempo. Básicamente la mayoría de las propuestas presentadas en la tabla 2.1 independientemente del enfoque o metodología empleada requiere de conocimientos previos en cuanto a la metodología y en cuanto a diseño y modelado de software en un grado avanzado, sin embargo es importante mencionar que no todos los desarrolladores de software son expertos en diseño y además se requiere ahorrar tiempo en cuanto al diseño y modelado de software, ya que esto reduce los costos en el proceso de desarrollo del software.

2.2. Herramientas para desarrollo de aplicaciones y desarrollo automático de aplicaciones

Existen diversos tipos de herramientas de desarrollo de aplicaciones tales como IDEs o herramientas CASE, por mencionar algunas, las cuales se utilizan dependiendo de la aplicación a desarrollar en función de la plataforma a utilizar y de las prestaciones de las propias herramientas. A continuación se presentan los trabajos más relevantes al respecto en cuanto a herramientas para desarrollo de aplicaciones y desarrollo automático de aplicaciones.

En lo que respecta a la generación automática de código para aplicaciones para dispositivos móviles, Yang, Prasad y Xie (2013) presentaron un nuevo método de caja gris (*grey-box*) para extraer automáticamente un modelo de una aplicación para dispositivos móviles dada. En este enfoque, el análisis estático extrae el conjunto de eventos soportados por la interfaz gráfica de usuario (GUI) de la aplicación. También presentaron una herramienta que implementa el enfoque desarrollado para la plataforma Android™. La evaluación empírica de la herramienta en varias aplicaciones de Android™ demostró que extrae eficientemente modelos compactos pero razonablemente completos de alta calidad para tales aplicaciones.

Asimismo, Amalfitano et al. (2012) presentaron AndroidRipper, una técnica automatizada que prueba las aplicaciones de Android™ a través de su interfaz gráfica de usuario (GUI).

AndroidRipper se basa en un ripper controlado por la interfaz de usuario que explora automáticamente la interfaz gráfica de usuario de la aplicación con el objetivo de ejercer la aplicación de forma estructurada. Además, evaluaron AndroidRipper en una aplicación de código abierto para Android™. Los resultados mostraron que los casos de prueba basados en GUI son capaces de detectar fallas severas, previamente desconocidas, en el código subyacente, y la exploración estructurada supera un enfoque aleatorio.

Mientras tanto, Hu y Neamtiu (2011) presentaron un enfoque para el proceso de pruebas automatizadas para aplicaciones Android™ con un enfoque en errores GUI, primero realizaron un estudio de minería de errores para entender la naturaleza y frecuencia de los errores que afectan a las aplicaciones de Android™; y el estudio reveló que los errores GUI son bastante numerosos. Posteriormente, presentaron técnicas para la detección de errores GUI por generación automática de casos de prueba, alimentando los eventos aleatorios de la aplicación, produciendo archivos de registro / rastreo y analizándolos después de la ejecución. Se demostró cómo estas técnicas ayudaron a descubrir los errores existentes y encontrar nuevos errores, y cómo se utilizarían para prevenir ciertas categorías de errores. Con lo cual se determinó que estas técnicas tienen el potencial de ayudar a los desarrolladores a aumentar la calidad de las aplicaciones de Android.

Mientras que Dinh et al. (2013) describió un enfoque de cómputo móvil en la nube (MCC). MCC integra el cómputo en la nube en el entorno móvil y supera obstáculos relacionados con el rendimiento (por ejemplo, duración de la batería, almacenamiento y ancho de banda), entorno (por ejemplo, heterogeneidad, escalabilidad y disponibilidad) y seguridad en cómputo móvil.

De manera similar, Mao (2016) discutió un estudio de usuario o registros de aplicaciones para evaluar QoE (Calidad de Experiencia) a través de métricas subjetivas, tales como puntuaciones de experiencia de usuario y participación de usuarios. Sin embargo estos experimentos son costosos en esfuerzos humanos y menos capaces de controlar las variaciones del comportamiento del usuario. Para superar estas limitaciones, propusieron Prometheus que mide las métricas QoE objetivas, como la relación de *rebuffering* de vídeo,

para eliminar la dependencia del comportamiento del usuario, pero requiere que el código fuente de la aplicación registre eventos de UI, limitando su aplicabilidad.

Asimismo, Amalfitano et al. (2015) presentaron MobiGUITAR (Mobile GUI Testing Framework) para pruebas automatizadas basadas en GUI de las aplicaciones Android™. MobiGUITAR se basa en la observación, extracción y abstracción del estado en tiempo de ejecución de los widgets GUI. La abstracción es un modelo de máquina de estado escalable que, junto con los criterios de cobertura de prueba, proporciona una forma de generar automáticamente casos de prueba.

Zhu et al. (2015) propuso un enfoque secuencial basado en el modelo de Markov oculto (HMM: Hidden Markov Model) para modelar la información de popularidad de las aplicaciones para dispositivos móviles hacia los servicios de este tipo de aplicaciones. Específicamente, se propuso un modelo de popularidad basada en HMM (PHMM) para modelar las secuencias de las observaciones heterogéneas de popularidad de las aplicaciones para dispositivos móviles. Además, demostraron que el PHMM es un modelo general y se aplica a diversos servicios de aplicaciones para dispositivos móviles, como la recomendación de aplicaciones basada en tendencias, la clasificación y la revisión de la detección de spam y la clasificación de la detección de fraudes.

Finalmente, Possatto y Lucrédio (2015) propusieron la automatización como una forma de reducir el esfuerzo adicional necesario para mantener sincronizadas las plantillas y el código de referencia, para lo cual desarrollaron un mecanismo para detectar y propagar semiautomáticamente cambios de código de referencia a plantillas, manteniéndolos sincronizados con menos esfuerzo.

En cuanto al código generado ejecutado o desplegado en diferentes plataformas de software, Acerbis et al. (2015) propuso una herramienta llamada WebRatio Mobile Platform para el desarrollo basado en modelos de aplicaciones para dispositivos móviles. La herramienta soporta desarrolladores en la especificación del modelo de dominio y del modelo de interacción para aplicaciones para dispositivos móviles. La especificación se construye de

acuerdo con una versión extendida del lenguaje estándar de la OMG llamado IFML (Interaction Flow Modeling Language).

Del mismo modo, Choi, Yang y Jeong (2009) definieron una arquitectura común para el software de aplicaciones para dispositivos móviles, denominado Application Framework, para soportar el desarrollo de software ágil. El marco se define siguiendo PIM (Platform Independent Model) en MDD para satisfacer la necesidad de soporte de una aplicación a varias plataformas para dispositivos móviles.

Mientras que Miravet et al. (2009) presentaron DIMAG (*Device Independent Mobile Application Generation*), para generar aplicaciones para dispositivos móviles para múltiples plataformas de software mediante una descripción declarativa de la aplicación. Además, la generación de una aplicación con este marco tiene en cuenta requisitos adicionales como la necesidad de las aplicaciones para dispositivos móviles de consumir y proporcionar datos desde y hasta máquinas remotas, o la carga dinámica de las clases para la generación de código para una plataforma para dispositivos móviles determinada después de la identificación del dispositivo exige una copia de la aplicación.

En un contexto similar, Cimitile, Risi y Tortora (2011) propusieron una arquitectura común y un proceso de desarrollo unificado que implementa aplicaciones portátiles basadas en servicios móviles. La arquitectura se basa en el patrón de diseño Modelo-Vista-Crontrolador (MVC) y proporciona un marco que genera el código a partir de una especificación algebraica formal. Esta especificación integra diferentes formalismos como las fórmulas LTL (Lógica Temporal Lineal) y la programación funcional, permitiendo la descripción de estructura y aspectos dinámicos de una aplicación para dispositivos móviles. Además, mostraron cómo el marco de trabajo generó aplicaciones de mapas Web que integran diferentes servicios móviles

Heitkötter, Hanschke y Majchrzak (2013) y Heitkötter, Hanschke y Majchrzak (2013) recopilaron un conjunto de criterios para evaluar los enfoques de desarrollo multiplataforma a fin de comparar soluciones concretas entre plataformas. Basándose en estos criterios,

evaluaron aplicaciones Web, aplicaciones desarrolladas con PhoneGap o Titanium Mobile y, aplicaciones desarrolladas de forma nativa. Presentaron los resultados como tablas de referencia y demostraron que estos criterios son viables para las evaluaciones. Por ultimo con respecto a los enfoques, encontraron que PhoneGap es muy cercano a un aspecto nativo.

Gavalas y Economou (2011) presentaron una revisión de las principales características, méritos relativos y deficiencias de las opciones de plataforma de desarrollo más populares, específicamente, Java ME, .NET Compact Framework, Flash Lite y Android™. Los autores evaluaron y compararon estas opciones de acuerdo con diversos criterios cuantitativos y cualitativos. Su objetivo es servir de punto de referencia y guía para los desarrolladores y profesionales en la elección de una plataforma móvil para el desarrollo de dispositivos de información.

Asimismo, Charland y Leroux (2011) discutieron algunas de las fortalezas y debilidades de los enfoques basados en la Web y los nativos, con especial atención a las áreas donde la brecha se está cerrando entre las tecnologías Web y sus contrapartes nativas. Llegando a la conclusión de que el resultado probable es una solución híbrida y que para producir la mejor experiencia de usuario posible, las implementaciones deben proporcionar diseños y código que soporten las expectativas establecidas por el contexto particular de un usuario.

Dalmasso et al. (2013) proporcionaron varios criterios de decisión más allá de los problemas de portabilidad para elegir herramientas de desarrollo de aplicaciones multiplataforma adecuadas. Se identificaron los requisitos deseables en un marco de trabajo multiplataforma y se discutió sobre una arquitectura general para el desarrollo de aplicaciones multiplataforma. Se revisaron varias herramientas como PhoneGap, Titanium y Sencha Touch y proporcionaron una clasificación y comparación entre las herramientas. Se examinó el rendimiento en términos de CPU, uso de memoria y consumo de energía, y se encontró que PhoneGap consume menos memoria, CPU y energía, ya que no incluye componentes de interfaz de usuario dedicados.

Por otra parte, Umuhoza et al. (2015) presentó los resultados de un estudio de investigación realizado para encontrar la mejor estrategia para WebRatio, una empresa de desarrollo de software, interesada en producir una herramienta MDD para diseñar y desarrollar aplicaciones para entrar en el mercado de aplicaciones para dispositivos móviles. Además, presentaron un estudio comparativo realizado para identificar los mejores equilibrios entre varios enfoques de generación de código automático.

De manera similar, Benouda et al. (2016) presentó una metodología basada en MDA para desarrollar aplicaciones para dispositivos móviles según el principio "desarrollar una vez, usar en todas partes". Este enfoque explora el modelado UML y Acceleo para generar código específico con el fin de acelerar y facilitar el desarrollo de aplicaciones para dispositivos móviles. La herramienta Acceleo se utilizó para generar el código de Android™. La generación de código de Android™ se basó en diagramas de clases, que son utilizados por Acceleo para generar la estructura de la aplicación. La relación entre interfaces o clases, como asociación y herencia, se respetan durante la generación del código.

Por otra parte Satyavathy, Priya & Chanthini (2017) mencionaron que los desarrolladores de software hacen una elección entre el tipo de desarrollo de las aplicaciones que desean hacer entre desarrollos híbridos, nativos o aplicaciones Web. Los autores realizaron una comparación entre estos tres tipos o enfoques de desarrollo, también discutieron las características y desafíos del desarrollo de aplicaciones para dispositivos móviles, así como las ventajas y desventajas de cada enfoque.

Finalmente, Colombo-Mendoza et al. (2013) presentó AlexandRIA una herramienta visual que implementa un enfoque basado en patrones GUI para la generación de código de RIAs para múltiples dispositivos, esta herramienta visual llamada AlexandRIA es un generador de código fuente y nativo para el desarrollo de aplicaciones rápidas (RAD), que genera automáticamente código basado en un conjunto de preferencias seleccionadas a través de un asistente. Así también presentaron una evaluación cualitativa / cuantitativa para precisar la legitimidad de la propuesta frente a otras propuestas académicas y comerciales similares.

Para comprender mejor el uso de herramientas para el desarrollo de aplicaciones mencionadas anteriormente y visualizar sus propiedades de una manera fácil, se presenta la Tabla 2.2 a continuación.

Tabla 2.2. Comparativa de las herramientas de desarrollo de aplicaciones

Autores	Enfoque de desarrollo de software / Técnica	Parámetros de entrada	Parámetros de salida	Multiplataforma /Multidispositivo
Yang, Prasad y Xie (2013)	GUI	El conjunto de eventos soportados por la GUI de la aplicación	Un modelo de una aplicación para dispositivos móviles dada	Android™
Amalfitano et al. (2012)	GUI	Una GUI	Resultados de la prueba	Android™
Hu y Neamtiu (2011)	GUI	Una GUI	Resultados de la prueba	Android™
Dinh et al. (2013)	MCC	NA	NA	NE
Mao (2016)	QoE	Código fuente de la aplicación	Resultados de la prueba	Si/Si
Amalfitano et al. (2015)	Pruebas basadas en GUI de las aplicaciones de Android™	Una GUI	Un modelo de máquina de estado escalable	Android™
Zhu et al. (2015)	Popularidad HMM (PHMM)	Una aplicación para dispositivos móviles	Resultados de la prueba	NE
Acerbis et al. (2015)	MDD / IFML (Interaction Flow Modeling Language).	Modelo de dominio y modelo de interacción para aplicaciones para dispositivos móviles	Aplicaciones para dispositivos móviles multiplataforma usando el marco de trabajo Apache Cordova	Si/Si
Choi, Yang y Jeong (2009)	MDD/ PIM (Platform Independent Model)	Reglas de transformación para cada plataforma para dispositivos móviles	Código fuente Java para plataformas para dispositivos móviles	Si/NE

Autores	Enfoque de desarrollo de software / Técnica	Parámetros de entrada	Parámetros de salida	Multiplataforma /Multidispositivo
Miravet et al. (2009)	DIMAG	Descripción declarativa de la aplicación	Aplicaciones para dispositivos móviles	Si/Si
Cimitile, Risi y Tortora (2011)	Basado en el patrón de diseño MVC	Descripción de estructura y aspectos dinámicos de una aplicaciones para dispositivos móviles	Aplicaciones para dispositivos móviles	Si/Si
Benouda et al. (2016)	MDA	Modelado UML y Herramienta Acceleo	Código fuente	Si/Si
Colombo-Mendoza et al. (2013)	PPRD	Configuración	Código fuente	Si/Si

NE: No Especificado
NA: No Aplicable

Como conclusiones principales, no todas son herramientas de desarrollo de aplicaciones multiplataforma, ya que la mayoría se enfocan en una sola plataforma, generalmente Android™. Además, se encontró que las pocas herramientas multiplataforma utilizan marcos de trabajo para empaquetar el código y generar aplicaciones híbridas en lugar de aplicaciones nativas. Además, todas estas propuestas requieren experiencia y conocimientos previos, y el proceso de desarrollo de aplicaciones toma mucho tiempo. Tales desventajas evidencian la necesidad de un enfoque ágil e intuitivo para el desarrollo automático de aplicaciones para dispositivos móviles multiplataforma que sea amigable incluso para el usuario no experimentado.

Capítulo 3: Metodología

Este capítulo presenta la metodología propuesta para el desarrollo de la presente tesis doctoral la cual se basa en el método científico inductivo y se plasma en 6 etapas. El seguimiento de cada una de estas etapas permite llegar a la solución del problema propuesto. A continuación se presenta en la Figura 3.1, el proceso para el desarrollo del proyecto, donde se observa que cada etapa iterativa y secuencial.

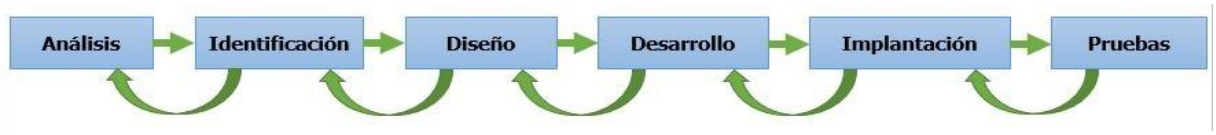


Figura 3.1. Etapas del proceso de desarrollo del proyecto

3.1. Descripción de la metodología propuesta

En esta sección se presentan de manera detallada el conjunto de actividades que se realizaron en cada etapa del proceso de desarrollo de la tesis. El proceso de desarrollo está compuesto de 6 etapas, las cuales se describen a continuación.

3.1.1. Análisis

Durante esta etapa se realizó la revisión del estado del arte de las técnicas de análisis y procesamiento de imágenes digitales así como la detección de patrones. También se realizó una revisión de sobre las redes neuronales artificiales para complementar el proceso de identificación de elementos en imágenes no ideales. Así también se realizó la revisión de los patrones de diseño de interfaces de usuario aplicados a la generación de aplicaciones. Resultado de dichas revisiones se generó un análisis comparativo tanto de las técnicas de análisis y procesamiento de imágenes digitales y los patrones de diseño de interfaces de usuario, lo que permitió realizar un aporte científico totalmente innovador con las mejores técnicas detectadas tras la revisión y análisis de la literatura.

Actividades

1) Análisis del conjunto de algoritmos, técnicas y bibliotecas de funciones para procesamiento de imágenes y reconocimiento de patrones.

2) Estudio comparativo de los algoritmos, técnicas y bibliotecas de funciones estudiadas previamente.
3) Identificación de los tipos de imágenes soportados y las restricciones de entrada y salida de cada uno de los algoritmos
4) Elaboración de un análisis comparativo de los algoritmos para el procesamiento de imágenes digitales.
5) Análisis de las redes neuronales artificiales
6) Análisis de los distintos patrones de diseño de interfaces de usuario aplicados en el desarrollo de aplicaciones.

3.1.2. Identificación

En esta etapa se revisaron los algoritmos, redes neuronales artificiales y patrones de diseño de interfaces de usuario identificados en la etapa anterior con el objetivo de identificar los aspectos importantes en el área del procesamiento de imágenes así como los patrones de diseño de interfaces de usuario, y así ofrecer mejores soluciones en términos de eficiencia, consumo de recursos computacionales, eficiencia en la identificación de elementos en imágenes no ideales y otros factores decisivos al momento de hablar de generación automática de software.

Actividades

1) Identificación de factores importantes para la selección de las técnicas, algoritmos y patrones de diseño de interfaces de usuario, tales como: menor consumo de recursos computacionales, mayor eficiencia, menor cantidad de restricciones por mencionar algunos, en los algoritmos de procesamiento de imágenes para la generación de software; para la selección de patrones de diseño de interfaces de usuario su índice de usabilidad y la posibilidad de aceptar otros componentes, entre otros.

3.1.3. Diseño

Con el objetivo de disminuir tiempo y costos durante el proceso de desarrollo y de facilitar las etapas de diseño e implementación del ciclo de vida del software se llevó a cabo el diseño de nuevos métodos computacionales para el reconocimiento de patrones en imágenes para la generación de software multidispositivo y multiplataforma a partir del uso de patrones de

diseño de interfaces de usuario y redes neuronales. Así como el diseño de la arquitectura que permitió integrar la herramienta generadora de código.

Actividades

1) Diseño de los algoritmos para la identificación de patrones en imágenes digitales, esto se llevó a cabo combinando diferentes técnicas y algoritmos para conformar los métodos computacionales.
2) Diseño de los métodos computacionales para la generación automática de software.
3) Diseño de la arquitectura para la integración de la herramienta generadora de código fuente de las aplicaciones.

3.1.4. Desarrollo

Con la finalidad de automatizar parte del proceso de desarrollo de software, en esta etapa se implementaron los métodos computacionales para la generación automática de software multidispositivo y multiplataforma. Así como el desarrollo de los módulos que conforman la arquitectura diseñada en la etapa 3.

Actividades

1) Codificación de los algoritmos propuestos como parte de los nuevos métodos computacionales
2) Desarrollo de los módulos de software necesarios para la integración de la herramienta generadora de las aplicaciones multidispositivo y multiplataforma.

3.1.5. Implantación

Durante esta etapa se implantó el software desarrollado en la etapa 4 como prueba de concepto para los métodos computacionales propuestos en la etapa de diseño. En esta etapa también se diseñaron los casos de estudio a los que se sometieron los métodos computacionales propuestos, y que se utilizaron en la etapa 6 de pruebas.

Actividades

1) Implantación e integración de los métodos computacionales desarrollados en la etapa de desarrollo
2) Diseño del caso de estudio con el que se pusieron a prueba los métodos desarrollados

3.1.6. Pruebas

Es importante evaluar los resultados obtenidos derivados de la utilización de los métodos computacionales existentes en comparación con los resultados obtenidos empleando los métodos computacionales desarrollados en este proyecto. En este sentido durante esta etapa se realizaron diversas pruebas a los métodos computacionales propuestos utilizando diversos casos de estudio para validar los resultados obtenidos.

Actividades

1) Pruebas de unidad sobre los módulos de generación automática de software
2) Pruebas de benchmarking sobre los métodos computacionales propuestos

3.2. Arquitectura de la herramienta ImagIng

La herramienta ImagIng es una herramienta basada en Web, la interfaz ImagIng se desarrolla en HTML5, CSS y JavaScript. En la siguiente sección se describe la arquitectura de la herramienta ImagIng. La arquitectura general se muestra en la Figura 3.2.

3.2.1. Descripción de la arquitectura

La herramienta ImagIng tiene un diseño en capas para organizar sus componentes. Este diseño en capas permite la escalabilidad y el mantenimiento fácil, ya que sus tareas y responsabilidades se distribuyen. Cada componente tiene una función que se explica a detalle a continuación:

Capa de datos. Esta capa está formada por el gestor de base de datos que es responsable del almacenamiento de información, como plantillas, las plataformas soportadas, la información de configuración de módulos y servicios, entre otros. La capa de datos es responsable de recibir las solicitudes de recuperación de información de la capa de gestión de datos.

Capa de acceso a datos. Un sistema de gestión de base de datos se encuentra en esta capa que mantiene la persistencia de datos ejecutando operaciones de inserción, actualización, supresión y consulta dentro de la arquitectura de ImagIng. Estas operaciones están encapsuladas en esta capa para proporcionar seguridad a las capas superiores y evitar atajos en el sistema de gestión de bases de datos.

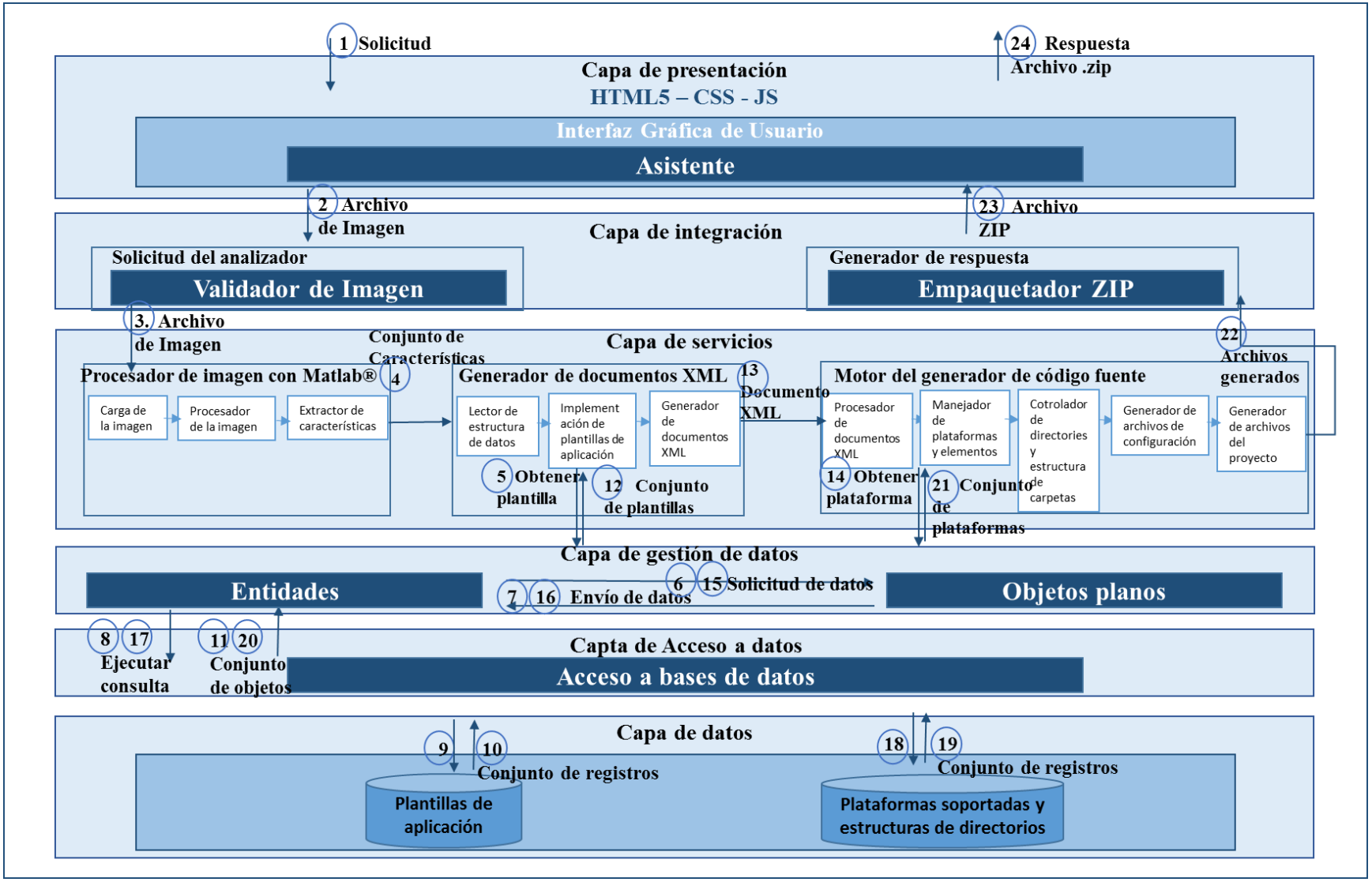


Figura 3.2. Arquitectura de la herramienta ImagIng

Capa de gestión de datos. Esta capa se comunica con la capa de acceso a datos y contiene la representación de objetos planos de cada relación del sistema de gestión de bases de datos. Además, proporciona una serie de entidades con operaciones de lectura / escritura para cada relación que se ejecuta utilizando la capa de acceso a datos.

Capa de servicios. Esta capa proporciona un conjunto de servicios (módulos) ofrecidos por la herramienta ImagIng. Estos servicios representan diferentes funciones como: (1) procesamiento de imágenes; (2) Generación de documentos XML; Y (3) generación de código fuente. Esta capa tiene acceso a la capa de integración que recibe una imagen de entrada y devuelve un conjunto de archivos generados, también se comunica con la capa de gestión de datos en la que pide información de las bases de datos.

Capa de integración. Esta capa contiene los componentes necesarios para la ubicación de los servicios solicitados a través de la interfaz gráfica de usuario. La capa de integración facilita la creación de aplicaciones al acceder a todas las interfaces públicas que proporcionan la herramienta ImagIng.

Capa de presentación. En esta capa, ImagIng herramienta mostrar el contenido mediante HTML5. La capa de presentación no sabe qué eventos están ocurriendo dentro de ImagIng y cómo se proporcionan los servicios, sino que sólo los utiliza para mostrar la interfaz de usuario final.

De acuerdo con el énfasis en la automatización, la arquitectura de ImagIng se accede usando un asistente. El asistente está dirigido a desarrolladores de sistemas inexpertos. Este asistente guía a los usuarios a través de una serie de pasos para desarrollar las interfaces del usuario para una aplicación.

3.2.2. Descripción de los componentes

Cada capa de la arquitectura de ImagIng contiene un conjunto de componentes. Las funcionalidades de estos componentes se describen en los siguientes párrafos.

Componente *GUI*. Este componente es responsable de procesar las solicitudes de usuario recibidas por el protocolo basado en HTTP. El componente GUI gestiona los eventos y valida los datos de entrada del usuario. El componente GUI incluye un módulo:

1) Asistente de aplicaciones: Este módulo proporciona un asistente que guía al usuario paso a paso en el proceso de desarrollo de una aplicación de varios dispositivos. Este asistente comienza con la imagen de carga y termina cuando se descarga el archivo ZIP.

Analizador de solicitudes. Este componente recibe las solicitudes de los usuarios, las analiza y las envía al componente del selector de servicios. En el caso específico de la creación de aplicaciones mediante el uso de imágenes, este componente es responsable de validar la imagen cargada.

Generador de respuestas. Este componente recibe la respuesta del sistema y es responsable de empaquetar los archivos generados.

Procesador de imágenes. Este módulo es responsable del procesamiento de la imagen, para este trabajo el componente se divide en tres pasos y se desarrolló con MATLAB®. El componente Procesador de imágenes incluye tres módulos:

1) Cargador de imagen: Es responsable de cargar la imagen y prepararla para su procesamiento.
2) Procesador de imagen: Es responsable de aplicar los métodos y técnicas de procesamiento de imágenes para extraer la información deseada en la imagen.
3) Extracción de características: Después de aplicar el procesamiento de imágenes, este módulo es responsable de almacenar la información extraída de todas las características de imagen

Generador de documentos XML. Este módulo construye un documento basado en XML en el que se describen las propiedades básicas y la configuración de implementación de una aplicación para cada tecnología multidispositivo. Este documento basado en XML es un manifiesto o un archivo de configuración donde se incluye la siguiente información: a) paths; b) dependencias de bibliotecas; c) permisos de acceso; d) nombre de la solicitud; e) icono de aplicación; f) versión; Y g) autor. El componente del generador de documentos XML incluye tres módulos:

1) Lector de estructura de datos: Este módulo es responsable de leer la información proveniente del componente Procesador de imágenes.
2) Implementación de plantillas de aplicación: Según la información recibida recuperar las plantillas almacenadas en la base de datos.

3) Generador de documentos basado en XML: Basándose en la plantilla recuperada, se genera el documento XML correspondiente.

Motor de generación de código fuente. Este componente es responsable de administrar el conjunto de contenidos entregados a través de ImagIng y generar una aplicación nativa a través de cualquiera de las tecnologías soportadas para la generación de aplicaciones multidispositivo. Este componente tiene los siguientes módulos, como se explica a continuación:

1) Procesador de documento XML: Este módulo es responsable de procesar el documento XML que viene del componente generador de documentos XML.
2) Controlador de plataformas y elementos: Una vez procesada la información del documento XML, se recupera la información necesaria de la base de datos, esta información se vincula directamente a la plataforma de software elegida para la generación del código fuente.
3) Controlador de directorios y estructura de carpetas: Se determinan los directorios y la estructura de carpetas de la aplicación generada con base en la plataforma elegida.
4) Generador de archivos de configuración: En este módulo se generan los archivos de configuración necesarios para cada plataforma.
5) Generador de archivos del proyecto: En este módulo se generan los archivos necesarios por plataforma y aplicación.

Objetos Planos. Este módulo es responsable de la representación a través de los objetos de datos utilizados por ImagIng. Un objeto plano es un objeto que no tiene lógica de aplicación y se utiliza sólo para representar datos como un POJO (Plain Old Java Object).

Entidades. Este componente incluye las entidades de la base de datos de ImagIng, que separan las operaciones CRUD de su representación en el nivel de objeto. CRUD significa Create Read Update Delete (Crear, Leer, Actualizar, Eliminar) utilizado para referirse a las funciones básicas de la base de datos o la capa de persistencia de un sistema de software.

Acceso a la base de datos. Este componente incluye los controladores necesarios para conectarse a las bases de datos, y también las funciones para conectarse y desconectarse con la base de datos, por mencionar algunos.

Una vez definidos los componentes de Imaging, el flujo de trabajo se describirá para aclarar más el proceso de generación de código basado en imágenes.

3.3. Flujo de trabajo

El flujo de trabajo describe la funcionalidad de la arquitectura de ImagIng, como se describe a continuación:

1. La capa de presentación identifica la solicitud del usuario y redirige la solicitud basada en HTTP a la capa de integración en la que se analiza la solicitud basada en HTTP para determinar qué tipo de servicio se solicita. En este paso se carga la imagen en el paso 1 del asistente de ImagIng,
2. En la capa de integración se determina si la solicitud HTTP es válida, si el servicio está disponible o no se ha solicitado una solicitud de servicio. En este punto la imagen se valida, la validación se realiza en cuanto a formato (JPG, PNG y GIF) tamaño de imagen en píxeles (se recomienda usar imágenes desde 400x200 píxeles), y el tamaño del archivo (se recomienda no utilizar imágenes de más de 1 MB).
3. Si se ha solicitado una solicitud de un servicio y la imagen es válida, se realiza el procesamiento de la imagen, para realizar el procesamiento de la imagen se utiliza el procesador de imagen mediante el módulo MATLAB® en este módulo se establece un conjunto de algoritmos, métodos y reglas que se explican a continuación: Para este proceso el primer paso es cargar la imagen, luego se aplica un filtro para convertir la imagen en escala de grises, la imagen se segmenta para obtener sus bordes horizontales y verticales, después se aplican operaciones morfológicas para eliminar el ruido y obtener los bordes afilados (esto se aplica tanto al borde horizontal como al borde vertical) y luego una vez obtenidos los dos bordes (horizontales y verticales), entonces se realiza un etiquetado por elementos, cada borde de cada elemento obtenido se marca para identificar cada elemento y obtener sus características, las características obtenidas de cada elemento se almacenan en una matriz, y la matriz se recorre para recuperar las características de cada elemento de forma independiente, después de haber identificado los elementos, se realiza el proceso de segmentación de cada elemento de la imagen original para procesarla por separado, se obtienen las coordenadas de cada elemento y se valida que las coordenadas no pertenecen a otro elemento, si se cumple la condición se almacena el recorte del, sus coordenadas y los valores RGB de los píxeles contenidos en la imagen. Finalmente, una vez que todos

los elementos están segmentados, las reglas se aplican para identificar cada elemento. Después de procesar todos los elementos segmentados, para identificar aquellos que cumplen las reglas y almacenar la información relevante, se genera un archivo de texto (.txt) con esta información.

4. El archivo de texto generado en el paso 3 se pone a disposición del módulo de generación de documentos XML, en el cual se realiza la lectura del archivo de texto para extraer los datos contenidos en él.
5. De acuerdo con la información contenida en el archivo de texto, el módulo de generación de documentos XML realiza la solicitud de las plantillas correspondientes a la capa de gestión de datos.
6. El módulo de entidades contiene la lógica de manipulación de datos y por lo tanto requieren una solicitud de los datos almacenados en el objeto plano correspondiente.
7. Mediante el uso de un método de acceso, los objetos planos recuperan los datos y los envían de nuevo a la entidad que los manipula.
8. Para lograr esta manipulación de datos, la entidad ejecuta una consulta a través de la capa de acceso a datos para que los datos obtenidos de la ejecución de la consulta cambien el valor de cada objeto plano requerido.
9. La conexión a la capa de datos se consigue a través del PDO con soporte nativo de PHP para acceder a la gestión de bases de datos o con ODBC y se ejecuta una consulta proporcionada por la entidad.
10. La administración de base de datos devuelve un conjunto de registros (*RecordSet)* con datos obtenidos de la capa de acceso a datos. Devuelve las plantillas solicitadas.
11. La capa de acceso a datos transforma el objeto *RecordSet* en una matriz de objetos planos. Esta matriz se devuelve a las entidades correspondientes de la capa de gestión de datos.
12. Las entidades se envían como objetos (instancias de clase) a la capa de servicio para su integración con la plantilla correspondiente. En este punto, se ha generado un documento basado en XML.
13. Se procesa el documento XML dentro del módulo Motor de Generación de Código Fuente, en este paso se identifica la plataforma para la que se generará el código fuente.

14. Según el documento XML se solicita la información correspondiente de la plataforma a la capa de gestión de datos.
15. El módulo de entidades contiene la lógica de manipulación de datos y por lo tanto requieren una petición para los datos almacenados en el objeto plano correspondiente.
16. Mediante el uso de un método de acceso, los objetos planos recuperan los datos y los envían de nuevo a la entidad que los va a manipular.
17. Para lograr esta manipulación de datos, la entidad ejecuta una consulta a través de la capa de acceso a datos para que los datos obtenidos de la ejecución de la consulta cambien el valor de cada objeto plano requerido.
18. La conexión a la capa de datos se consigue a través del PDO con soporte nativo de PHP para acceder a la gestión de la base de datos o con ODBC y se ejecuta una consulta proporcionada por la entidad.
19. La administración de la base de datos devuelve un conjunto de registros (*RecordSet*) con datos obtenidos de la capa de acceso a datos. Se devuelve la información de la plataforma.
20. La capa de acceso a datos transforma el objeto *RecordSet* en una matriz de objetos planos. Esta matriz se devuelve a las entidades correspondientes de la capa de gestión de datos.
21. Las entidades se envían como objetos (instancias de clase) a la capa de servicio para su integración con la plantilla correspondiente. Una vez recuperada la información relevante para la plataforma, esta información se almacena y pasa al módulo manejador de directorios, también se crean las carpetas necesarias de acuerdo con la estructura de directorios, se recuperan los archivos de configuración necesarios y finalmente se generan los archivos con el código fuente correspondiente.
22. Una vez que se ha generado la aplicación nativa, la aplicación se envía al módulo Empaquetador ZIP para que se empaquete con el formato y la extensión de archivo correspondiente.
23. La aplicación empaquetada en un archivo .zip se envía al componente Generador de respuesta que es responsable de procesar la información y enviar el archivo .zip al usuario.
24. Finalmente, se entrega al usuario el archivo .zip.

3.4. Proceso de identificación de elementos en imágenes

En principio se define un conjunto de reglas para identificar cada elemento de la interfaz. El propósito de definir el conjunto de reglas se basa en el hecho de tener un conjunto de especificaciones para cada elemento o PDIU que sirven como identificadores únicos en el momento del procesamiento de imágenes, tales reglas se representan en un árbol que usa el lenguaje RuleML versión 1.0. Es una iniciativa abierta que busca establecer un sistema de reglas de inferencia lógica a partir de ontologías y documentos RDF con su propia especificación y ejecución lingüística (Boley, Paschke and Shafiq 2010). Se generaron un total de 5 reglas para los siguientes patrones de diseño de interfaz de usuario: *video, carousel, dashboard, login* y *splashscreen*, cada patrón de diseño de interfaz de usuario o PDIU se identifica como un elemento. En la Figura 3.3 se muestra un fragmento de un archivo basado en XML generado para representar reglas, donde se presenta la regla para el patrón de inicio de sesión.

```
<Implies>
    <If>
        <And>
            <Atom>
                <Var>Shape external</Var>
                <Rel>Rectangle</Rel>
            </Atom>
            <Atom>
                <Var>Elements</Var>
                <Rel>Interface elements</Rel>
                <Ind>2 input text </Ind>
                <Ind>1 button </Ind>
            </Atom>
        </And>
    </If>
    <Then>
            <Atom>
                <Var>Shape external</Var>
                <Var>Elements</Var>
                <Rel>Login</Rel>
            </Atom>
    </Then>
</Implies>
```

Figura 3.3. Regla para el PDIU: Inicio de sesión (*login*)

En la Figura 3.4, se muestra una representación parcial del árbol que describe la regla definida para el patrón de diseño de UI de inicio de sesión.

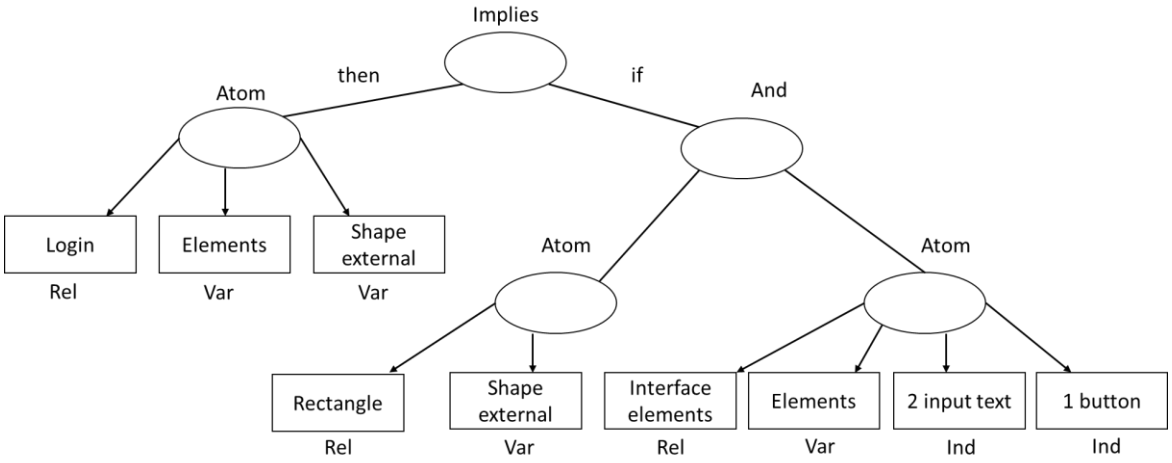


Figura 3.4. Representación parcial del árbol de la regla para el patrón de diseño de interfaz de usuario de inicio de sesión

En la Figura 3.5 se muestra un fragmento de un archivo basado en XML que representa la regla para el patrón Splashscreen.

Los árboles de reglas definidos en *RuleML* sirvieron de guía para la implementación en un módulo generado con MATLAB® (*Matrix Laboratory*) (Guide 1998), un software matemático que proporciona una biblioteca de funciones: *Image Processing Toolbox*™ para el procesamiento de imágenes. Las herramientas de procesamiento de imágenes de MATLAB® proporcionan un conjunto completo de algoritmos y herramientas gráficas para el procesamiento, análisis y visualización de imágenes digitales. La estructura general del módulo consiste en un conjunto de funciones para segmentar primero la imagen de entrada y obtener la posición de cada elemento que es parte de la imagen, otra función obtiene para cada elemento un conjunto de firmas dependiendo de las figuras que integran cada una de los elementos definidos. Otra función, obtiene el número de objetos encontrados según las firmas obtenidas por la función anterior. Con estas funciones es posible identificar qué elementos están en la imagen de entrada.

```
<Implies>
    <If>
        <And>
            <Atom>
                <Var>Shape external</Var>
                <Rel>Rectangle</Rel>
            </Atom>
            <Atom>
                <Var>Elements</Var>
                <Rel>Interface elements</Rel>
                <Ind>Dark shaded rectangle</Ind>
                <Ind>Shaded rectangle</Ind>
            </Atom>
        </And>
    </If>
    <Then>
            <Atom>
                <Var>Shape external</Var>
                <Var>Elements</Var>
                <Rel>Splashscreen</Rel>
            </Atom>
    </Then>
</Implies>
```

Figura 3.5. Regla para el PDIU: Splashscreen

En la Figura 3.6 se muestra un fragmento de un archivo basado en XML que representa la regla para el patrón Dashboard.

```
<Implies>
    <If>
        <And>
            <Atom>
                <Var>Shape external</Var>
                <Rel>Rectangle</Rel>
            </Atom>
            <Atom>
                <Var>Elements</Var>
                <Rel>Interface elements</Rel>
                <Ind>Rectangle</Ind>
                <Ind>3 rectangles inside a rectangle</Ind>
                <Ind>Rectangle</Ind>
            </Atom>
        </And>
    </If>
    <Then>
            <Atom>
                <Var>Shape external</Var>
                <Var>Elements</Var>
                <Rel>Dashboard</Rel>
            </Atom>
    </Then>
</Implies>
```

Figura 3.6. Regla para el PDIU: Dashboard

En la Figura 3.7 se muestra un fragmento de un archivo basado en XML que representa la regla para el PDIU Carousel.

```
<Implies>
    <If>
        <And>
            <Atom>
                <Var>Shape external</Var>
                <Rel>Rectangle</Rel>
            </Atom>
            <Atom>
                <Var>Elements</Var>
                <Rel>Interface elements</Rel>
                <Ind>2 Dark shaded rectangle</Ind>
                <Ind>2 Shaded rectangle inside</Ind>
            </Atom>
        </And>
    </If>
    <Then>
            <Atom>
                <Var>Shape external</Var>
                <Var>Elements</Var>
                <Rel>Carousel</Rel>
            </Atom>
    </Then>
</Implies>
```

Figura 3.7. Regla para el PDIU: Carousel

En la Figura 3.8 se muestra un fragmento de un archivo basado en XML que representa la regla para el PDIU Video.

```
<Implies>
    <If>
        <And>
            <Atom>
                <Var>Shape external</Var>
                <Rel>Rectangle</Rel>
            </Atom>
            <Atom>
                <Var>Elements</Var>
                <Rel>Interface elements</Rel>
                <Ind>Rectangle</Ind>
                <Ind>Rectangle inside a rectangle</Ind>
                <Ind>Triangle inside a rectangle</Ind>
            </Atom>
            <Atom>
                <Var>Elements</Var>
                <Rel>Interface elements</Rel>
                <Ind>Rectangle</Ind>
                <Ind>Triangle inside a rectangle</Ind>
                <Ind>Rectangle inside a rectangle</Ind>
            </Atom>
        </And>
    </If>
    <Then>
            <Atom>
                <Var>Shape external</Var>
                <Var>Elements</Var>
                <Rel>Video</Rel>
            </Atom>
    </Then>
</Implies>
```

Figura 3.8. Regla para el PDIU: Video

Las principales funciones y algoritmos utilizados para el procesamiento de imágenes con MATLAB® son:

1) **bwlabel**: es una función utilizada para etiquetar los píxeles de cada objeto únicamente para su análisis, así como algunas de sus propiedades como obtener el objeto de la imagen.
2) **bwmorph**: es una función para realizar algunas operaciones morfológicas en la imagen.
3) **imerode**: es una función utilizada para erosionar la imagen y obtener los bordes.
4) **rgb2gray**: es una función para convertir una imagen de color a escala de grises.

El esquema de funcionamiento del módulo comienza al obtener la interfaz ADV a procesar, una vez que se carga la imagen de la interfaz en memoria, se aplica un conjunto de operaciones morfológicas para obtener una imagen con sólo el contorno; Después de completar las operaciones anteriores, un proceso de segmentación comienza a identificar y obtener cada contenido de imagen, para cada elemento obtenido sus coordenadas se almacenan en una matriz. Después del proceso de segmentación, de acuerdo con las coordenadas almacenadas de la imagen original se obtiene un elemento recortado, que se envía a otra función que recibe un conjunto de firmas. La técnica de firma consiste en obtener la distancia del píxel centroide de una figura con límites de la misma figura, de 1 a 360 grados.

Cuando se ha obtenido la firma de todas las figuras que componen cada elemento, se almacenan en una matriz. Con la matriz de firma el número de figuras que componen el elemento se obtiene en otra función, esto se hace a través de la correlación cruzada entre la firma de cada figura a identificar y las firmas de todas las formas: círculos, rectángulos, cuadrados, triángulos, líneas, Y esa figura está asociada con un valor más alto que se aproxima más a la firma de cada figura, respectivamente. Lo siguiente es validar el número de componentes encontrados de acuerdo con la correlación para identificar qué elemento es, por ejemplo, un elemento que está formado por un rectángulo y una línea, entonces para validarlo se toman dos firmas, una de las cuales es la de un rectángulo y la otra perteneciente a una línea.

Cuando se ha identificado el elemento, algunas de sus características se almacenan en una matriz, como el formato de color RGB, el ancho, la altura, las posiciones (X, Y) y el nombre del elemento identificado. Si el elemento no está identificado, entonces no se almacenan datos en él. Después de los pasos anteriores, se genera un archivo .txt con datos de todos los elementos identificados. Este archivo contiene en cada línea y separado por comas características de cada elemento. Una vez hecho esto, el módulo finaliza el procesamiento de la imagen.

Representación de las interfaces de usuario a través de PDIUs

Cada imagen de entrada de la herramienta generadora de código fuente es una interfaz gráfica de usuario que se representa con un dibujo realizado a mano alzada, a su vez esta interfaz gráfica de usuario está conformada por patrones de diseño de interfaces de usuario (PDIUs). A continuación se muestra una representación de los PDIUs más representativos tanto de forma ideal (ADV con figuras geométricas bien trazadas) como no ideal (figuras hechas a mano alzada).

***Splashscreen*:** este patrón permite mostrar al usuario una pantalla personalizada al iniciar una aplicación (Neil, 2014).

En la figura 3.9 se muestra la representación del PDIU splashscreen no ideal y en la figura 3.10 se muestra la representación ideal.

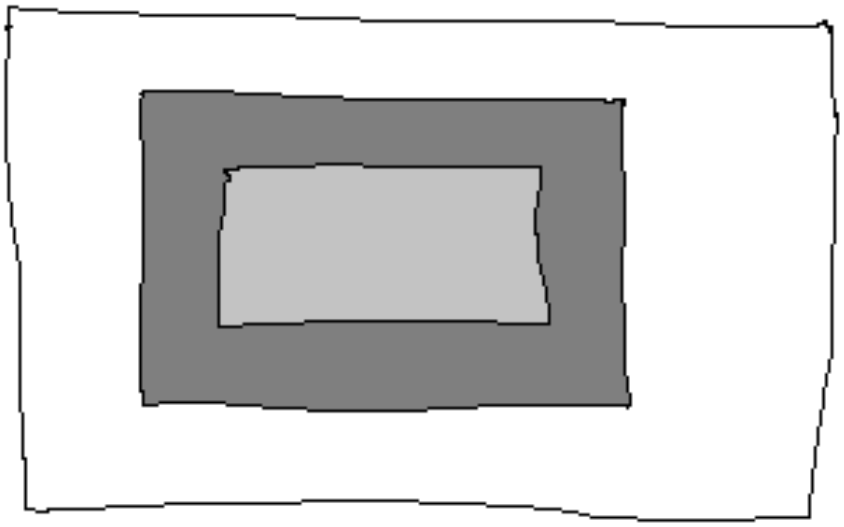


Figura 3.9. Ejemplo de *PDIU*: *Splashscreen* no ideal

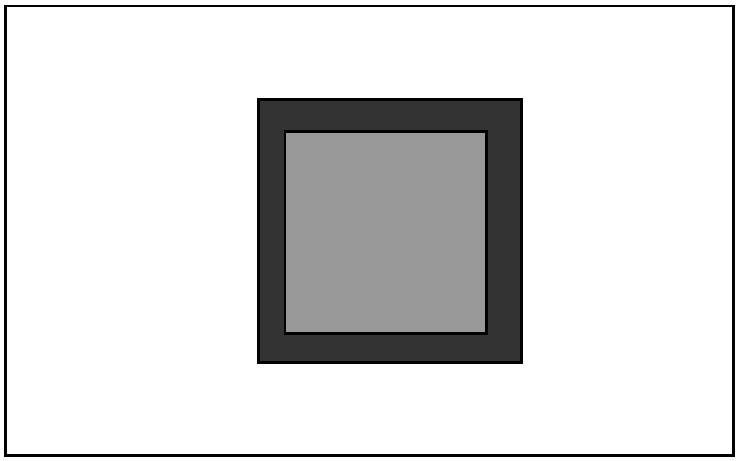


Figura 3.10. Ejemplo de *PDIU*: *Splashscreen* ideal

Ejemplos del uso del PDIU Spashscreen se observa en la figura 3.11, en la cual se muestra la imagen de inicio de las populares aplicaciones Twitter, Dropxbox y Facebook® en dispositivos móviles.

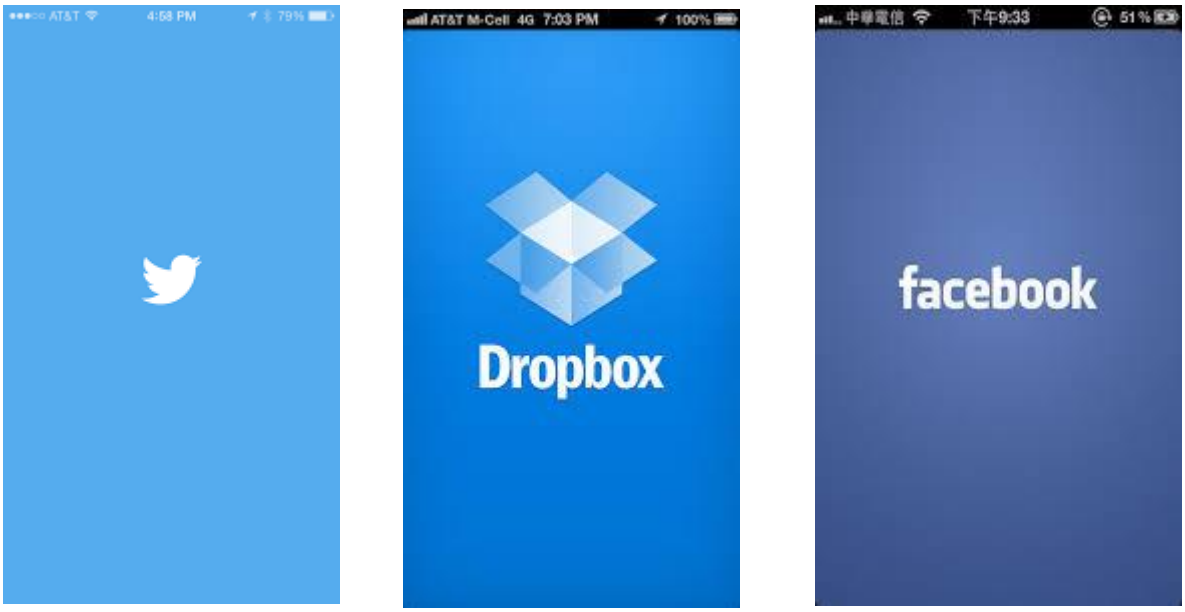


Figura 3.11. Ejemplo de uso del PDIU Splashscreen

***Login*:** patrón que permite controlar el acceso de usuarios a un área restringida (Neil, 2014).

En las figuras 3.12 y 3.13 se muestran las representaciones del PDIU Login de manera no ideal e ideal respectivamente.

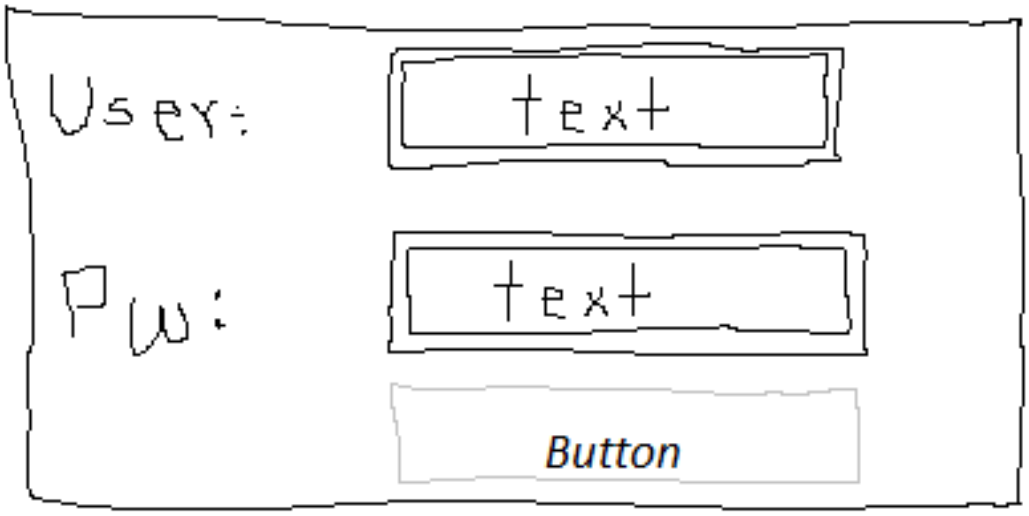


Figura 3.12. Ejemplo de *PDIU*: *Login* no ideal

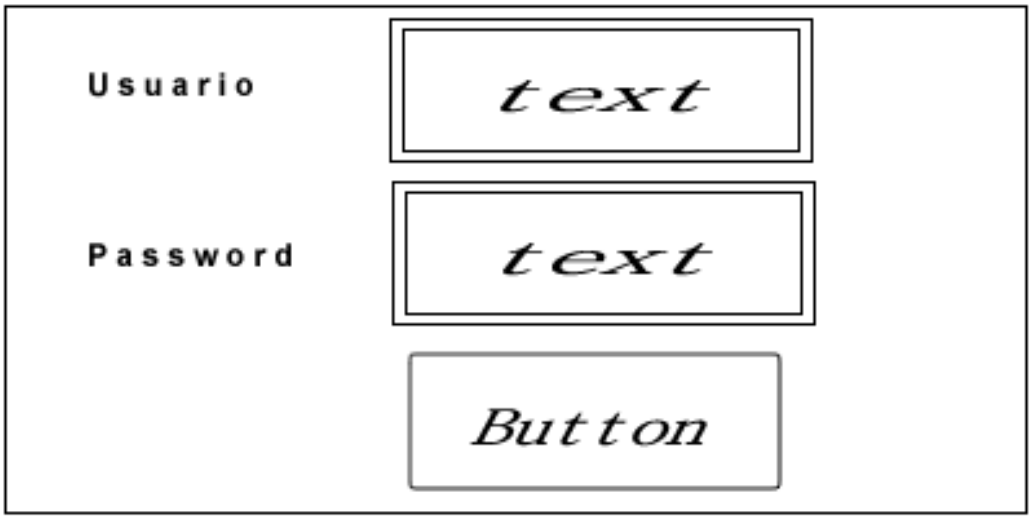


Figura 3.13. Ejemplo de *PDIU*: *Login* ideal

Ejemplos de uso del PDIU Login se observan en la figura 3.14, la primera interface corresponde al inicio de sesión en Twitter y la segunda al inicio de sesión para las cuentas de Microsoft®, en este caso para Outlook.

Figura 3.14. Ejemplo de uso del PDIU Login

***Video*:** Es un patrón de diseño ideal para incorporar contenido multimedia en una aplicación (Neil, 2014).

En la figura 3.15 se muestra la representación del PDIU Video de manera no ideal y en la figura 3. 16 de manera ideal.

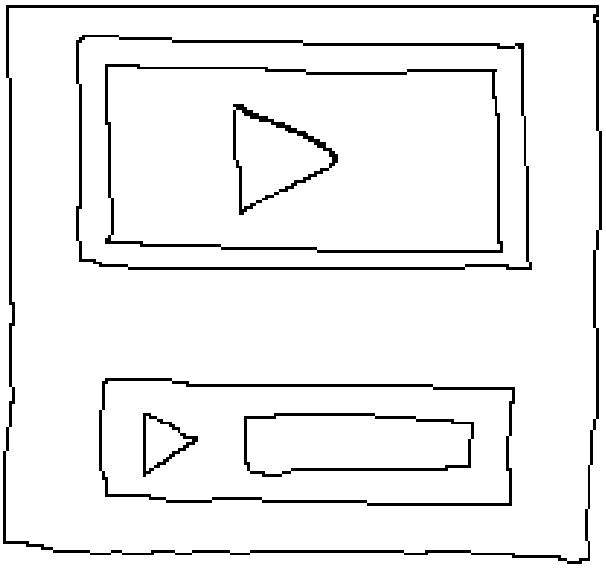


Figura 3.15. Ejemplo de *PDIU*: *Video* no ideal

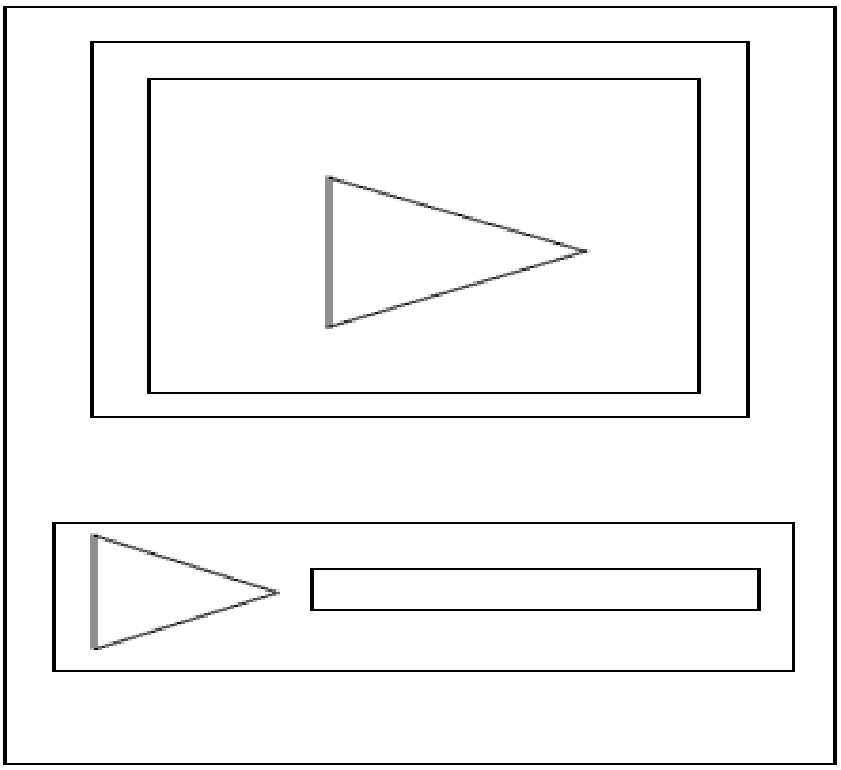


Figura 3.16. Ejemplo de *PDIU*: *Video* ideal

El ejemplo de uso del PDIU Video está representado por una de las más populares aplicaciones de reproducción de videos como es YouTube, en la figura 3.17 se observa la reproducción de un video en interface gráfica de esta aplicación.

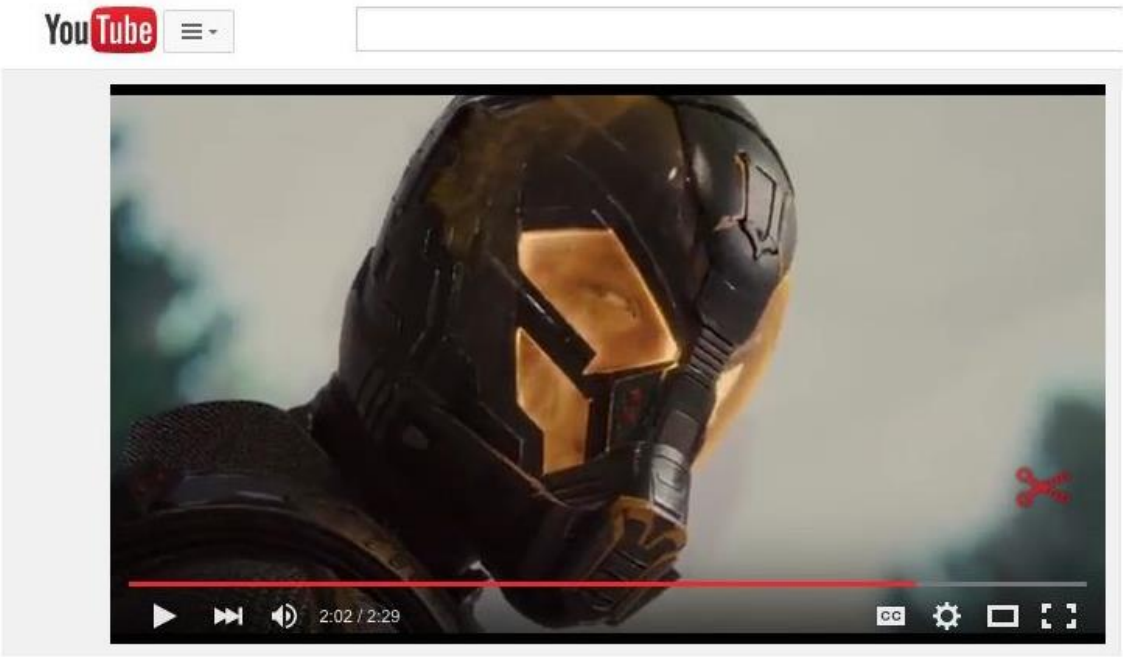


Figura 3.17. Ejemplo de uso del PDIU Video

***Dashboard*:** Permite al usuario visualizar de forma rápida el contenido más importante de una aplicación, sin la necesidad de navegar en otra pantalla (UI-Patterns, 2016).

En las figuras 3.18 y 3.19 se muestras las representaciones del PDIU Dashboard, en la primera de manera no ideal y en la segunda de manera ideal.

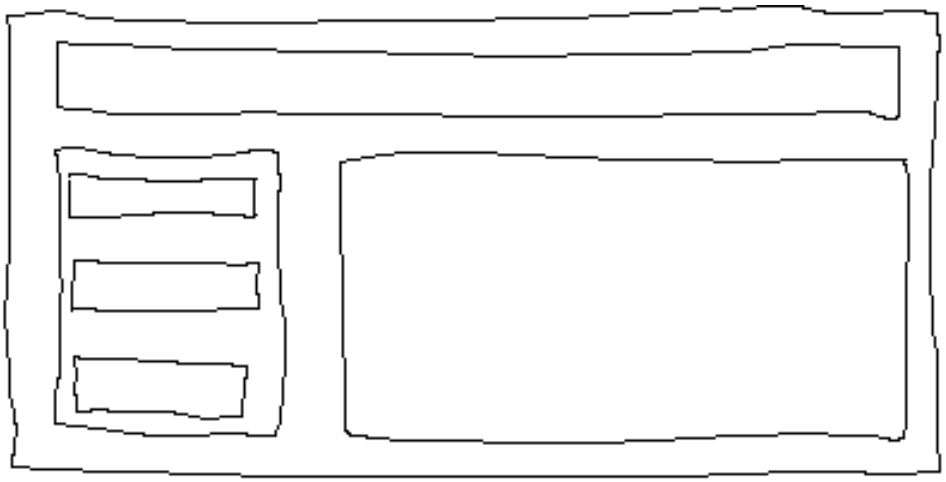


Figura 3.18. Ejemplo de *PDIU*: *Dashboard* no ideal

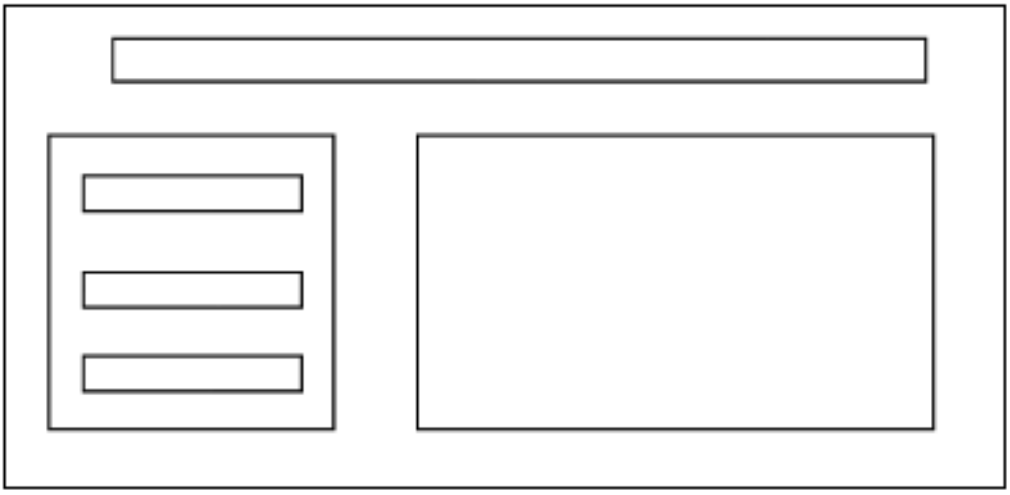


Figura 3.19. Ejemplo de *PDIU*: *Dashboard* ideal

El ejemplo de uso del PDIU Dashboard se observa en la figura 3.20, la primera interfaz corresponde al dashboard del menú de Windows 10 y la segunda interfaz corresponde al dashboard de la aplicación para dispositivos móviles de Twitter.

Figura 3.20. Ejemplo de uso del PDIU Dashboard

***Carousel*:** Este patrón se utiliza para navegar rápidamente por un conjunto de páginas o imágenes utilizando el gesto de desplazamiento (Neil, 2014; UI-Patterns, 2016).

En la figura 3.21 se muestra la representación del PDIU Carousel de manera no ideal y en la figura 3.22 la representación ideal.

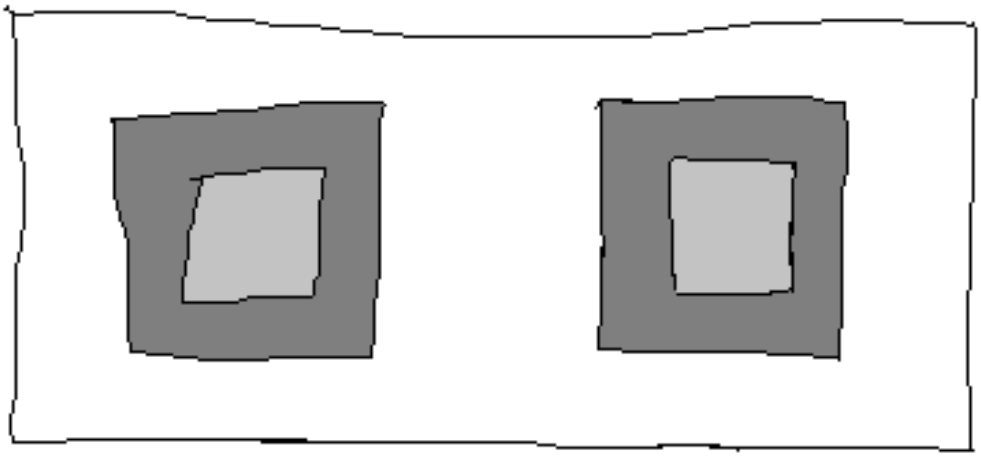


Figura 3.21. Ejemplo de *PDIU*: *Carousel* no ideal

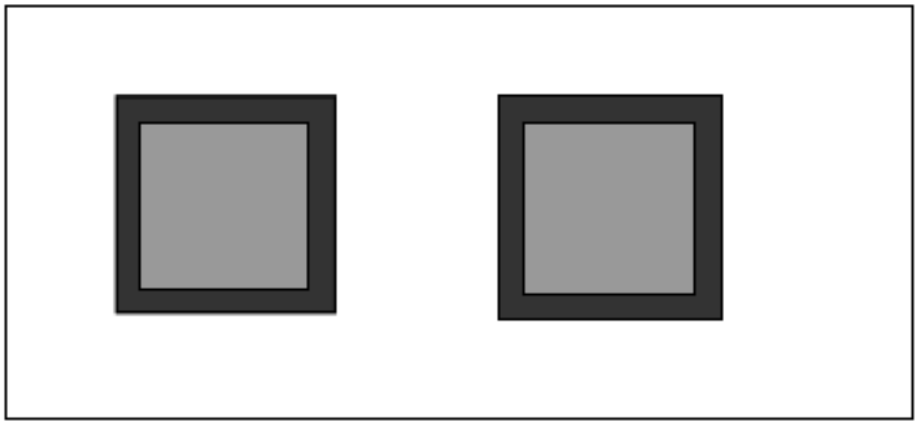


Figura 3.22. Ejemplo de *PDIU*: *Carousel* ideal

El ejemplo de uso del PDIU Carousel se observa en la figura 3.23, en esta figura se presenta un carrusel de imágenes de la famosa red social Facebook®.

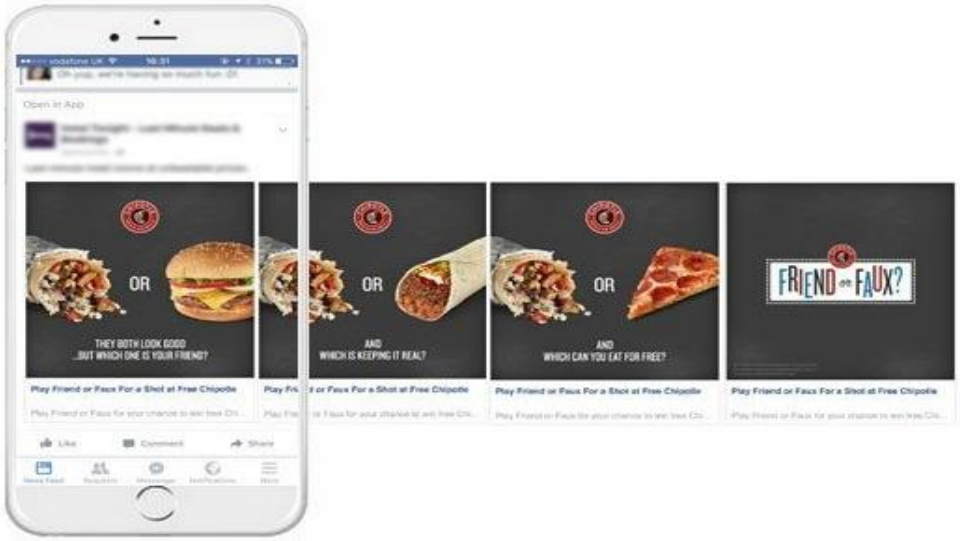


Figura 3.23. Ejemplo de uso del PDIU Carousel

A continuación se describe la implementación de la red neuronal artificial aplicada al procesamiento de imágenes y al reconocimiento de PDIUs dentro de las imágenes procesadas.

3.5. Implementación de una Red neuronal artificial

Las Redes Neuronales son un enfoque computacional que se basa en una gran colección de unidades neuronales modelando vagamente la forma en la que el cerebro resuelve problemas con grandes grupos de neuronas biológicas conectadas por axones. ImagIng considera la posibilidad de que las interfaces utilizadas no sean elementos ideales, ya que en cualquier momento el usuario ingresa imágenes generadas a mano alzada, por lo que es necesaria e importante la adición de una red neural para identificar elementos no ideales que representen interfaces de usuario. Dado que las redes neuronales se basan en cómo funciona el cerebro humano, resulta ser muy eficiente cuando se trata de resolver problemas que implican análisis y conocimientos previos, por ejemplo, en una imagen tridimensional proyectada con varias figuras, las redes neuronales identifican las diferentes figuras a través de la visión, haciendo uso de experiencias, pistas, entre otras habilidades.

ImagIng utiliza una red neuronal debido a la capacidad de aprender y reconocer patrones en los conjuntos de datos, que se utiliza para identificar elementos en las imágenes relacionadas

con las interfaces de usuario. El tipo de red neural empleada es un perceptrón multicapa, que emplea el algoritmo de activación llamado retropropagación (*back propagation*).

El perceptrón multicapa que utiliza un algoritmo de retropropagación es el algoritmo estándar para cualquier proceso de reconocimiento de patrones de aprendizaje supervisado y el tema de la investigación en curso en neurociencia computacional y procesamiento distribuido paralelo. Son útiles en la investigación en términos de su capacidad para resolver problemas estocásticamente, lo que a menudo permite obtener soluciones aproximadas para problemas extremadamente complejos como la aproximación de la aptitud. Algunos trabajos en los que este tipo de redes neuronales se utilizan con éxito son, para la recuperación de imágenes basadas en el contenido a la reducción de la brecha semántica (Qi, Zhang and Li 2016), para una clasificación de género de la huella digital (Abdullah, Rahman and Abas 2016), para la conversión de caracteres alfanuméricos en mapa de bits a su equivalente código ASCII (El-Henawy and Ahmed 2016). Para la clasificación de imágenes hiperespectrales (Garcia-Salgado, Ponomaryov and Robles-Gonzalez 2016), para el reconocimiento de la actividad de los teléfonos inteligentes por el sensor del acelerómetro (Dash, Kumar and Patle 2016), y para el reconocimiento de dígitos (Cireşan et al. 2012).

Para los ajustes de la red se realizó un conjunto de pruebas para obtener tasas aceptables de entrenamiento y evaluación. Los parámetros utilizados para el entrenamiento y la evaluación se describen a continuación:

1) **4 Momentos Hu**: Son parámetros estadísticos, invariantes al tamaño y posición de una figura en una imagen.
2) **Número de objetos por elemento**: Este número de objetos no considera las letras.
3) **Número de elementos incluyendo letras**: Esto incluye el número total de objetos de cada elemento y las letras que contiene el mismo elemento.
4) **Número de firmas en la figura circular**: Representa el número de objetos identificados en el elemento en forma de círculo.
5) **Número de firmas en la figura del rectángulo**: Representa el número de objetos encontrados dentro del elemento que es un rectángulo.

6) **Número de firmas en la figura cuadrada**: Representa el número de objetos dentro del elemento que es un cuadrado.
7) **Número de firmas en la figura del triángulo**: Representa el número de objetos contenidos en el elemento que es un triángulo.
8) **Número de firmas en la figura de línea**: Representa el número de objetos dentro del elemento que es una línea.
9) **Número de firmas en un patrón de UI**: Representa el número de esqueletos encontrados dentro del elemento conformado igual a un patrón de UI, patrones de interfaz de usuario tales como: video, login y carrusel, por mencionar sólo algunos.

Se consideraron parámetros de entrada relacionados con el número de cifras de firmas, ya que la mayoría de los elementos definidos están formados por figuras geométricas y el contorno de firmas adjunto de algunos elementos complejos, los mismos listados son parte de los parámetros de entrada para el entrenamiento y evaluación de la red neuronal. La técnica de la firma se explicó anteriormente, se describió cómo obtener para cada figura la distancia entre su centroide y el borde que limita la figura, de 1 a 360 grados. Para el número de firmas asociadas a las figuras ideales listadas arriba, se calcula la correlación cruzada entre la firma de cada figura a identificar (cifras no ideales) y las firmas de las figuras ideales, y se asocia con esa cifra con un valor más alto que se acerca más a la firma de cada figura ideal.
Para obtener los parámetros finales para el entrenamiento de la red neuronal, se realizó un conjunto de pruebas, los resultados se describen a continuación.

Entrenamiento y evaluación de la red neuronal

El entrenamiento y las pruebas de la red neuronal consistieron en generar un conjunto de muestras para cada elemento. Se consideraron 5 patrones de UI que son: Video, Splashscreen, Dashboard, Carousel y Login, para los cuales se generaron 50 imágenes para el entrenamiento y se consideraron 50 más para la evaluación de la red neuronal, haciendo un total de 250 imágenes para el entrenamiento y 250 imágenes adicionales para la evaluación.
Las características de las imágenes que se utilizaron para el entrenamiento y evaluación de la red neuronal fueron imágenes en formato .png con dimensiones de 400x200 píxeles. Todos los bocetos utilizados en las pruebas se generaron con la herramienta de dibujo Paint

considerando las dimensiones y el formato establecido. En la Figura 3.24, se muestra una muestra de los bocetos utilizados durante el entrenamiento y la evaluación de la red neuronal.

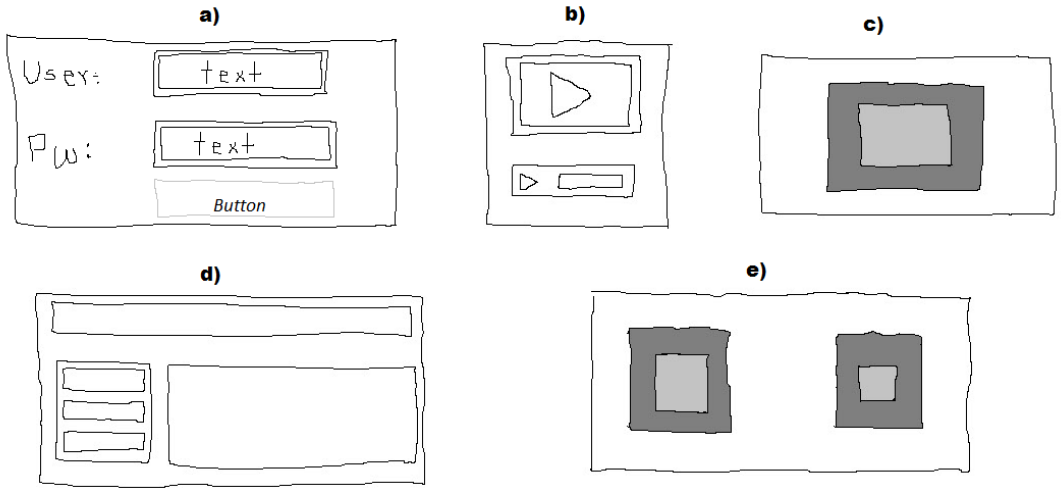


Figura 3.24. Ejemplo de *UI Design Patterns*: a) *Login,* b) *Video,* c) *Splashscreen, d) Dashboard y e) Carousel*

Los elementos utilizados en esta prueba fueron los patrones de diseño de interfaz de usuario: *carousel*, *dashboard*, *login*, *video* y *splashscreen*; Y como se mencionó anteriormente en cada PDIU se utilizó una muestra de 50 imágenes para el entrenamiento y 50 más para la evaluación de la red neuronal, es importante mencionar que con un mayor número de muestras utilizadas en el entrenamiento y evaluación de la red neuronal se amplía más el rango de precisión, sin embargo el número de muestras utilizadas es aceptable para una red neuronal como la que se está utilizando. Para los ajustes adecuados de la red neural se realizaron un conjunto de pruebas con diferentes parámetros de entrada, en estas pruebas sólo se tuvo en cuenta los patrones de diseño de interfaz de usuario antes mencionados; De estos elementos se extrajeron las siguientes características sin normalizarse: 4 Momentos invariantes de Hu, número de objetos, número de firmas en la figura circular, número de firmas en la figura del rectángulo, número de firmas en la figura cuadrada, número de firmas en la figura del triángulo, número de firmas de figuras de líneas y el número de firma de cada PDIU.

Prueba 1: La Tabla 3.1 presenta los resultados en los que se muestran el número de neuronas utilizadas en cada prueba con una sola capa oculta, el porcentaje de eficiencia en el entrenamiento y el porcentaje de eficiencia en la evaluación.

Tabla 3.1. Resultados de entrenamiento y evaluación con una capa oculta

No. Prueba	No. De neuronas	% Eficiencia en el entrenamiento	% Eficiencia en la evaluación
1	4	96.0 %	88.0 %
2	5	98.3 %	92.7 %
3	6	99.3 %	94.7%
4	7	100 %	91.3%
5	8	99.7 %	90.7%
6	9	99.7%	86.0%
7	10	100.0%	92.0%
8	11	99.3%	84.0%

Prueba 2: En esta prueba, se utilizó una configuración de dos capas ocultas en la red neuronal con diferentes números de neuronas en cada capa. La Tabla 3.2 muestra los porcentajes de entrenamiento y evaluación de la red neuronal con diferentes configuraciones. Cuando el porcentaje de eficiencia en el entrenamiento es inferior al 50%, la evaluación no se realizó, debido a esto es que algunas celdas aparecen vacías en la columna de evaluación.

Tabla 3.2. Resultados de entrenamiento y evaluación con dos capas ocultas

No. Prueba	No. De neuronas	% Eficiencia en el entrenamiento	% Eficiencia en la evaluación
1	1 / 1	33.3%	
2	1 / 2	40.7%	
3	1 / 3	58.3%	51.7%
4	1 / 4	75.7%	74.7%
5	1 / 5	74.7%	64.0%
6	1 / 6	70.0%	69.0%
7	1 / 7	59.0%	53.0%
8	1 / 8	61.0%	59.7%
9	1 / 9	68.7%	64.0%
10	1 / 10	71.3%	61.7%
11	2 / 1	33.3%	
12	2 / 2	50.0%	49.3%
13	2 / 3	66.7%	60.3%

No. Prueba	No. De neuronas	% Eficiencia en el entrenamiento	% Eficiencia en la evaluación
14	2 / 4	76.7%	73.3%
15	2 / 5	94.7%	87.7%
16	2 / 6	92.3%	82.7%
17	2 / 7	93.7%	89.3%
18	2 / 8	83.7%	74.3%
19	2 / 9	95.0%	87.7%
20	2 / 10	96.0%	87.3%
21	3 / 1	33.3%	
22	3 / 2	50.0%	48.3%
23	3 / 3	66.7%	64.0%
24	3 / 4	83.0%	76.0%
25	3 / 5	99.7%	88.3%
26	3 / 6	99.7%	90.7%

Como se observa en ambos casos, se obtienen porcentajes de entrenamiento y evaluación superiores al 90%, lo cual es suficientemente alto para que el reconocimiento de los patrones de diseño de IU a través de redes neuronales sea totalmente factible.

En consecuencia, la red neuronal fue seleccionada en base al porcentaje de entrenamiento y evaluación de mayor eficiencia. Dicho esto, la configuración final seleccionada es la obtenida en la prueba 1, con 18 parámetros de entrada, una capa oculta, 99,3% de eficiencia en el entrenamiento, 94,7% de eficiencia en la evaluación y 6 neuronas.

En la figura 3.25 se observa de manera gráfica los resultados en el entrenamiento de la red neuronal, los resultados se muestran en cuestión de entrenamiento, validación y pruebas, y en cada gráfica es posible observar la dispersión de los datos de entrada, así como los resultados de salida del proceso de entrenamiento. Cabe mencionar que para obtener los resultados presentados se realizaron diversas pruebas y se repitió el entrenamiento en diversas ocasiones hasta lograr un resultado que se acercara más al 100% de eficiencia.

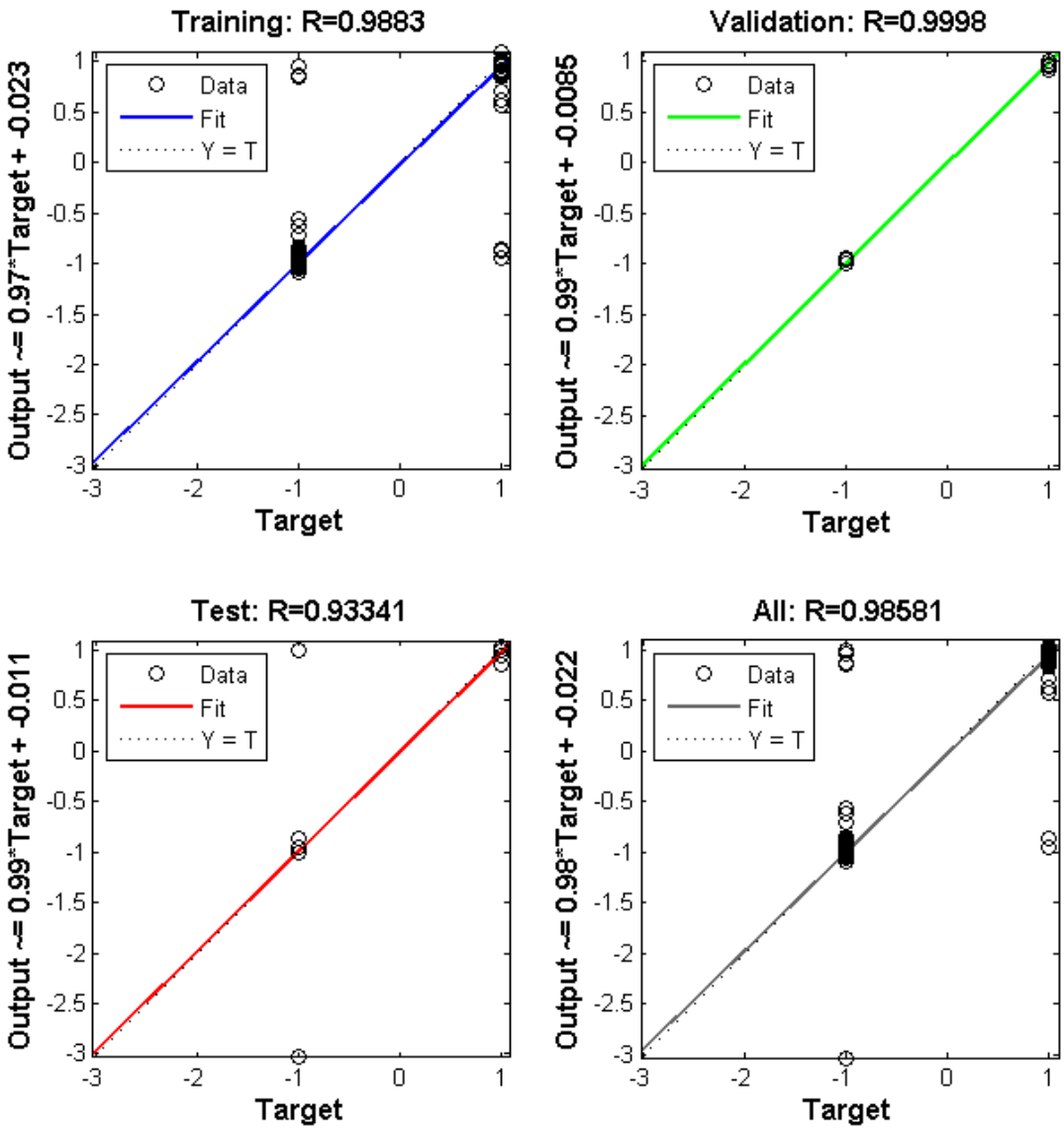


Figura 3.25. Gráficas de los resultados del entrenamiento de la red neuronal artificial

Pruebas de medición de tiempo en la identificación de elementos

Con la finalidad de evaluar el tiempo de reconocimiento de los distintos PDIUs se presentan un conjunto de pruebas y los resultados obtenidos.

Esta prueba consistió en procesar 10 veces una interfaz con solo un elemento de los antes listados en la tabla 4.4 y tomar el tiempo que se tardó en reconocer al elemento; una vez tomado los tiempo se obtiene el promedio de tiempo de reconocimiento. A continuación se muestran los resultados obtenidos para cada elemento en la tabla 3.3.

Tabla 3.3. Promedio de procesamiento por PDIU

PDIU	**Tiempo de procesamiento promedio en segundos**
Login	8.2
Video	8.7
Splashscreen	4.7
Dashboard	8.4
Carousel	6.5

A continuación se muestran los resultados obtenidos de la prueba que consistió en procesar interfaces con 1 y hasta 10 PDIUs iguales. En la prueba se toma el tiempo que se tarda en reconocer primero un solo elemento o PDIU, luego con dos elementos, tres elementos; y así sucesivamente hasta tomar el tiempo en una interfaz con 10 elementos, estos elementos como antes se menciona son de un mismo tipo. A continuación se presenta en la figura 3.26 la gráfica generada del PDIU Login.

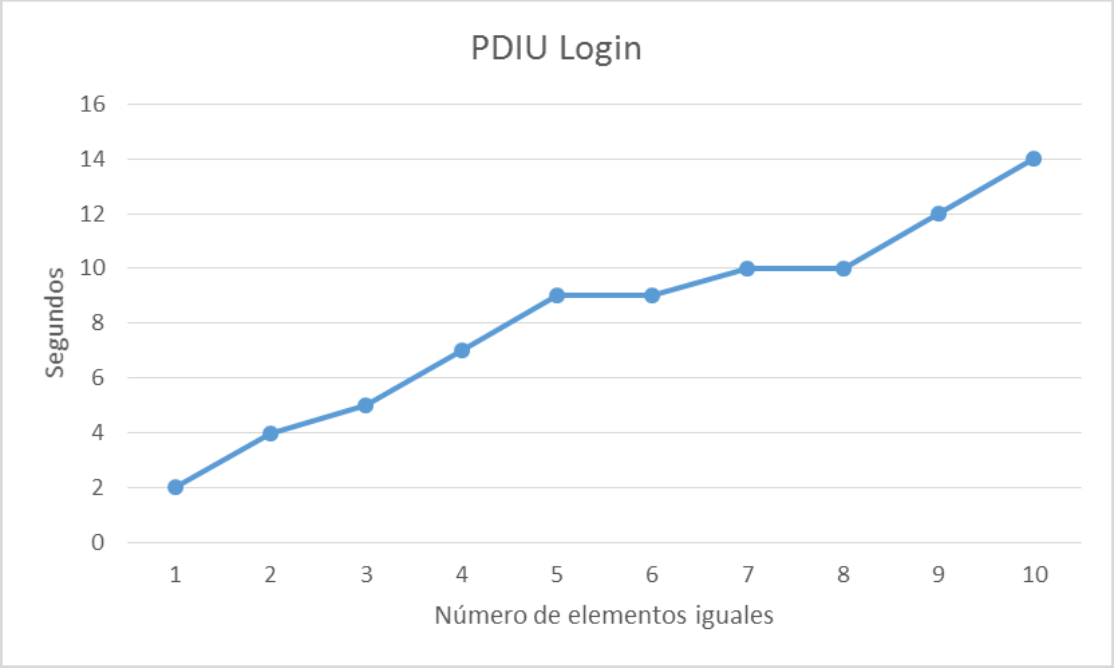


Figura 3.26. Gráfica del tiempo de procesamiento del PDIU Login

En la figura 3.27 se observa la gráfica del tiempo empleado en el procesamiento de 1 a 10 elementos iguales del PDIU Dashboard.

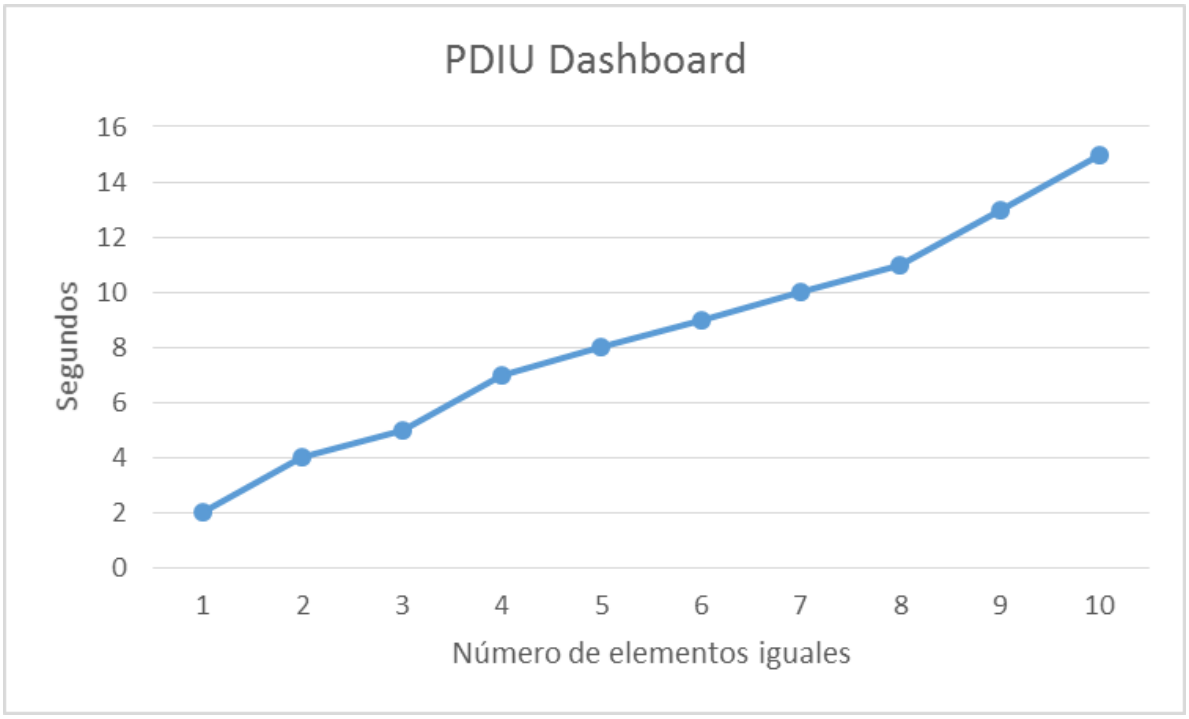


Figura 3.27. Gráfica del tiempo de procesamiento del PDIU Dashboard

En la figura 3.28 se observa la gráfica del PDIU Video.

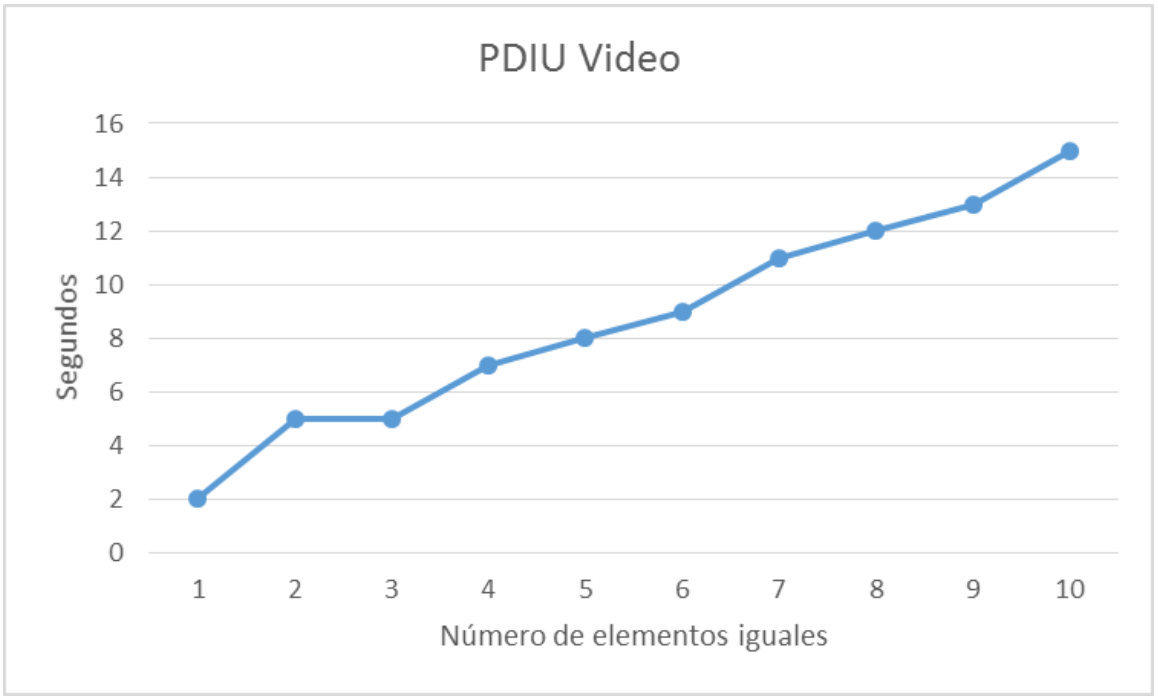


Figura 3.28. Gráfica del tiempo de procesamiento del PDIU Video

En la figura 3.29 se observa la gráfica del tiempo de procesamiento de las imágenes del PDIU Splashscreen.

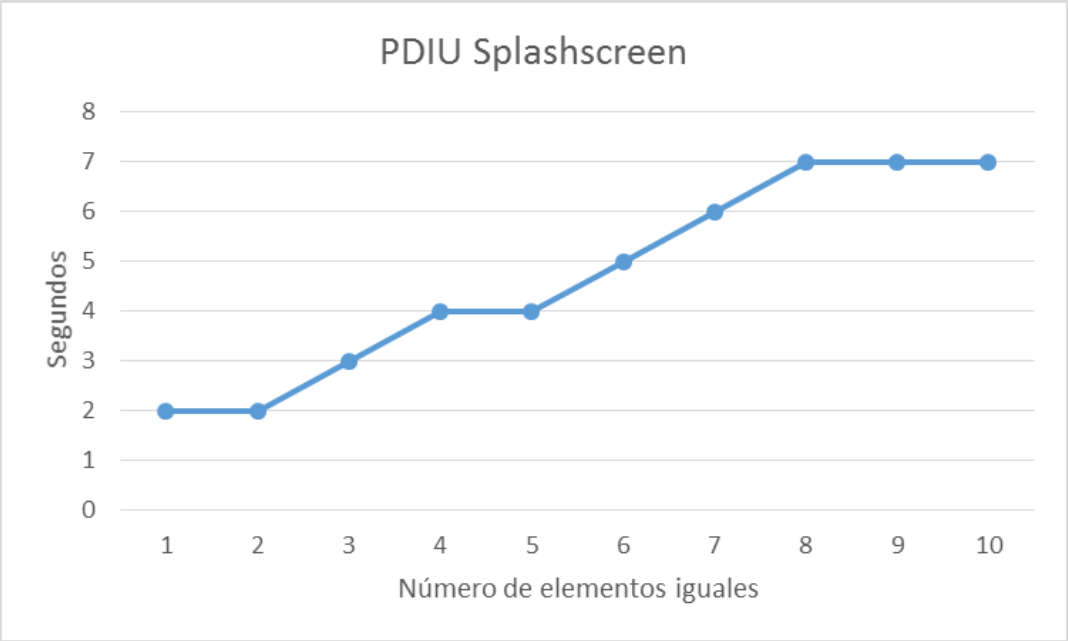


Figura 3.29. Gráfica del tiempo de procesamiento del PDIU Splashscreen

En la figura 3.30 se muestra la gráfica del tiempo de procesamiento del PDIU Carousel.

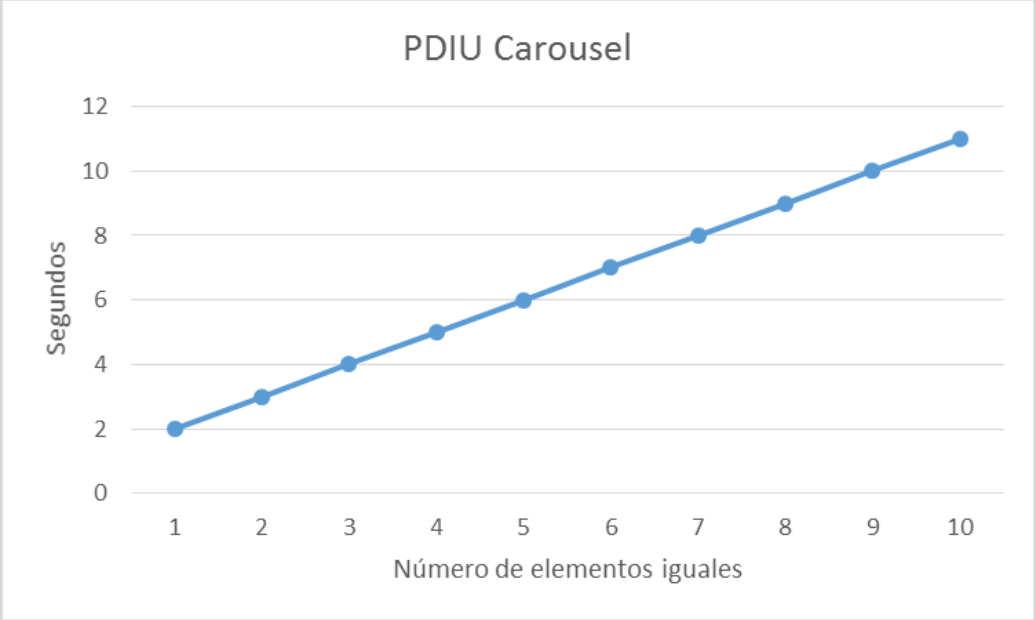
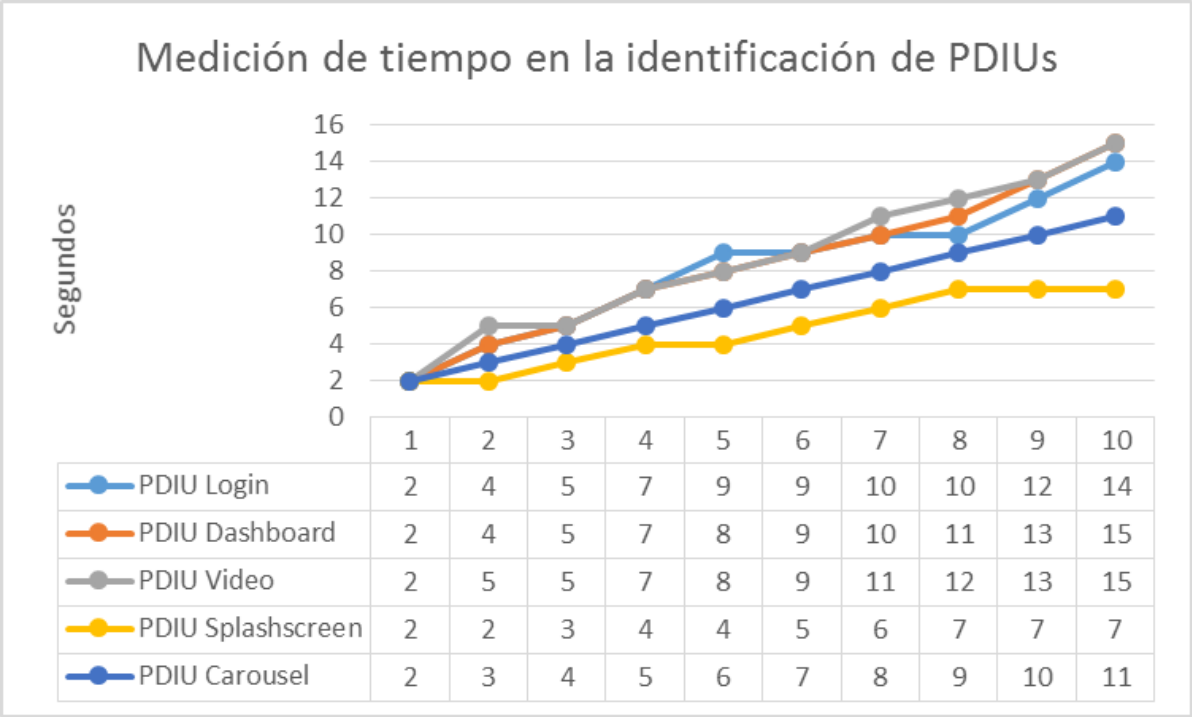


Figura 3.30. Gráfica del tiempo de procesamiento del PDIU Carousel

En la figura 3.31 se muestra la comparativa de los tiempos empleados para los 5 PDIU abordados en el presente trabajo de tesis.



	1	2	3	4	5	6	7	8	9	10
PDIU Login	2	4	5	7	9	9	10	10	12	14
PDIU Dashboard	2	4	5	7	8	9	10	11	13	15
PDIU Video	2	5	5	7	8	9	11	12	13	15
PDIU Splashscreen	2	2	3	4	4	5	6	7	7	7
PDIU Carousel	2	3	4	5	6	7	8	9	10	11

Figura 3.31. Gráfica del tiempo de procesamiento de PDIUs

Por el momento sólo se toman en cuenta los patrones mencionados anteriormente (*carousel*, *dashboard*, *login*, *video* y *splashscreen*), sin embargo, más patrones de interfaz de usuario se añadirán posteriormente. Para agregar más patrones de diseño de interfaz de usuario es necesario agregarlos tanto en la red neuronal para su reconocimiento como en el generador de código codificado para cada una de las plataformas soportadas. A continuación se presenta la estructura del documento XML utilizado en el generador de código fuente.

3.6. Estructura del documento XML del generador de código fuente

El documento XML se genera a partir de la información recabada durante el procesamiento de imágenes y se complementa con la configuración que realiza el usuario durante la generación de código a través del *wizard*, a continuación en la figura 3.32 se muestra la representación de la estructura del documento XML. Para comprender mejor la descripción

de los módulos que componen el documento XML en la figura 3.33, 3.34 y 3.35 se muestran fragmentos de cada una de las secciones que componen el documento XML respectivamente.

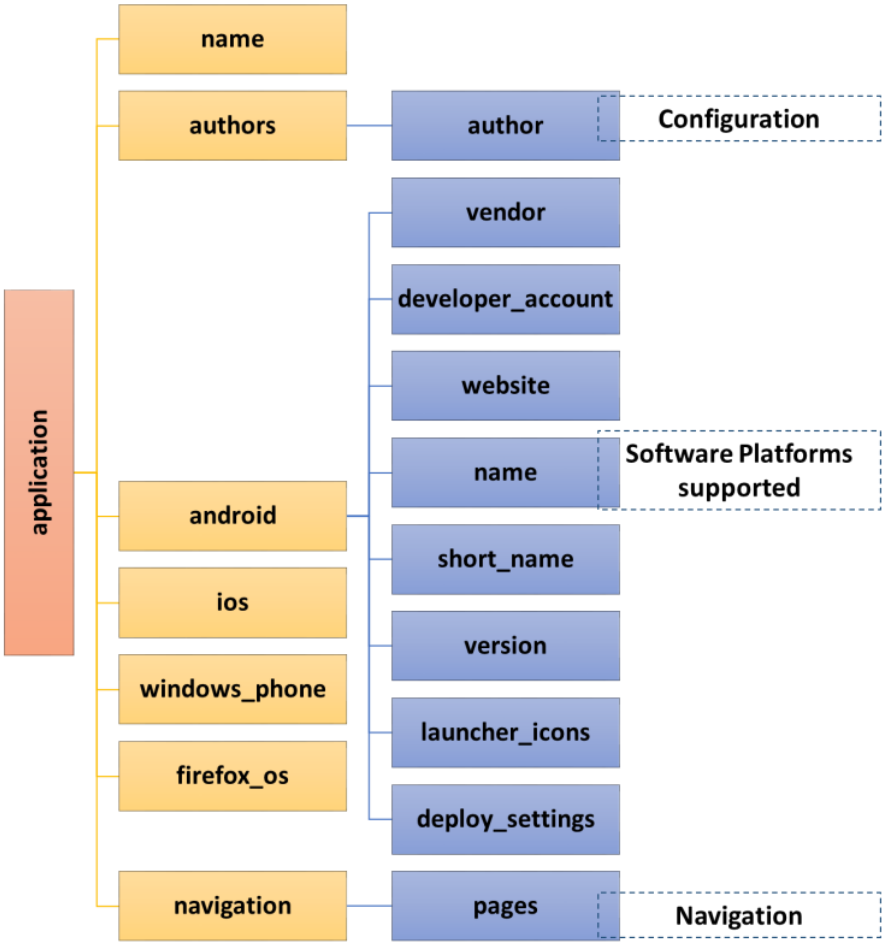


Figura 3.32. Estructura de documentos XML para generar aplicaciones

Tal como se muestra en la figura 3.32 la estructura de documentos basada en XML para generar aplicaciones se divide en 3 secciones:

1. **Configuración:** en esta sección se requiere toda la información sobre el usuario, información como nombre, correo electrónico, nombre de la institución, entre otros.
2. **Plataformas de software soportadas:** en esta sección se almacena la plataforma de desarrollo y la información para configurar la plataforma seleccionada, para cada plataforma la información requerida es distribuidor, cuenta de desarrollador, sitio Web, nombre, entre otros.

3. **Navegación:** en esta sección se guardan los datos de navegación de páginas de cada aplicación que se va a generar; Para cada página se requiere, nombre, título, referencia a la página siguiente y anterior, y posición del patrón de diseño de interfaz de usuario en la plantilla.

En la figura 3.33 se presenta un fragmento del documento XML de la sección de configuración, en él se observan las etiquetas usadas tal como se describieron anteriormente, etiquetas tales como nombre del autor(name), correo electrónico (email), sitio Web (website), por mencionar algunas. En esta sección se encuentra toda la información referente a la aplicación.

```
1   <?xml version="1.0"?>
2   <application xmlns="http://www.itodepi.edu.mx"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xsi:schemaLocation="http://www.itodepi.edu.mx define.xsd">
5
6       <authors>
7           <author>
8               <name>@@@author@@@</name>
9               <email>omeg_22@hotmail.com</email>
10              <website>www.itorizaba.edu.mx</website>
11              <company>@@@institution@@@</company>
12          </author>
13      </authors>
14      <deploySettings>
48      <navigation>
148 </application>
```

Figura 3.33. Fragmento del documento XML de la sección de configuración

En la figura 3.34 se presenta un fragmento del documento XML de la sección de plataformas de software soportadas, las etiquetas usadas son distribuidor (vendor), cuenta de desarrollador (developerAccount), sitio Web (website), nombre (name), entre otros, además de la información del dispositivo, la orientación y las plataformas.

```
4     xsi:schemaLocation="http://www.itodepi.edu.mx define.xsd">
5
6     <authors>
14    <deploySettings>
15        <userName>%%%author%%%</userName>
16        <name>%%%appname%%%</name>
17        <vendor>%%%institution%%%</vendor>
18        <developerAccount>ITO</developerAccount>
19        <website>www.itorizaba.edu.mx</website>
20        <shortName>%%%appname%%%</shortName>
21        <version>1.5</version>
22        <icon>icono.png</icon>
23
24        <device>
25            <desktop>true</desktop>
26            <tablet>true</tablet>
27            <smartphone>true</smartphone>
28        </device>
29        <orientation>
30            <portrait>%%%portrait%%%</portrait>
31            <landscape>%%%landscape%%%</landscape>
32        </orientation>
33        <plataforms>
34            <android>%%%android%%%</android>
35            <macOS>%%%macOS%%%</macOS>
36            <firefox>%%%firefox%%%</firefox>
37            <windowsPhone>%%%windowsPhone%%%</windowsPhone>
38            <web>%%%web%%%</web>
39        </plataforms>
40        <template>
41            <head>true</head>
42            <body>true</body>
43            <foot>true</foot>
44            <left>true</left>
45            <right>true</right>
46        </template>
47    </deploySettings>
48    <navigation>
148   </application>
```

Figura 3.34. Fragmento del documento XML de la sección de Plataformas de software soportadas

Por ultimo en la figura 3.35 se presenta un fragmento del documento XML de la sección de navegación, las etiquetas utilizadas son, nombre (name), título (title), referencia a la página

siguiente (next) y anterior (previous), y posición del patrón de diseño de interfaz de usuario en la plantilla (position).

```
<navigation>
    <page>
        <id>1</id>
        <name>@@@UIDP@@@</name>
        <title>@@@UIDP@@@</title>
        <type>@@@typeAPP@@@</type>
        <modal>false</modal>
        <next>2</next>
        <previous>2</previous>

        <position>
            <head></head>
            <body>
                @ac@@@UIDP@@@ac@
            </body>
            <left></left>
            <right></right>
            <foot></foot>
        </position>

        <navigationArray>
            <item>
                <id>5</id>
                <title>Option 1</title>
                <url>2</url>
                <type>blank</type>
                <target></target>
                <img></img>
                <backgroundcolor>red</backgroundcolor>
                <foreground></foreground>
                <visible>true</visible>
                <disabled>false</disabled>
```

Figura 3.35. Fragmento del documento XML de la sección de Navegación

3.7. Proceso de generación de código fuente

El proceso de generación de software consta de los siguientes módulos: 1) Análisis de la imagen, 2) Configuración y 3) Generación de código fuente, cada módulo se describe seguido de la Figura 3.36, donde se muestra de manera esquemática los módulos del proceso de generación de software a partir del uso de técnicas de inteligencia artificial y patrones de diseño de interfaces de usuario.

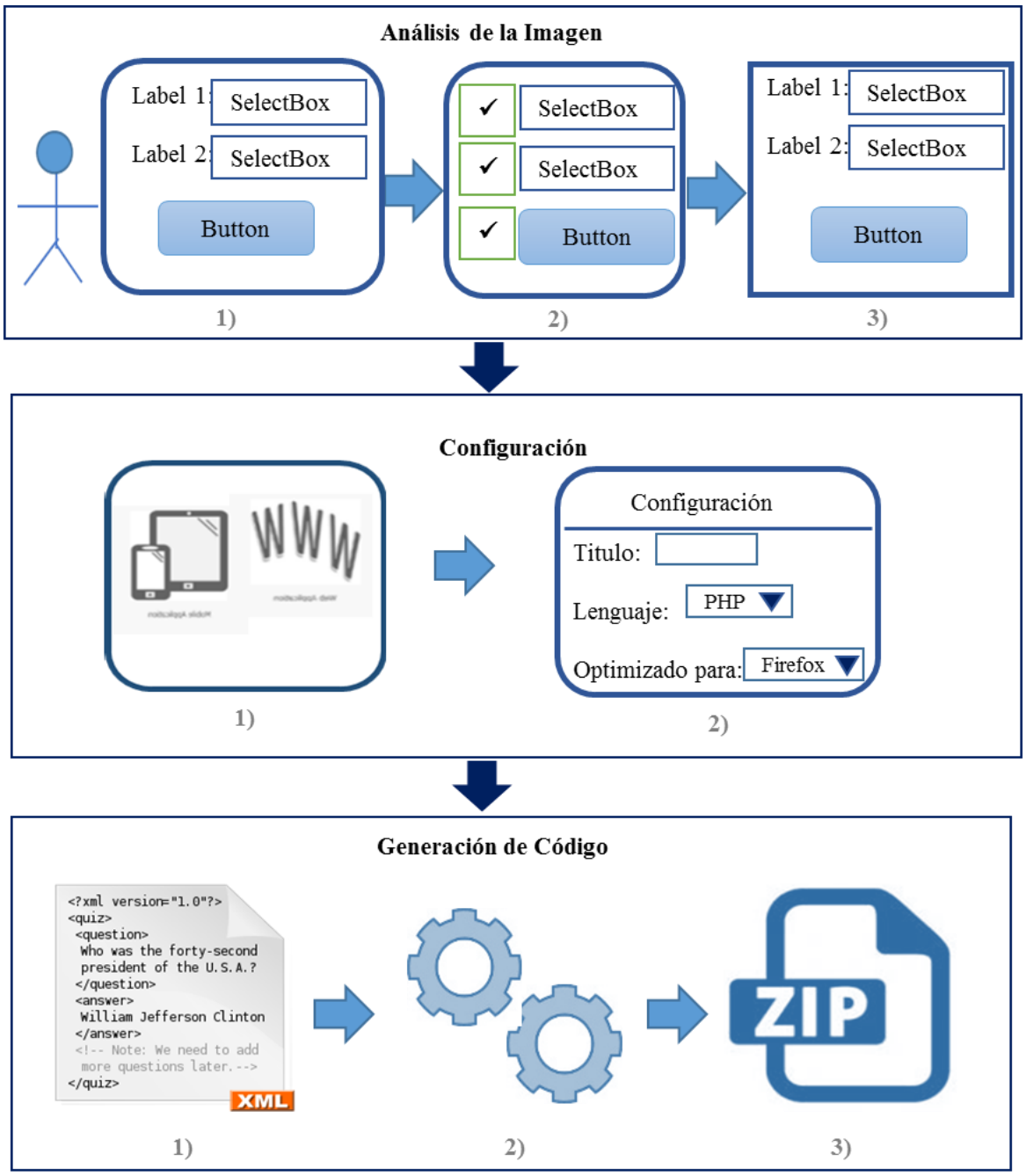


Figura 3.36. Módulos del proceso de generación de software

Análisis de la imagen: esta etapa considera la realización de un análisis de las imágenes que serán soportadas por los métodos para posteriormente generar el código fuente de una aplicación. Para lograr el análisis se consideran tres pasos importantes los cuales se explican con más detalle a continuación.

1. **Entrada de la imagen a procesar:** las imágenes están delimitadas previamente, las imágenes soportadas son modelos de diseños ADVs (*Abstract Data Views*) que permiten especificar clara y formalmente interfaces de usuario separadas de los componentes de la aplicación de un sistema de software. Los ADVs se generan por el usuario en cualquier herramienta de dibujo. Los formatos de imagen permitidos son: JPG o JPEG y PNG.
2. **Validación de la imagen de entrada**: La validación de la imagen consiste en aplicar los algoritmos de procesamiento de imágenes digitales tales como filtro de mediana, operaciones morfológicas, operaciones de convolución y detección de bordes para obtener los distintos elementos dentro de la imagen con la finalidad de asegurar que la imagen es una interfaz de usuario. Los elementos que la imagen contiene son las representaciones de patrones de diseño de interfaz de usuarios, los cuales serán seleccionados y definidos previamente.
3. **Identificación de elementos:** esto es, verificar si los encontrados en la imagen son elementos que pertenecen a una aplicación Web, móvil o ambas.

Configuración: el objetivo principal de esta etapa es, permitir la configuración del tipo de aplicación a generar (Web o móvil), después de la identificación de los elementos en la imagen durante la etapa anterior. Esta etapa comprende los siguientes pasos.

4. **Selección del tipo de aplicación a generar:** una aplicación Web o una aplicación para dispositivos móviles. Esta decisión se toma con la finalidad de que la generación de código se realice según las características de cada una de estas aplicaciones. Otro factor relevante que se considera en esta fase son los elementos encontrados en la imagen de entrada, ya que no todas las imágenes tendrán elementos válidos para generar cualquier tipo de aplicación.
5. **Configuración de la aplicación:** de acuerdo con el tipo de aplicación seleccionada en el paso anterior, se selecciona la configuración general de la aplicación (título principal, lenguaje, entre otras).

Generación de código: durante esta etapa se realizara la generación del código fuente de la aplicación para lo cual se ha considerado un conjunto de pasos que permitan realizar la

transformación de la información obtenida durante la etapa de análisis y la configuración. A continuación se describen los pasos considerados para esta etapa.

6. **Generación de un documento XML**: se pretende generar un documento XML que contenga etiquetas para almacenar el conjunto de datos obtenidos durante la configuración, dichos datos serán: nombre de la aplicación, autor (es), etiquetas con la configuración de acuerdo a cada tipo de aplicación y lenguaje seleccionado esto es; resolución, orientación y elementos de plantilla (header, body, footer, entre otros), por mencionar algunos; finalmente un conjunto de etiquetas con la representación de cada uno de los elementos identificados en la imagen.
7. **Transformación del documento XML a código fuente:** Se procesara el documento XML por medio de un documento XML Schema generando su equivalente en código HTML 5 de las etiquetas asociadas a cada elemento en la imagen, dicho código incrustado en Frameworks tales como Sencha Touch un Framework de HTML 5 que permite desarrollar aplicaciones Web para diferentes dispositivos móviles [52]; JQuery Mobile un Framework que permite agregar complejidad y enriquecer páginas [53]; y PhoneGap, un Framework de desarrollo de aplicaciones Web móviles que permite a los desarrolladores construir aplicaciones Web basadas en HTML5 y JavaScript con envoltorios para más de seis plataformas para dispositivos móviles, incluyendo iOS, Android™ y BlackBerry [54]. Por mencionar algunos; siguiendo con la descripción de este paso, la estructura generada se agrupa en clases de acuerdo con el lenguaje de programación seleccionado por el usuario.
8. **Entrega al usuario de un archivo ZIP con el código fuente generado:** Una vez que el usuario descargue el código fuente podrá realizar las modificaciones o adecuaciones al código fuente de acuerdo a sus necesidades y como mejor le convenga, puesto que esta propuesta pretende generar código fuente que sea apto para exportase a entornos de desarrollo sin dificultad alguna, y desde los cuales le sea posible al usuario editar el código fuente generado.

Una vez que el código generado es entregado al usuario es responsabilidad de este tanto la importación a los IDEs correspondientes como la implementación y ejecución del proyecto. A continuación se presenta la estructura final de los proyectos generados.

3.8. Estructura de los proyectos generados

Los proyectos generados tienen la estructura que corresponde a la plataforma seleccionada, en algunos casos es necesario incluir archivos de manifiesto, en otros archivos de configuración, entre otros. A continuación se muestra de manera real los archivos y estructura de carpetas que componen cada proyecto según la plataforma seleccionada.

Proyecto Web: Para un proyecto Web se genera la página HTML de la interfaz representada por el PDIU, y los archivos de recursos en la carpeta "rsc" tales como bibliotecas de funciones que se encuentran en la carpeta lib y archivos que se colocan por default, por ejemplo el video que se reproduce cuando se genera una aplicación de video en la carpeta "video". Los archivos y la estructura de carpetas se muestran en la figura 3.37.

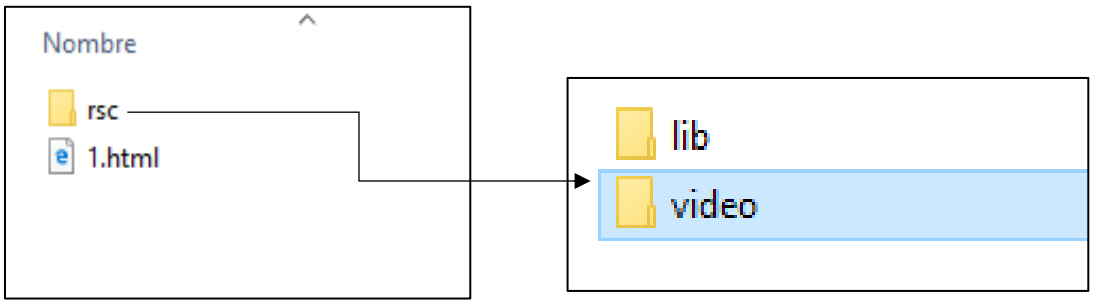


Figura 3.37. Archivos y estructura de carpetas de una aplicación Web

Proyecto Android™: Para un proyecto en Android™ es necesario incluir una serie de archivos, en un caso particular se incluyen los archivos de Gradle en la carpeta "gradle", Gradle es una herramienta utilizada para automatizar la construcción de aplicaciones. Además también se incluyen los archivos de configuración como el AndroidManifest.xml ubicado dentro de la carpeta "src" dentro de la carpeta "app", los recursos de la aplicación y los archivos propios de las interfaces que sean solicitadas en el *wizard* del generador. Los principales archivos generados y la estructura de carpetas se muestran en la figura 3.38.

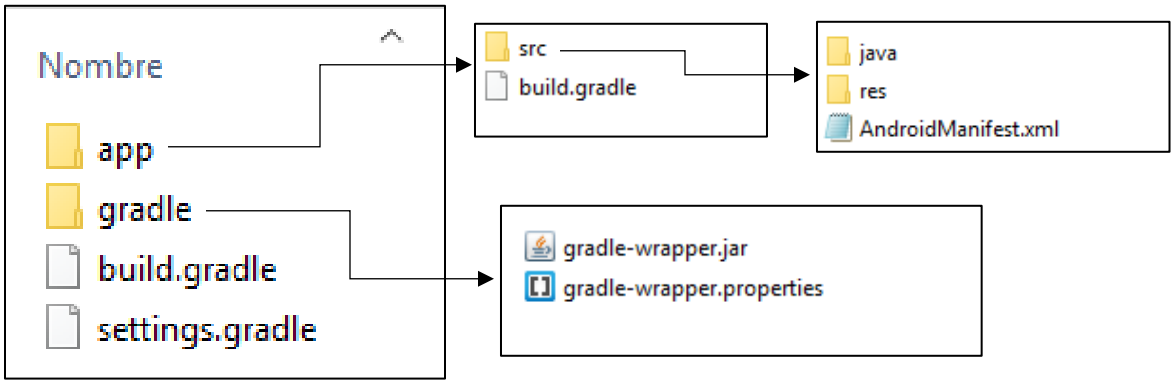


Figura 3.38. Archivos y estructura de carpetas de una aplicación Android™

Proyecto Firefox® OS: Para un proyecto en Firefox® OS es necesario incluir el archivo de manifiesto (manifest.webapp), las páginas html generadas de las interfaces solicitadas en el *wizard* del generador, así como también la carpeta de recursos llamada "rsc" en la cual se incluyen imágenes, bibliotecas de funciones y recursos por default como por ejemplo el video que se reproduce en una aplicación de video. Los principales archivos y la estructura de carpetas se muestran en la figura 3.39.

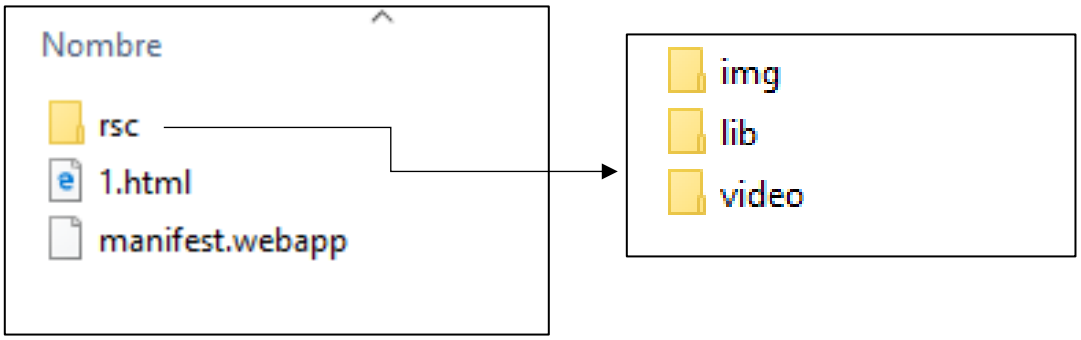


Figura 3.39. Archivos y estructura de carpetas de una aplicación Firefox® OS

Proyecto iOS o MacOS®: Para un proyecto en MacOS® es necesario incluir el archivo del proyecto que será importado en el IDE Xcode (project.pbxproj), además de los archivos de recursos y las páginas solicitadas en el *wizard* del generador, estas páginas se ubican en la carpeta "html" y en ella se encuentran las bibliotecas de funciones y los recursos necesarios para generar la aplicación como imágenes y videos, por mencionar algunos. Los principales archivos y la estructura de carpetas se muestran en la figura 3.40.

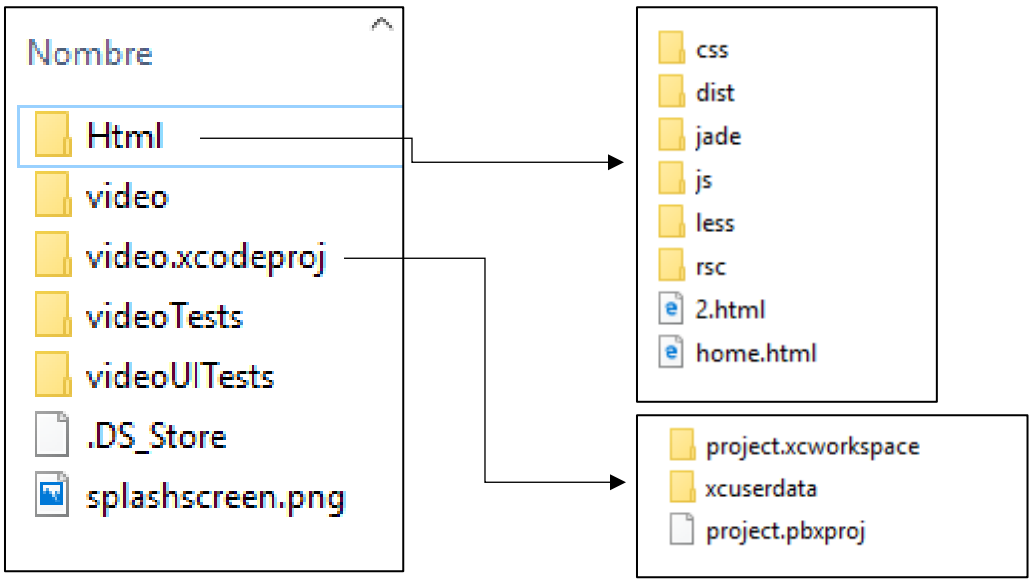


Figura 3.40. Archivos y estructura de carpetas de una aplicación MacOS®

Proyecto Windows Phone®: Para el caso de los proyectos de Windows Phone® es necesario incluir el archivo con terminación ".sln" el cual será importado en el IDE Visual Studio®, además se incluyen los archivos de recursos y las páginas solicitadas en el *wizard* del generador. Los archivos principales y la estructura de carpetas se muestran en la figura 3.41.

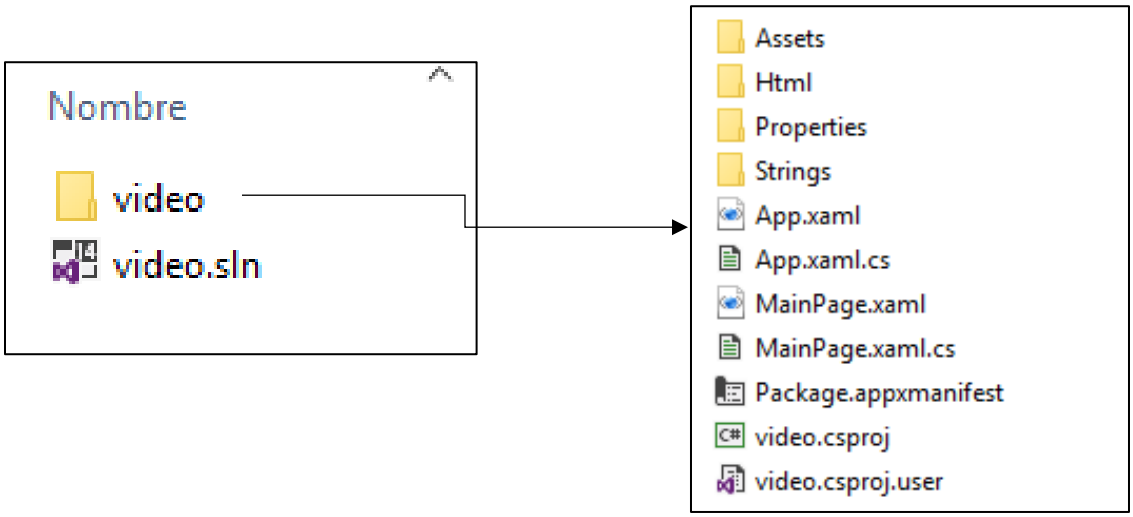


Figura 3.41. Archivos y estructura de carpetas de una aplicación Windows Phone®

Capítulo 4: Resultados

4.1. Caso de estudio 1: Generación de una aplicación de videos con un PDIU ideal

Supongáse que un desarrollador novato necesita desarrollar una aplicación de entretenimiento que sea capaz de reproducir videos. Para utilizar la herramienta ImagIng el desarrollador requiere diseñar u obtener una imagen que represente un reproductor de video, es decir una representación del PDIU video, para este caso particular se usó una imagen ideal (un ADV), es decir una imagen trazada con figuras geométricas en la herramienta de dibujo Paint. En la figura 4.1 se observa la imagen utilizada.

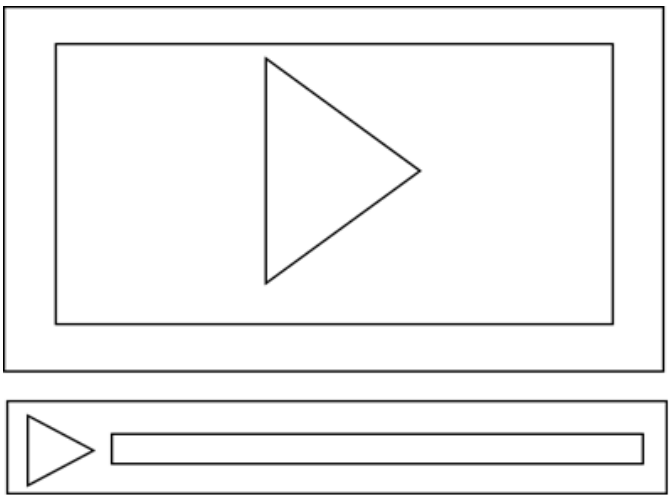


Figura 4.1. Imagen que representa el PDIU video

El desarrollador (usuario) quiere desarrollar la aplicación multiplataforma y multidispositivo en el menor tiempo posible y a través de pocos pasos. Bajo condiciones regulares, el proceso de desarrollo plantea dificultades a un desarrollador novato, ya que tomará tiempo en el desarrollo de las interfaces.

¿Cómo realiza el desarrollador (usuario) el prototipo de la aplicación de entretenimiento sin ser experto en programación Web ni móvil? Una solución a este problema es utilizar ImagIng. La herramienta ImagIng desarrolla rápidamente el prototipo de la aplicación en pocos pasos. El primer paso del *wizard* es subir la imagen que representa el PDIU de video (IconoVideo.png) como se muestra en la Figura 4.2. ImagIng soporta formatos JPG y PNG, y se recomienda el uso de imágenes de 400x200 píxeles aproximadamente, y no mayor de 1 MB.

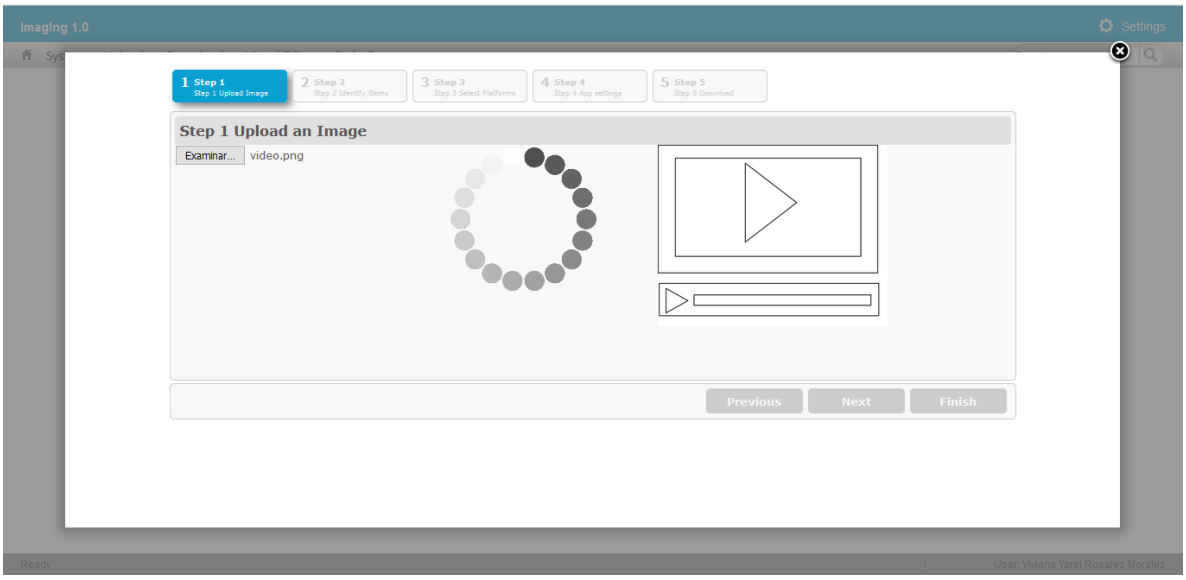


Figura 4.2. Paso 1 de la herramienta ImagIng: Subir imagen

Después de subir y procesar la imagen, en el paso 2 del *wizard*, se muestra el PDIU identificado como se muestra en la Figura 4.3. Si el usuario está de acuerdo, el proceso continúa.

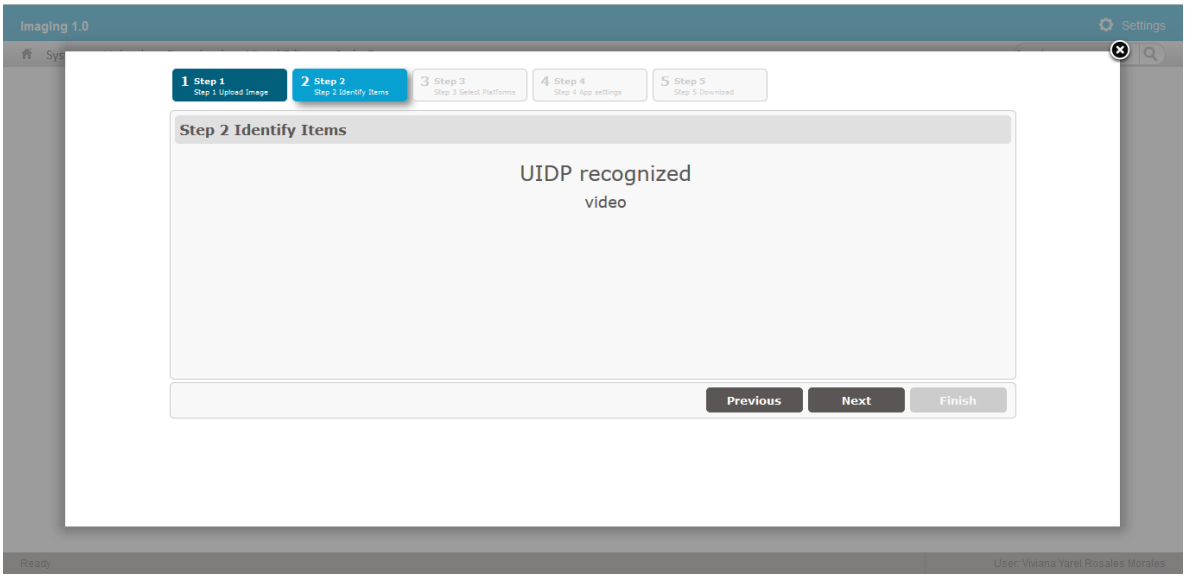


Figura 4.3. Paso 2 de la herramienta ImagIng: Identificación de los PDIUs

En el paso 3 del *wizard*, el usuario selecciona las plataformas deseadas para la aplicación, tal como se muestra en la figura 4.4. El usuario selecciona una o varias plataformas al mismo tiempo, en este caso seleccionará todas las plataformas disponibles, y el código se generará

en una carpeta para cada plataforma seleccionada. Las plataformas disponibles en ImagIng son: Android™, iOS, Windows Phone® y Firefox® OS, además de Web.

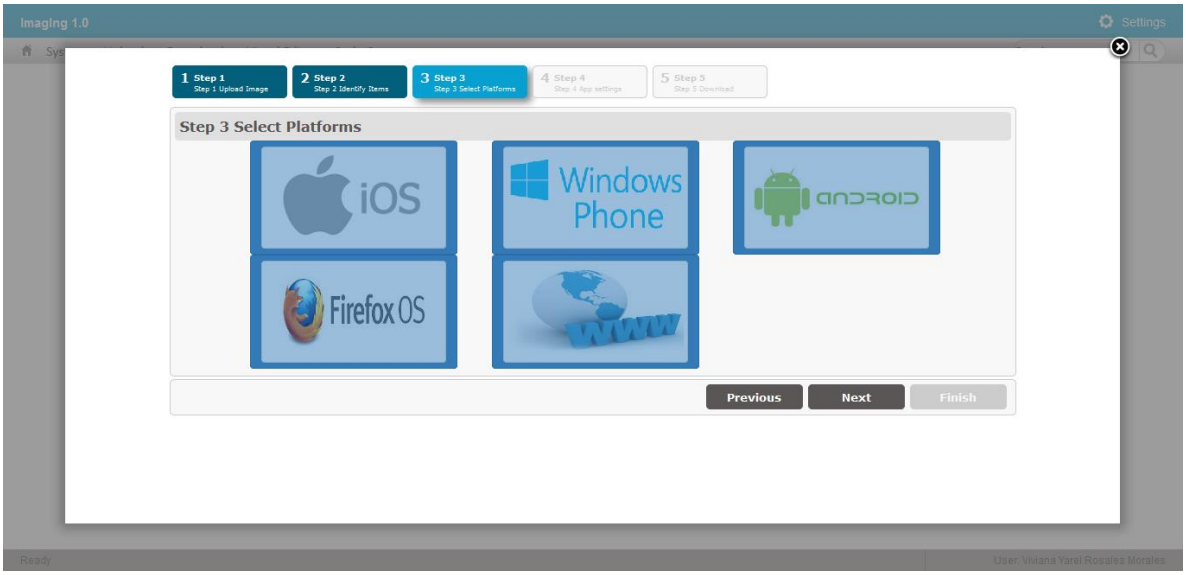


Figura 4.4. Paso 3 de la herramienta ImagIng: Selección de las plataformas

Una vez que la plataforma o plataformas se seleccionan, el usuario ingresa la configuración general de la aplicación. Este cuarto paso se muestra en la Figura 4.5., el usuario introduce la configuración general de la aplicación, como el nombre de la aplicación a generar, del autor y de la institución, así como la orientación.

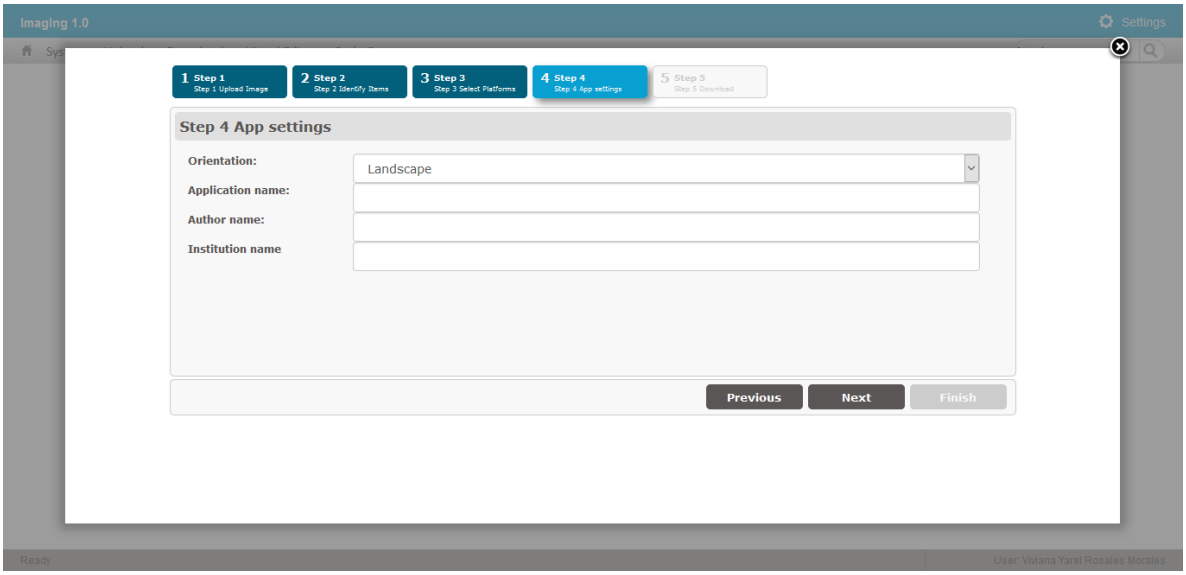


Figura 4.5. Paso 4 de la herramienta ImagIng: Configuración

Antes de pasar al paso 5 la aplicación pide confirmar los datos con los que será generada la aplicación tal como se muestra en la figura 4.6.

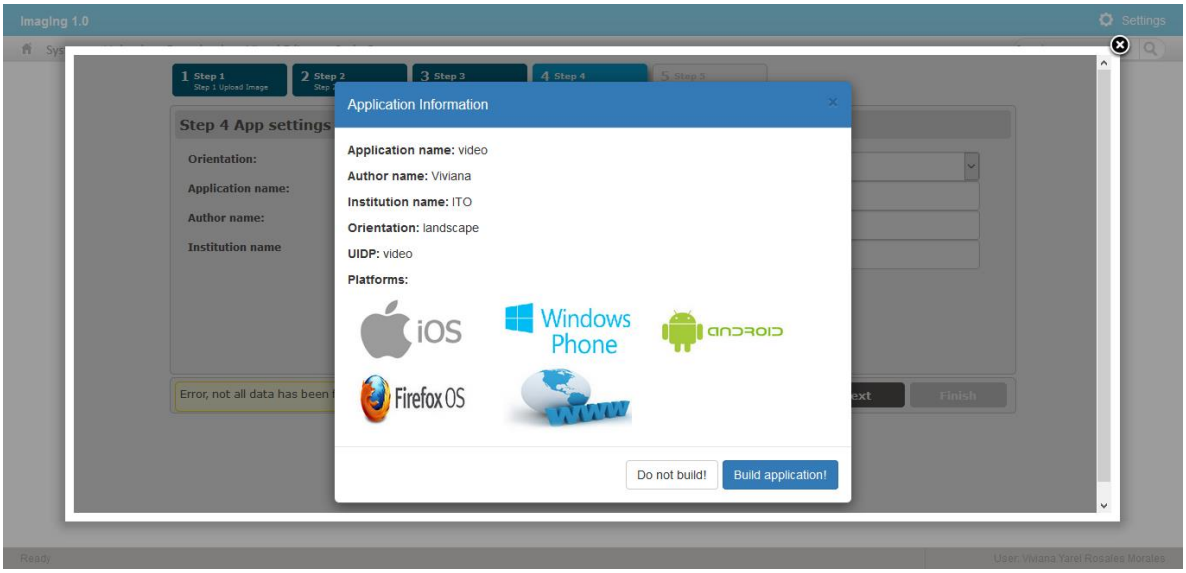


Figura 4.6. Paso 4 de la herramienta ImagIng: Información de la aplicación

Una vez que se seleccionó la plataforma y es configurada la aplicación, ImagIng genera un documento basado en XML que contiene la información necesaria para generar la aplicación o el código fuente de las aplicaciones. Este proceso es interno y, por lo tanto, no es visible para el usuario. El documento basado en XML se procesa en el componente Generador de código fuente para generar el código para la plataforma seleccionada. A continuación, en el paso 5, el usuario descarga finalmente el código generado tal como se muestra en la figura 4.7. El proceso de generación de código que se sigue se muestra en la arquitectura de la herramienta ImagIng, en la capa de servicios. El código fuente se genera en el lenguaje de programación soportado por cada plataforma y en carpetas separadas, como se mencionó anteriormente.

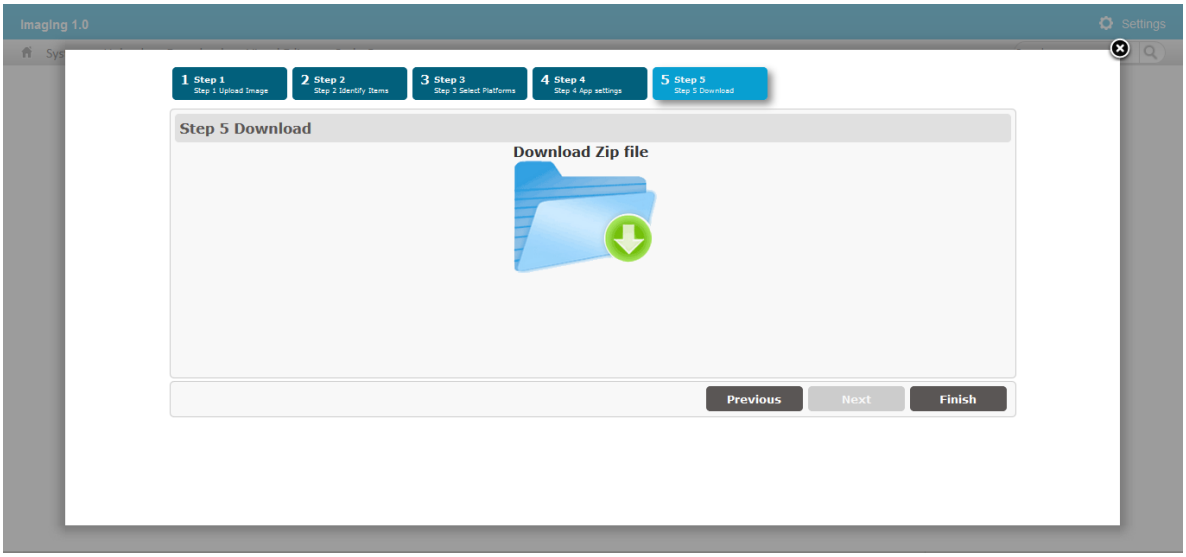


Figura 4.7. Paso 5 de la herramienta ImagIng: Descarga

La Figura 4.8 presenta el reproductor de video generado.

Figura 4.8. Interfaz generada para el PDIU video

Para desplegar la aplicación generada es suficiente con abrirla con algún browser disponible en su equipo de cómputo o importarla en los IDEs correspondientes según sea el caso.

4.2. Caso de estudio 2: Generación de una aplicación de galería fotográfica con un PDIU no ideal

Ahora suponga que un desarrollador novato necesita desarrollar una aplicación de entretenimiento que haga las funciones de una galería fotográfica. Para utilizar la herramienta ImagIng el desarrollador dibuja una imagen que represente un carrusel, es decir una representación del PDIU carrusel o *carousel*, para este caso particular se usó una imagen no ideal, es decir una imagen trazada a mano alzada en la herramienta de dibujo Paint. En la figura 4.9 se observa la imagen utilizada.

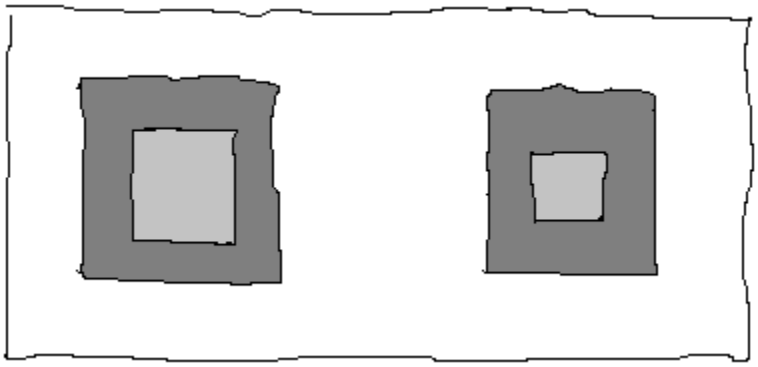


Figura 4.9. Imagen que representa el PDIU carrusel

En este caso el desarrollador (usuario) quiere desarrollar la aplicación Web en el menor tiempo posible y a través de pocos pasos. Bajo condiciones regulares, el proceso de desarrollo plantea dificultades a un desarrollador novato, ya que tomará tiempo en el desarrollo de las interfaces.

Utilizando ImagIng el desarrollador genera rápidamente el prototipo de la aplicación en pocos pasos. Para comenzar se requiere subir la imagen que representa el PDIU de carrusel hecha a mano alzada (carousel.png) en el primer paso del *wizard* como se muestra en la Figura 4.10.

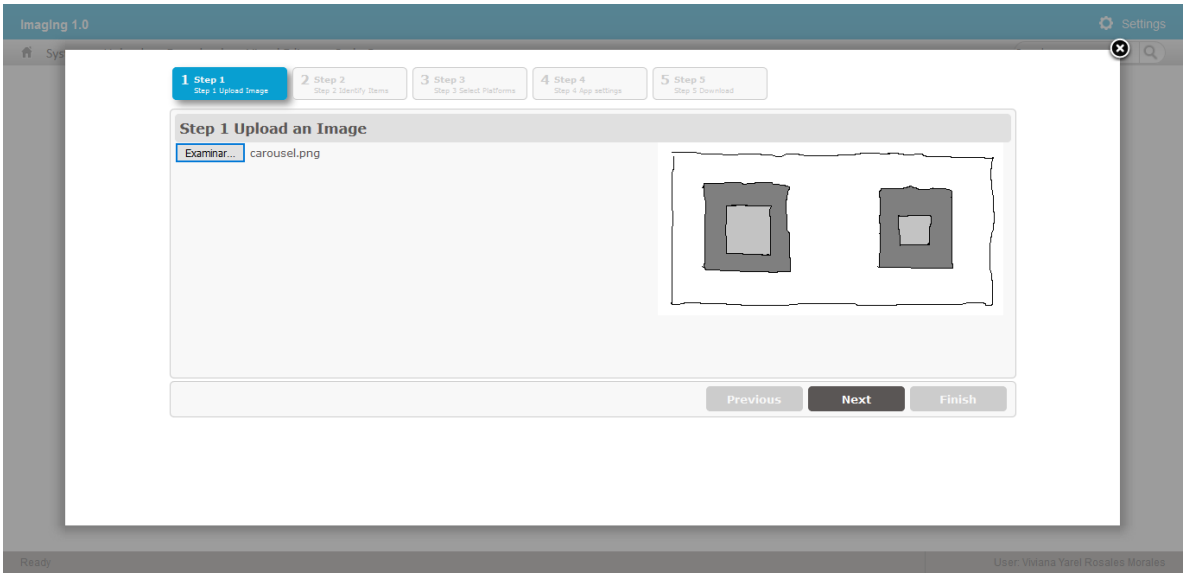


Figura 4.10. Paso 1 de la herramienta ImagIng: Subir imagen

En el paso 2 del *wizard*, se muestra el PDIU que se identificó por medio del procesamiento de la imagen como se muestra en la Figura 4.11. Si el usuario está de acuerdo, el proceso continúa.

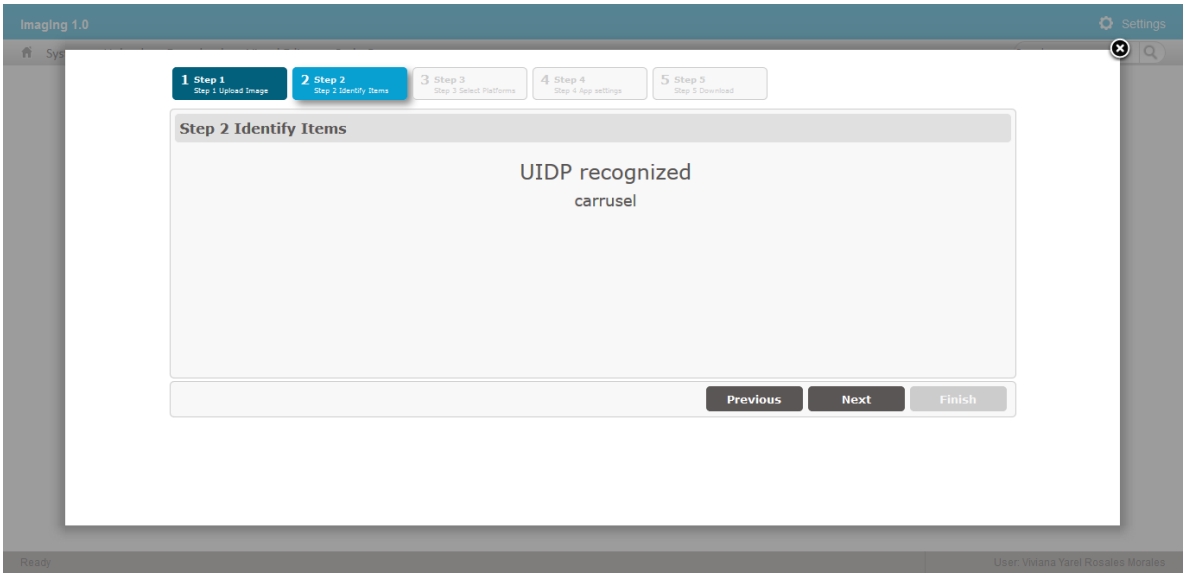


Figura 4.11. Paso 2 de la herramienta ImagIng: Identificación de los PDIUs

En el paso 3 del *wizard*, el usuario selecciona las plataformas deseadas para la aplicación como se muestra en la figura 4.12. En este caso solo se desea generar la aplicación Web.

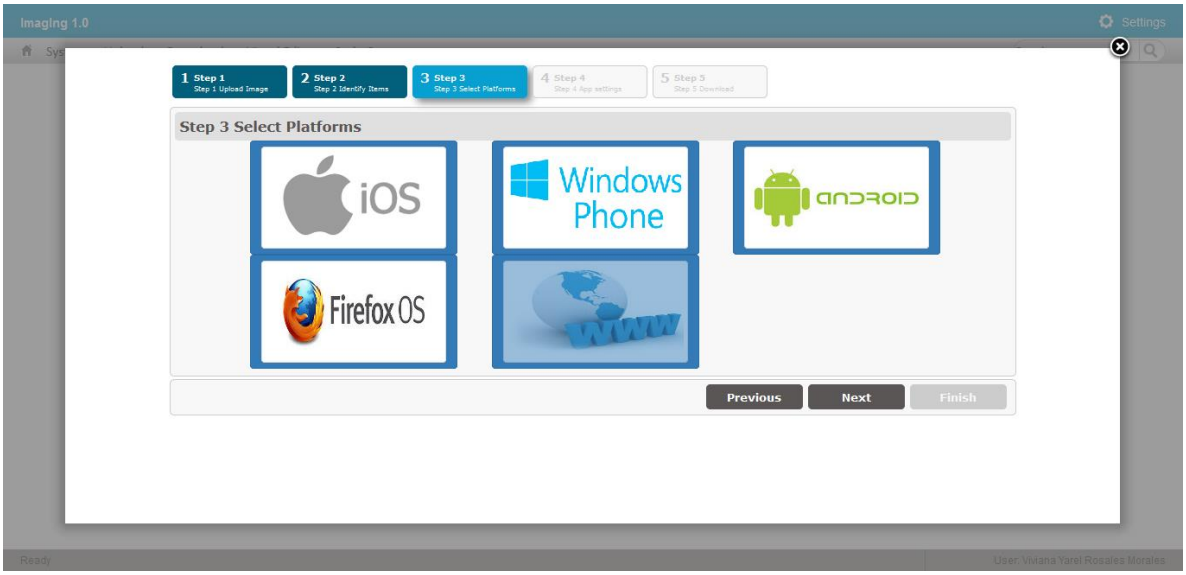


Figura 4.12. Paso 3 de la herramienta ImagIng: Selección de las plataformas

Una vez que la plataforma fue seleccionada, el usuario ingresa la configuración general de la aplicación. Este cuarto paso se muestra en la Figura 4.13., el usuario introduce la configuración general de la aplicación, como el nombre de la aplicación a generar, del autor y de la institución, así como la orientación.

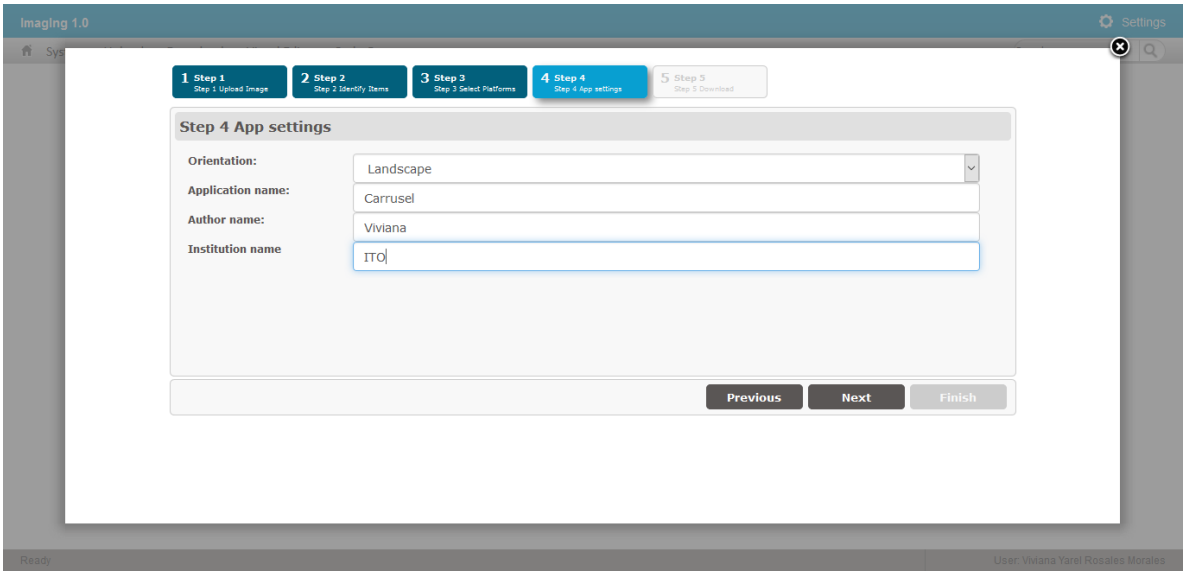


Figura 4.13. Paso 4 de la herramienta ImagIng: Configuración

Antes de pasar al paso 5 la aplicación pide confirmar los datos con los que será generada la aplicación tal como se muestra en la figura 4.14.

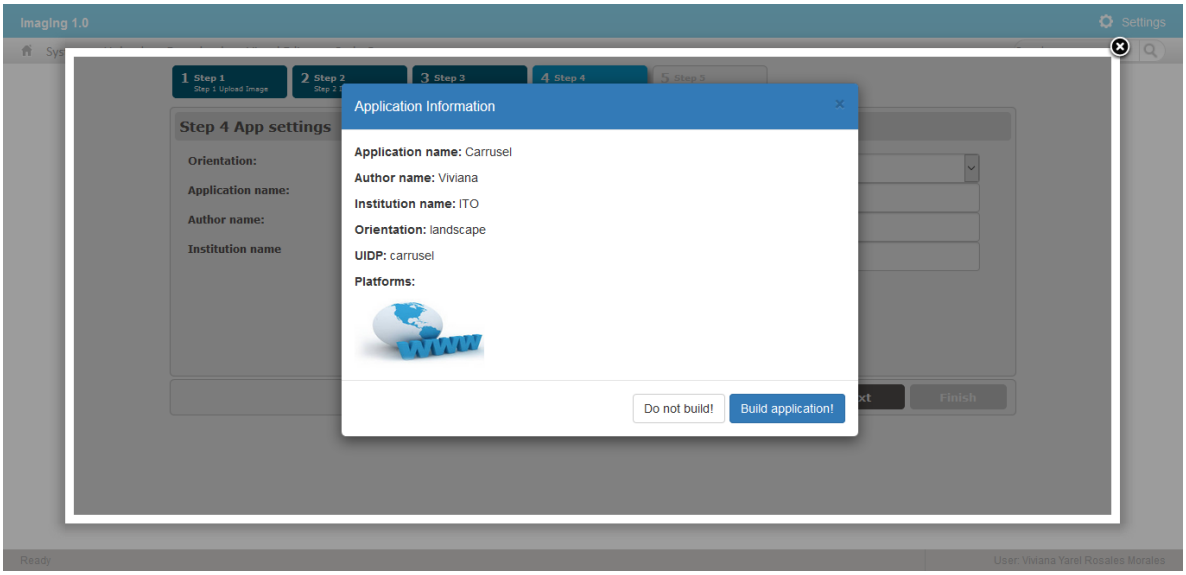


Figura 4.14. Paso 4 de la herramienta ImagIng: Información de la aplicación

A continuación, en el paso 5, el usuario descarga finalmente el código generado tal como se muestra en la figura 4.15.

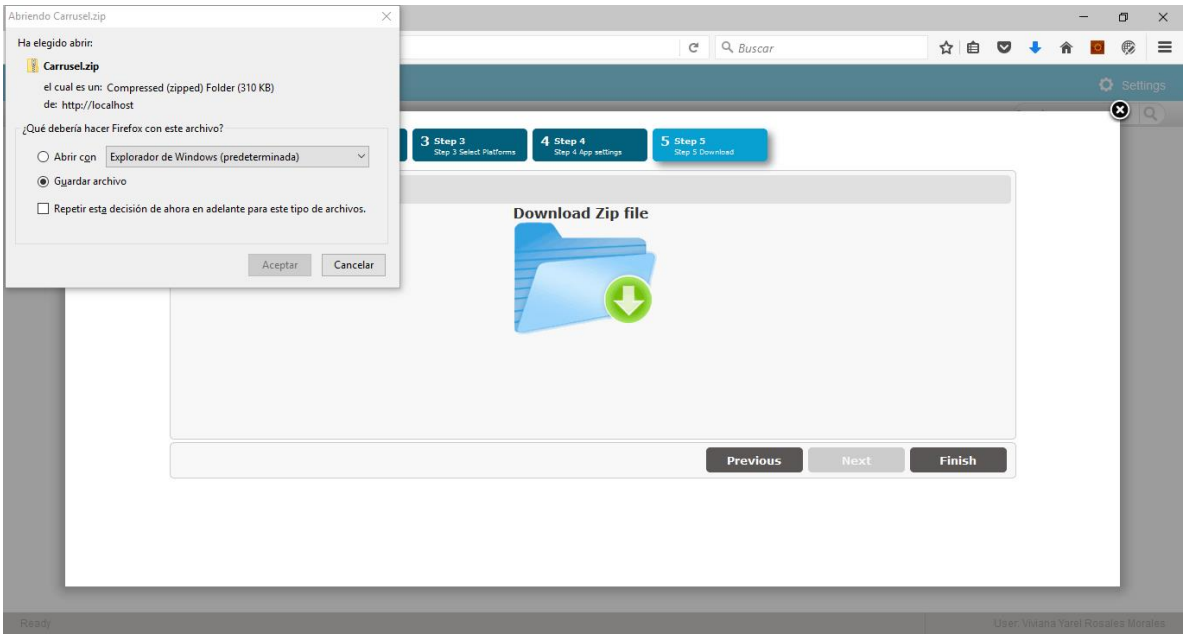


Figura 4.15. Paso 5 de la herramienta ImagIng: Descarga

La Figura 4.16 presenta la galería fotográfica generada (las fotos que se incluyen son recursos por default, es responsabilidad del desarrollador cambiarlas).

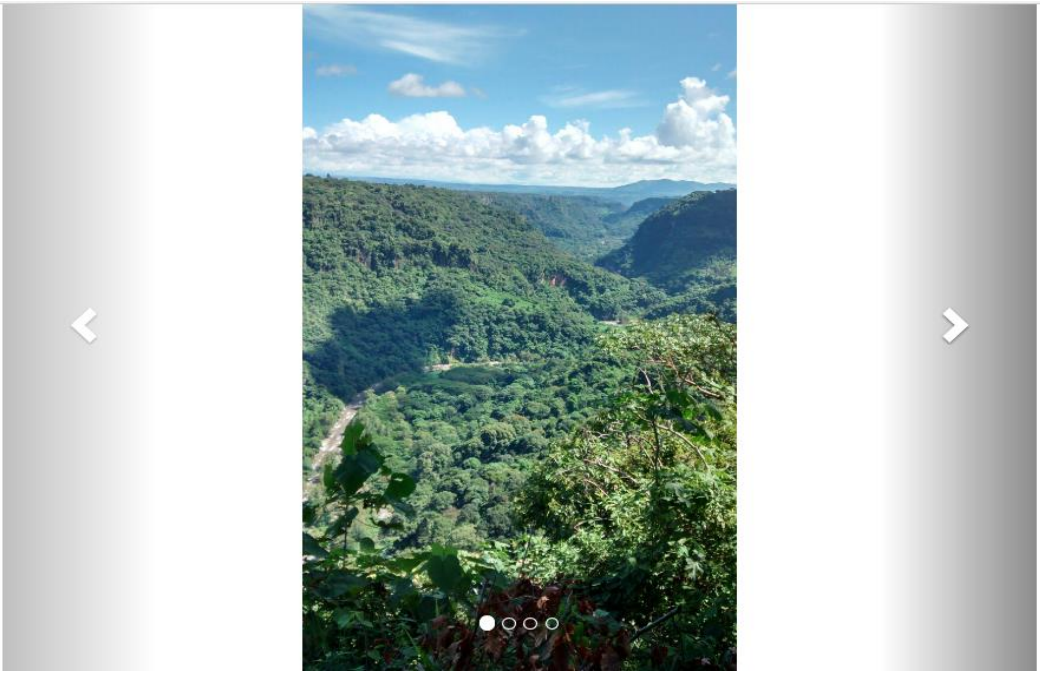


Figura 4.16. Interface generada para el PDIU carrusel

Para desplegar la aplicación generada es suficiente con abrirla con algún browser disponible en su equipo de cómputo.

4.3. Caso de estudio 3: Generación de una aplicación sistema de planificación con PDIUs no ideales

Para este caso supongáseque un desarrollador novato necesita desarrollar una aplicación para un sistema de planificación, esta aplicación es capaz de autenticar a los usuarios y mostrar una pantalla con los enlaces para la administración de la información del sistema. Se requiere que la aplicación sea generada para múltiples dispositivos y plataformas para dispositivos móviles. Para utilizar la herramienta ImagIng el desarrollador diseña una imagen que

represente a cada PDIU que requiere el sistema, es decir una representación del PDIU *login* para la autenticación de usuarios y una representación del PDIU *dashboard* para la administración de la información del sistema, para este caso se utilizaron imágenes no ideales, es decir imágenes trazada a mano alzada en la herramienta de dibujo Paint. En la figura 4.17 y 4.18 se muestran las imágenes utilizadas.

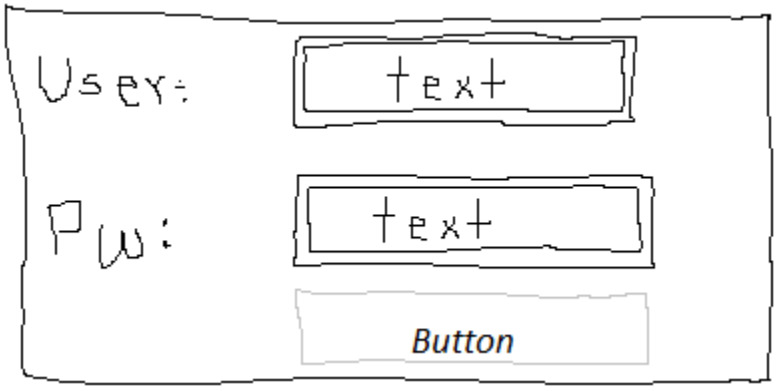


Figura 4.17. Imagen que representa el PDIU *login*

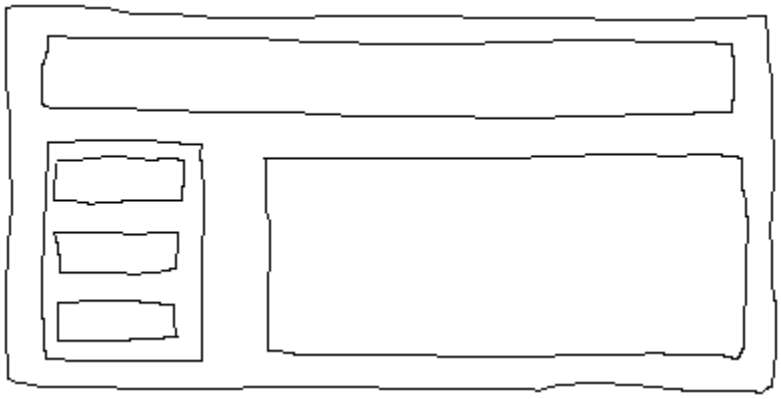


Figura 4.18. Imagen que representa el PDIU dashboard

El desarrollador (usuario) quiere desarrollar la aplicación multiplataforma y multidispositivo en el menor tiempo posible y a través de pocos pasos. Bajo condiciones regulares, el proceso de desarrollo plantea dificultades a un desarrollador novato, ya que tomará tiempo en el desarrollo de las interfaces, además de las dificultades que implicaría que el desarrollador no sea experto en programación móvil.

Utilizando ImagIng se desarrolla rápidamente el prototipo de la aplicación en pocos pasos. Para iniciar se requiere subir las imágenes que representan los PDIUs requeridos, en este caso

el PDIU *login* y el PDIU *dasboard* (login.png y dashboard.png) como se muestra en la figura 4.19.

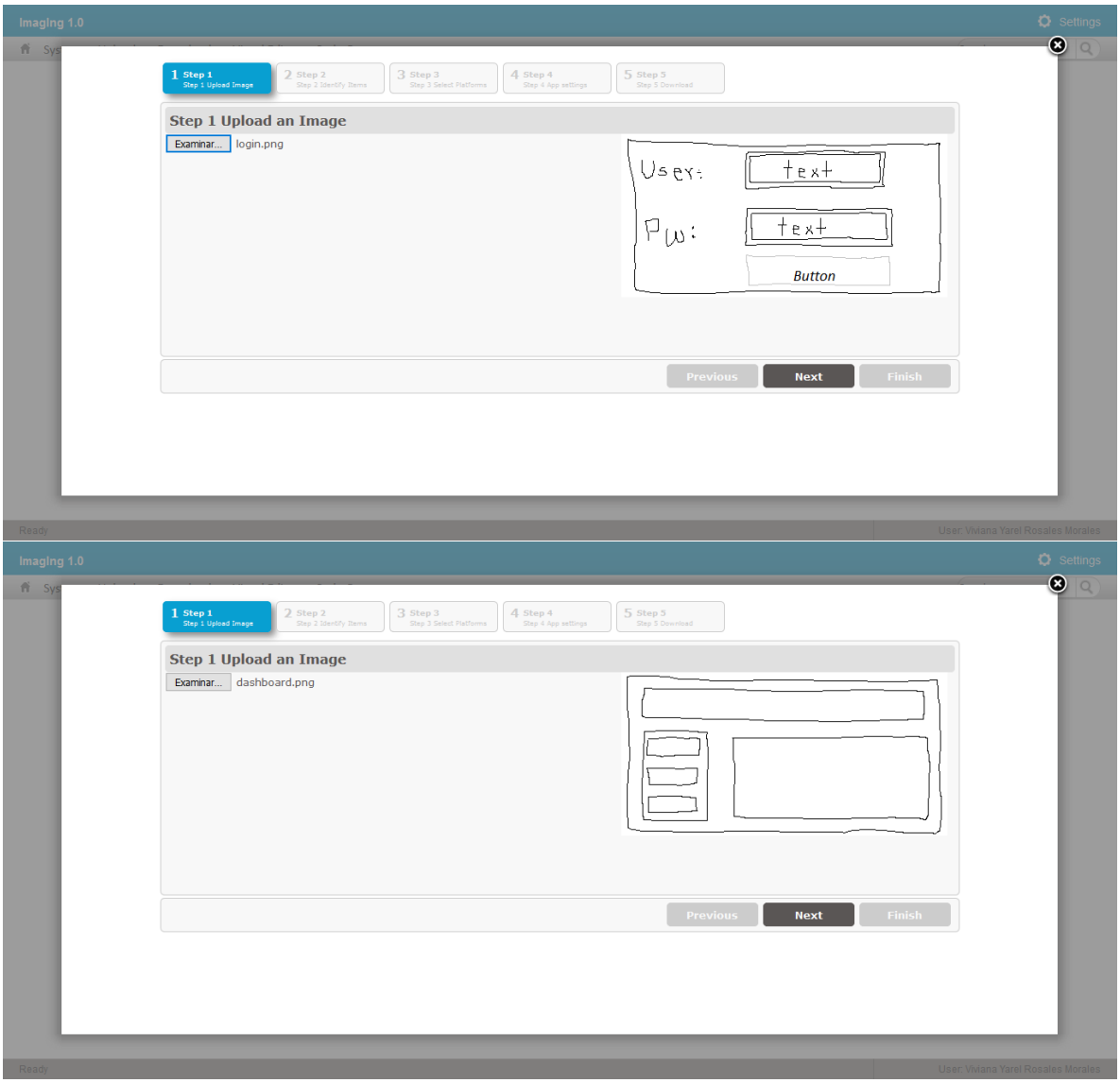


Figura 4.19. Paso 1 de la herramienta ImagIng: Subir imagen

En el paso 2 del *wizard*, se muestra los PDIUs identificados por la herramienta como se muestra en la Figura 4.20. Si el usuario está de acuerdo, es decir, que sean los UIDPs correctos, el proceso continúa.

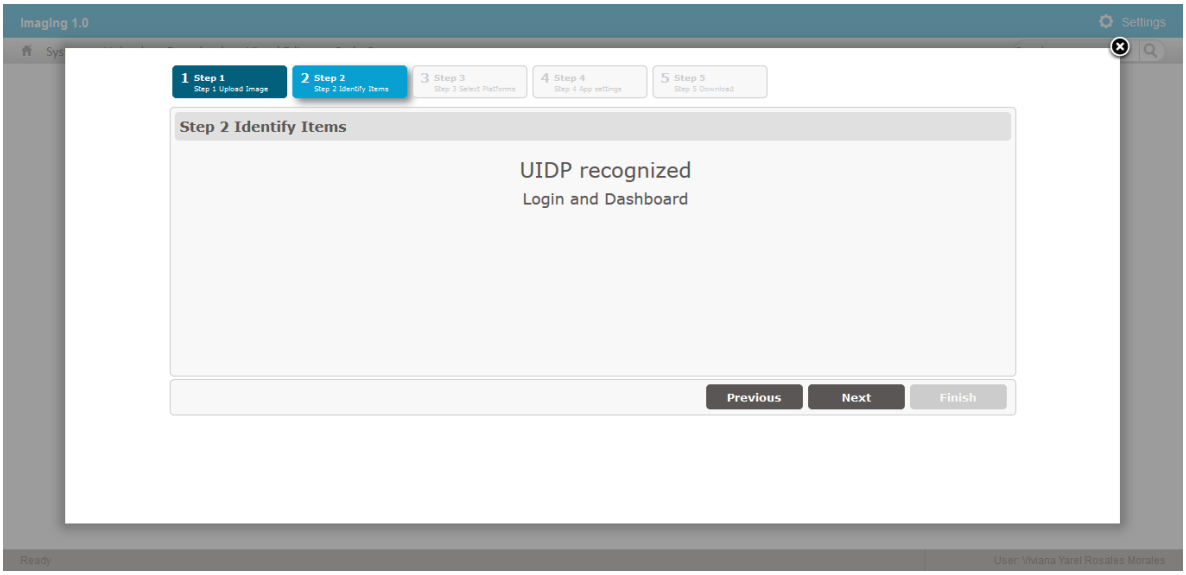


Figura 4.20. Paso 2 de la herramienta ImagIng: Identificación de los PDIUs

En el paso 3 del *wizard*, el usuario selecciona las plataformas deseadas para la aplicación, tal como se muestra en la figura 4.21. El usuario selecciona una o varias plataformas al mismo tiempo, en este caso como se requiere una aplicación para dispositivos móviles se seleccionaron todas las plataformas a excepción de Web.

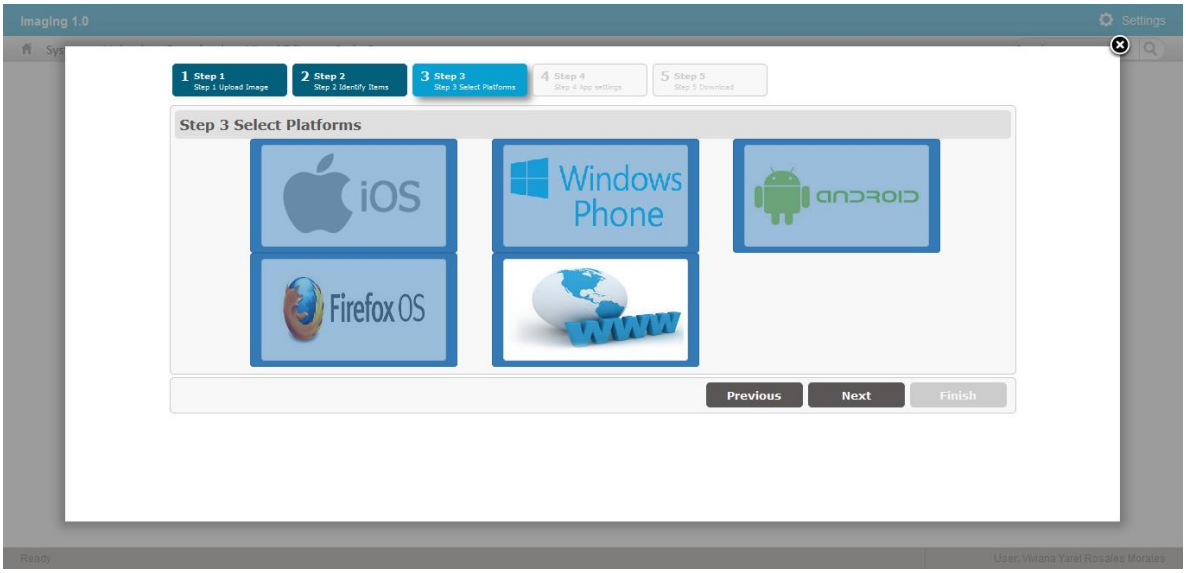


Figura 4.21. Paso 3 de la herramienta ImagIng: Selección de las plataformas

Después de seleccionar la o las plataformas, el usuario ingresa la configuración general de la aplicación. Este cuarto paso se muestra en la Figura 4.22., el usuario introduce la

configuración general de la aplicación, como el nombre de la aplicación a generar, del autor y de la institución, así como la orientación.

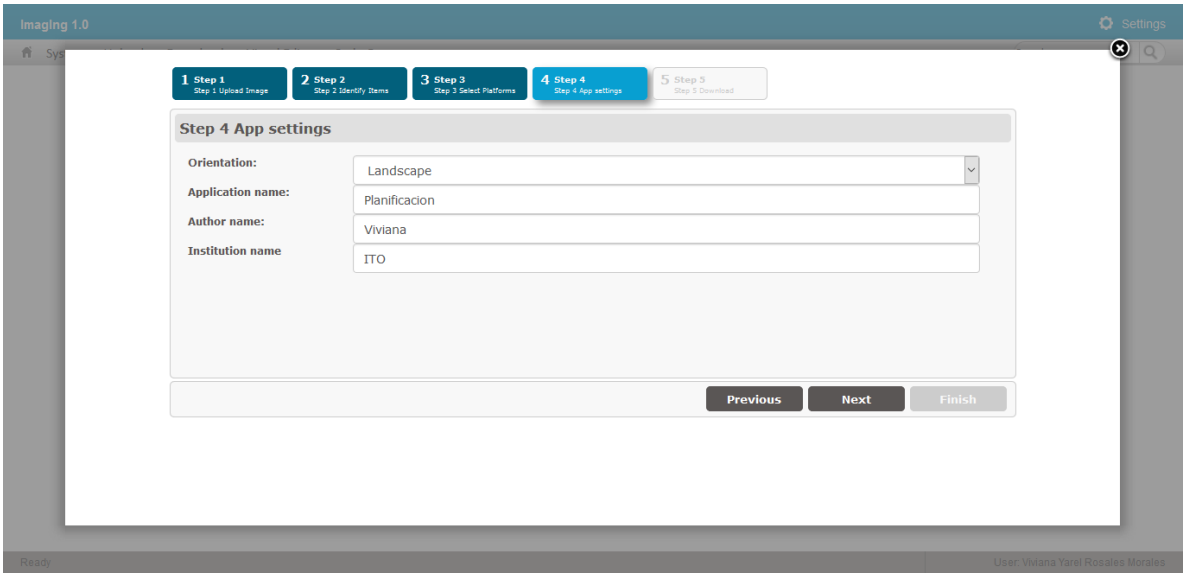


Figura 4.22. Paso 4 de la herramienta ImagIng: Configuración

Antes de pasar al paso 5 la aplicación pide confirmar los datos con los que será generada la aplicación tal como se muestra en la figura 4.23.

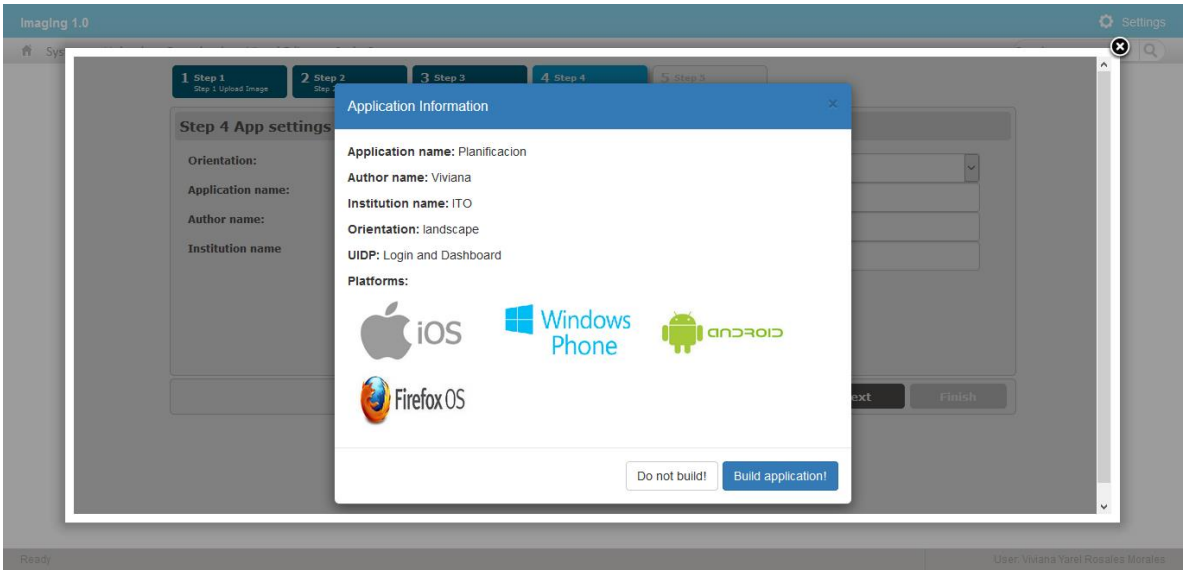


Figura 4.23. Paso 4 de la herramienta ImagIng: Información de la aplicación

A continuación, en el paso 5, el usuario descarga finalmente el código generado tal como se muestra en la figura 4.24. El código fuente se genera en el lenguaje de programación soportado por cada plataforma y en carpetas separadas, como se mencionó anteriormente.

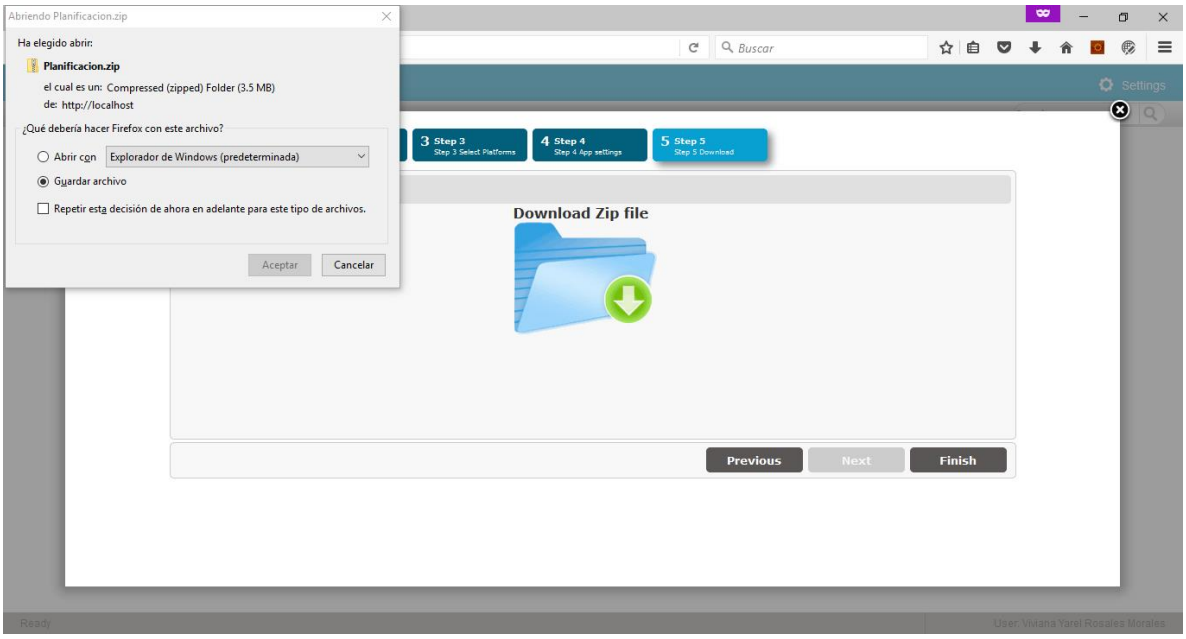


Figura 4.24. Paso 5 de la herramienta ImagIng: Descarga

En la figura 4.25 se muestran las carpetas que se generaron para este proyecto.

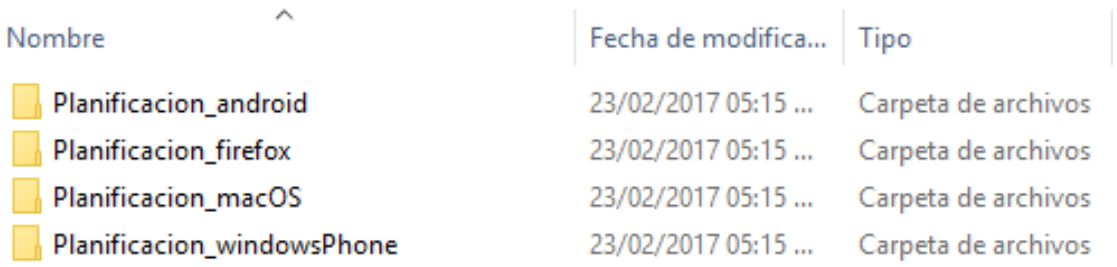


Figura 4.25. Carpetas generadas

Para desplegar la aplicación generada se requiere importarla en los IDEs correspondientes según sea el caso. A continuación, se muestra la implementación del proyecto en el IDE correspondiente para cada plataforma.

Implementación en Android™

Para implementar el proyecto generado en la plataforma Android™ es necesario abrir el proyecto existente en el IDE Android™ Studio, una vez cargados los archivos del proyecto tal como se muestra en la figura 4.26 se corre el proyecto dando clic en Run.

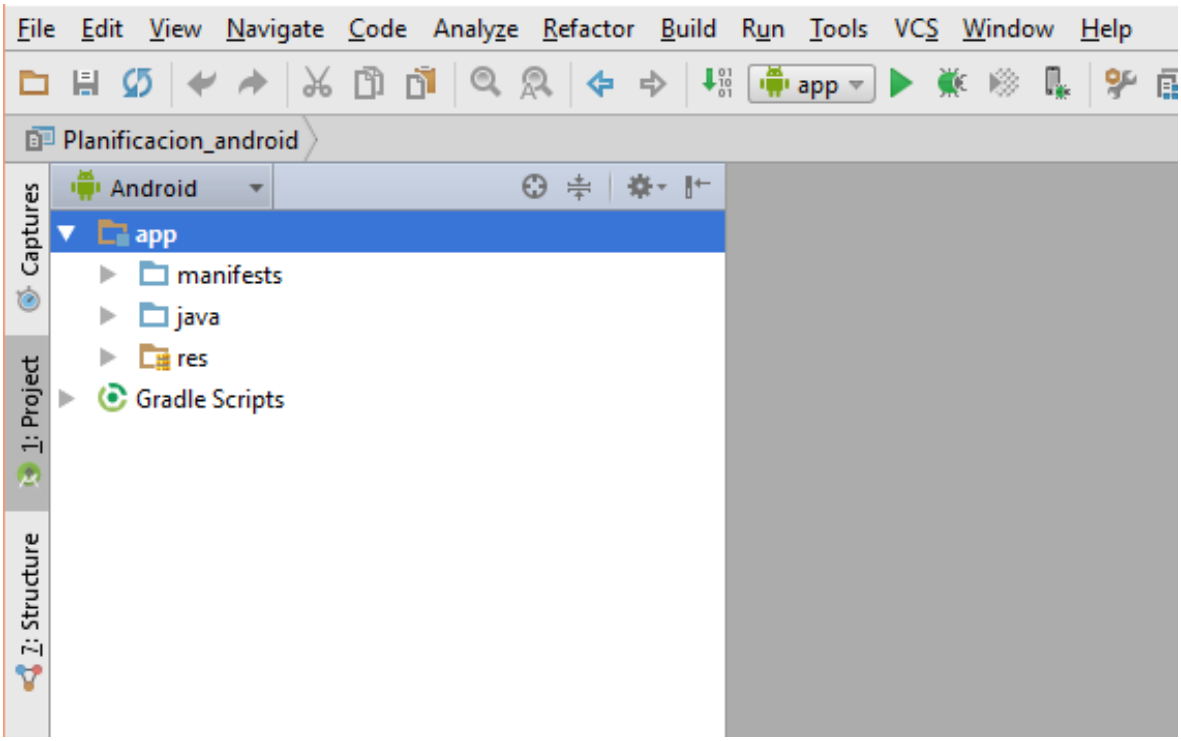


Figura 4.26. Proyecto en Android™

Cuando se ejecuta el proyecto se abre el emulador de Android™ y se despliega la primera interfaz que es el *login* tal como se muestra en la figura 4.27.

Figura 4.27. Interfaz *login* en Android™

Una vez autenticado en el sistema (con los datos por default), se despliega la segunda interfaz generada, el *dashboard*, tal como se muestra en la figura 4.28.

Figura 4.28. Interfaz *dashboard* en Android™

Implementación en Firefox® OS

En el caso de Firefox® OS se utiliza el IDE que proporciona el browser de Mozilla Firefox®, el Firefox® WebIDE y se abre la aplicación generada tal como se muestra en la figura 4.29, posteriormente se selecciona el emulador a utilizar y se inicia la ejecución.

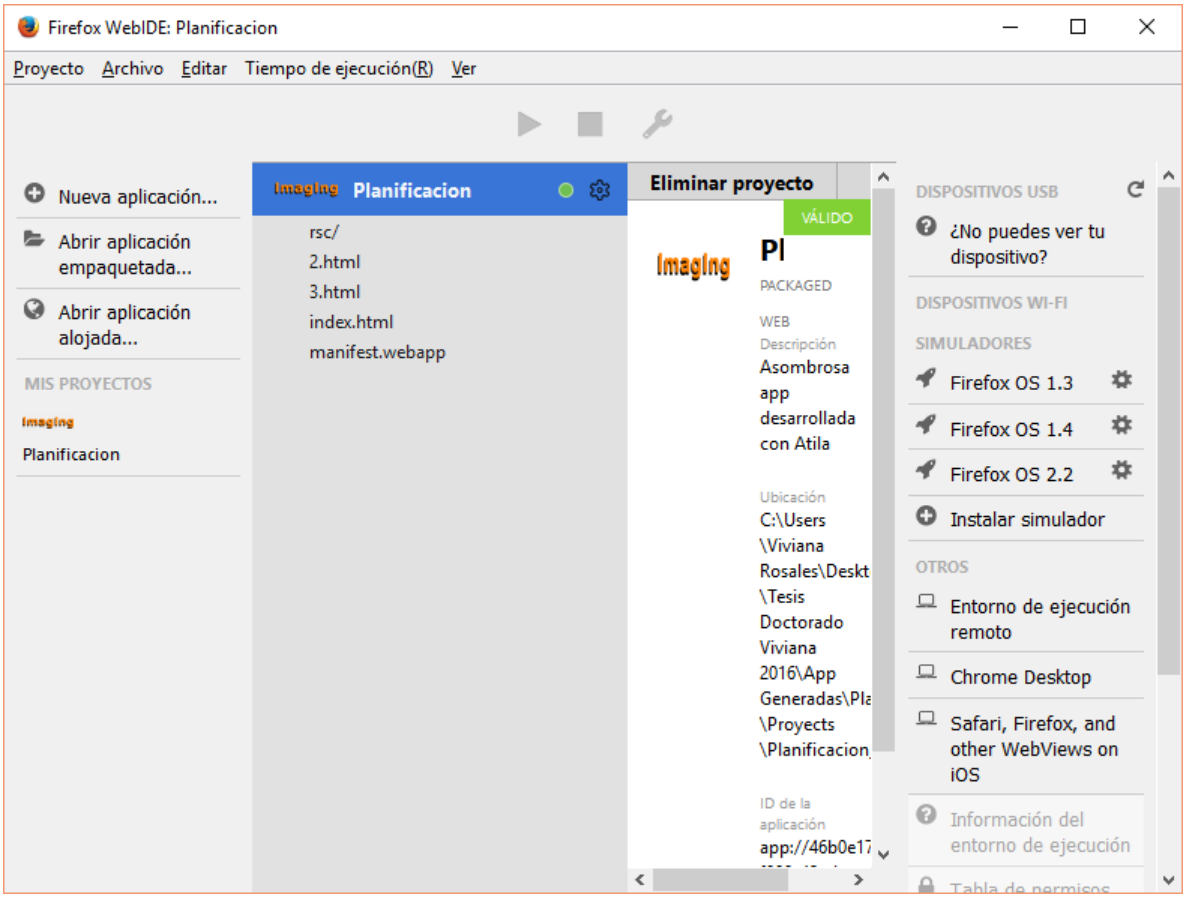


Figura 4.29. Proyecto en Firefox® OS

Se inicia la ejecución del proyecto y se abre el emulador de Firefox® OS, en la figura 4.30 se observa la aplicación instalada en el dispositivo del emulador.

Figura 4.30. Aplicación instalada en Firefox® OS

Posteriormente al iniciar la aplicación instalada se despliega la primera interfaz de la aplicación que es el login, tal como se muestra en la figura 4.31.

Figura 4.31. Interfaz *login* en Firefox® OS

Y por último al autenticarse correctamente en el *login* se despliega la segunda interfaz que es el *dashboard*, tal como se muestra en la figura 4.32.

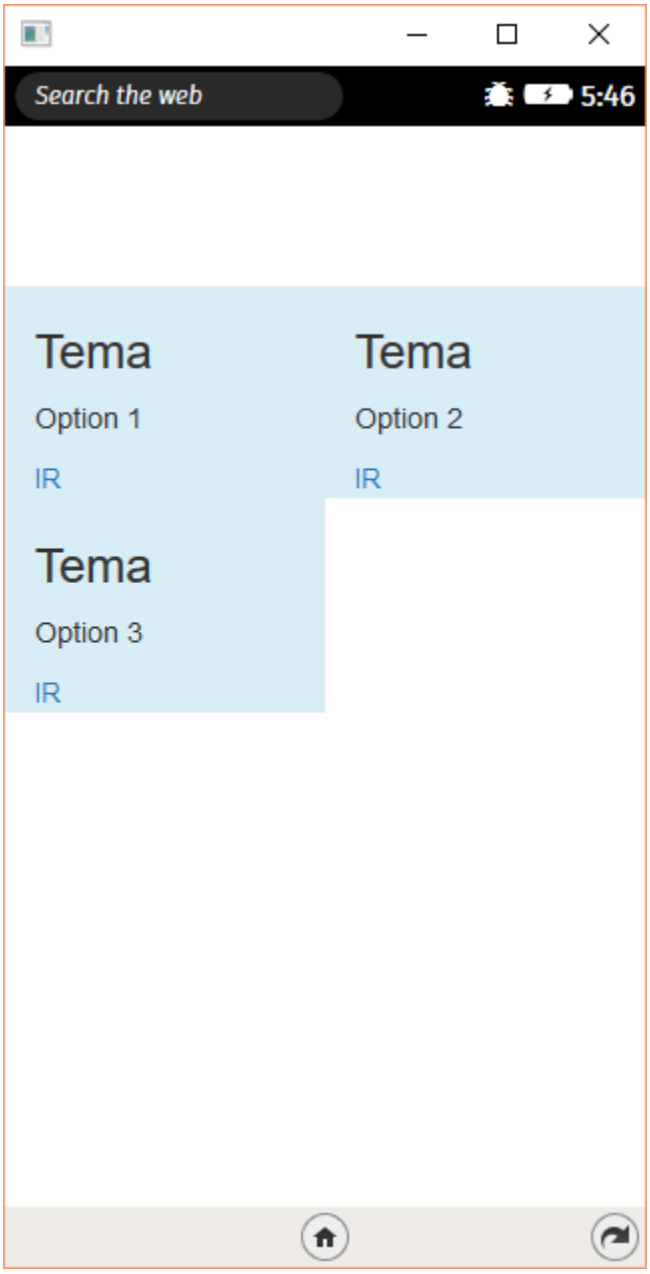


Figura 4.32. Interfaz *dashboard* en Firefox® OS

Implementación en MacOS® o iOS

Para el caso de MacOS® se requiere del IDE Xcode, dentro del cual se abre el proyecto generado y una vez cargados todos los archivos como se muestra en la figura 4.33 se ejecuta el proyecto.

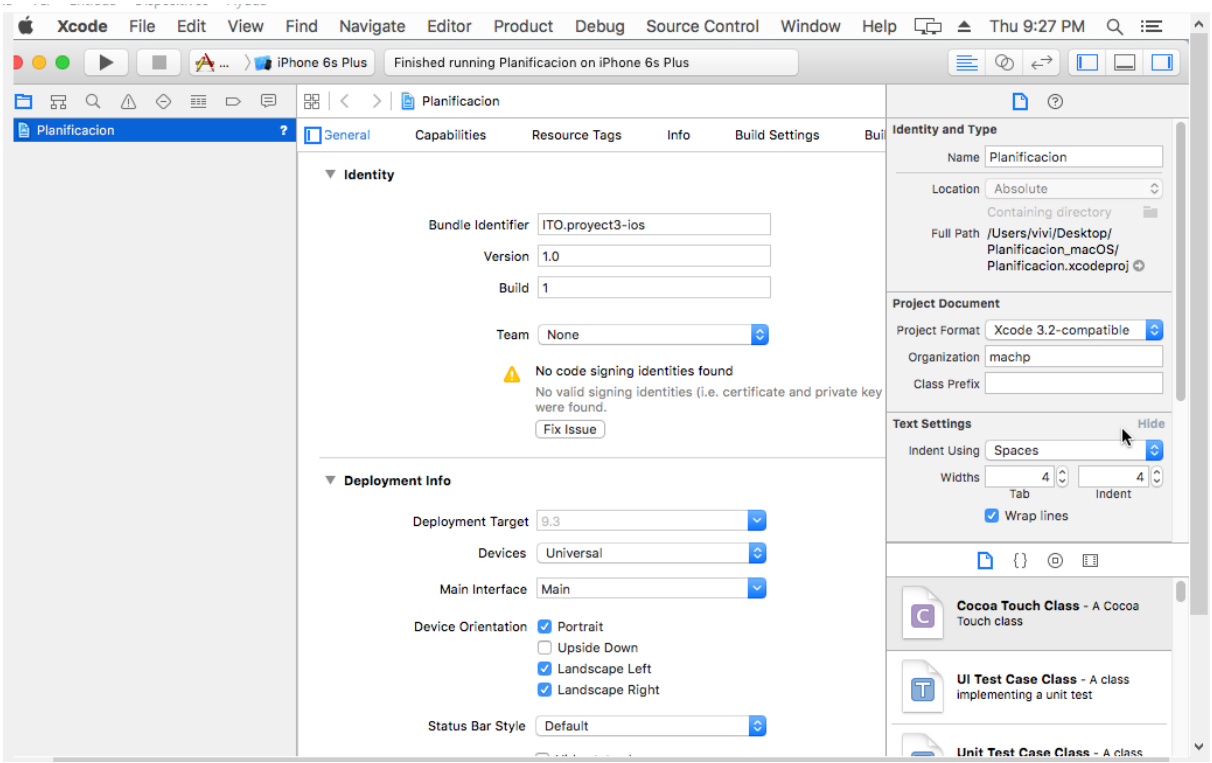


Figura 4.33. Proyecto en MacOS®

Al iniciar la aplicación en el emulador se despliega la primera interfaz como se muestra en la figura 4.34.

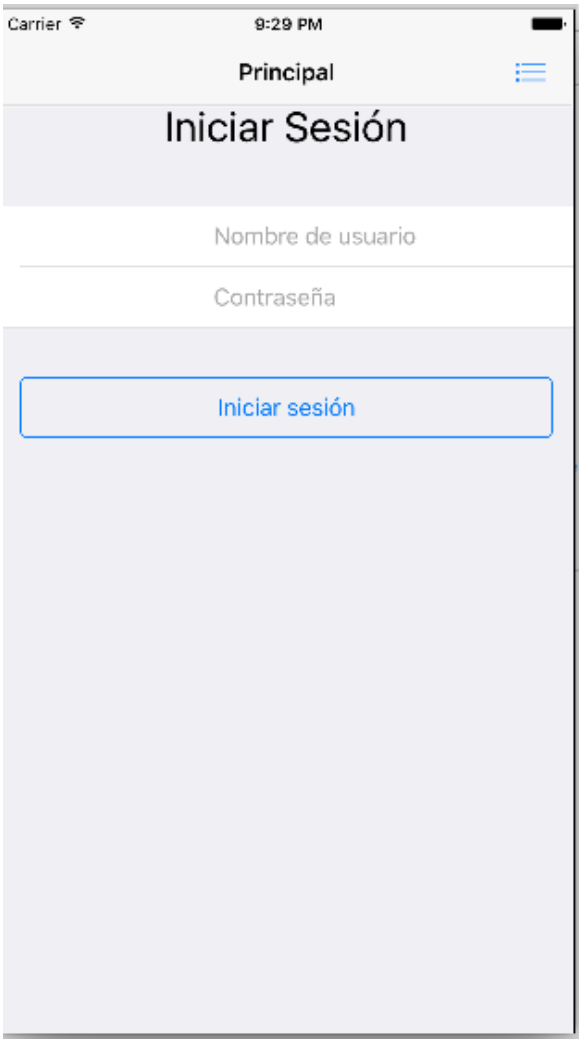


Figura 4.34. Interfaz *login* en MacOS®

Posteriormente al autenticarse correctamente en el *login* se despliega la segunda interfaz que es el *dashboard* tal como se muestra en la figura 4.35.

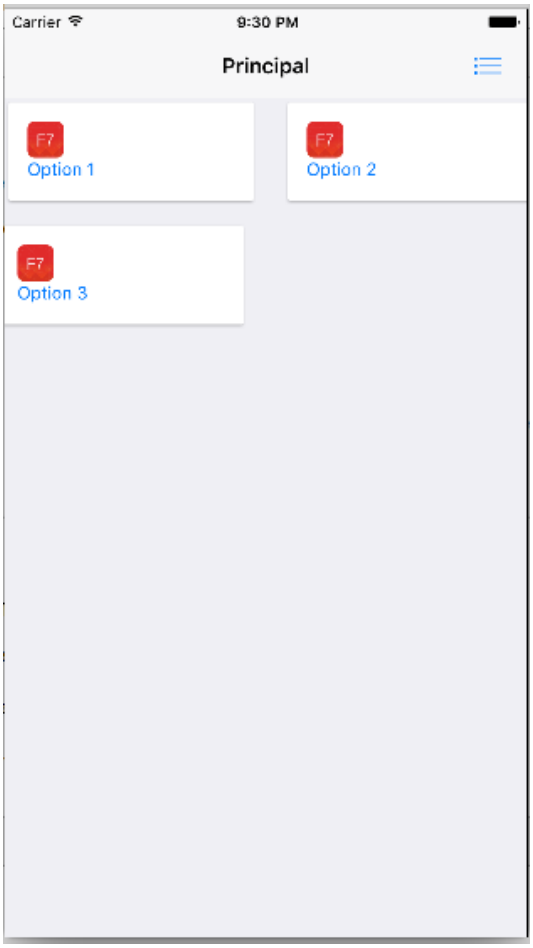


Figura 4.35. Interfaz *dashboard* en MacOS®

Implementación en Windows Phone®

Para el caso de Windows Phone® se require del IDE Microsoft® Visual Studio® en el cual se abre el proyecto generado y cargar lo archivos como se muestra en la figura 4.36, posteriormente se selecciona el emulador y se ejecuta el proyecto.

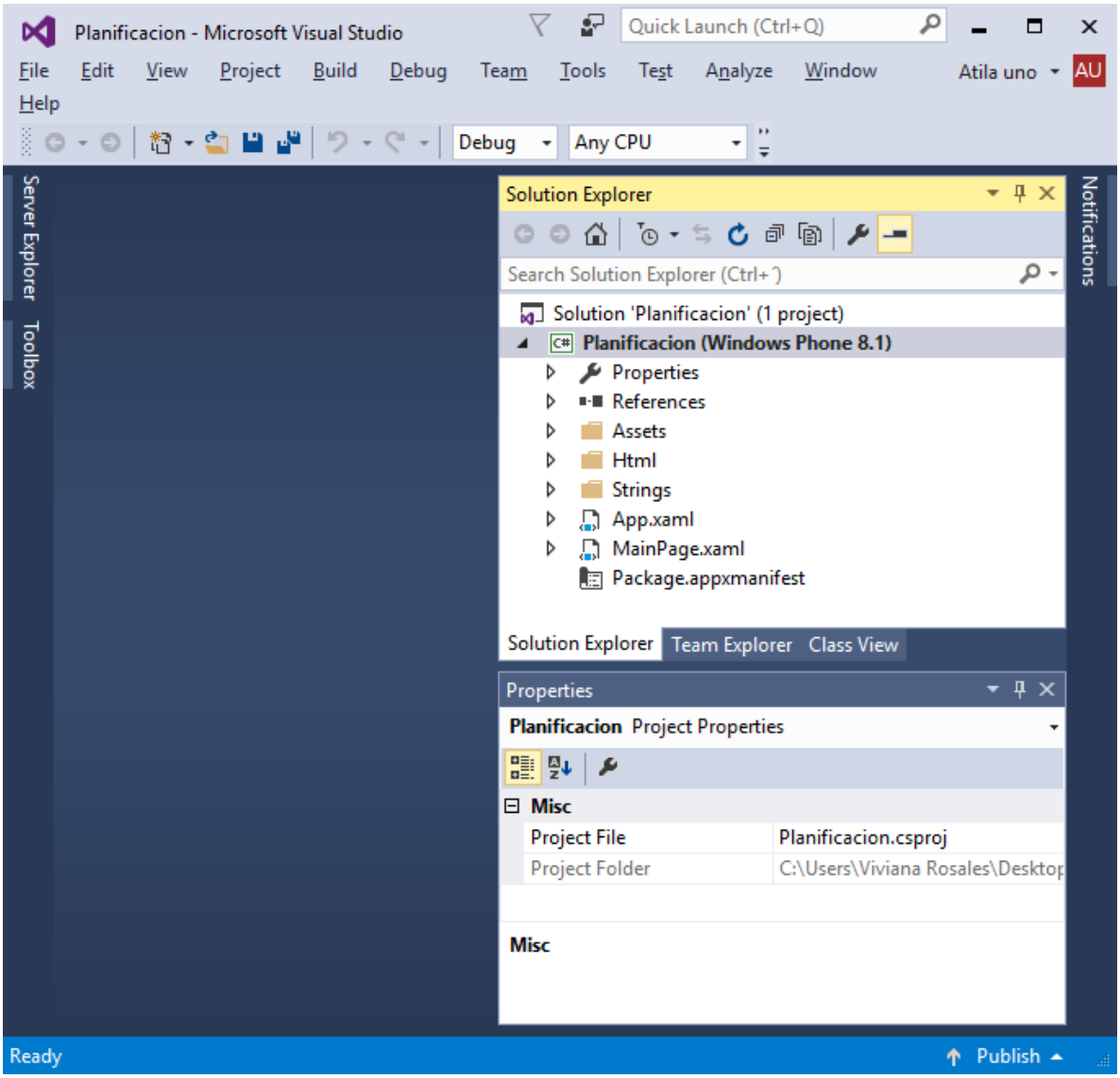


Figura 4.36. Proyecto en Window Phone™

Al iniciar el emulador se despliega la primera interfaz que es el login, como se muestra en la figura 4.37.

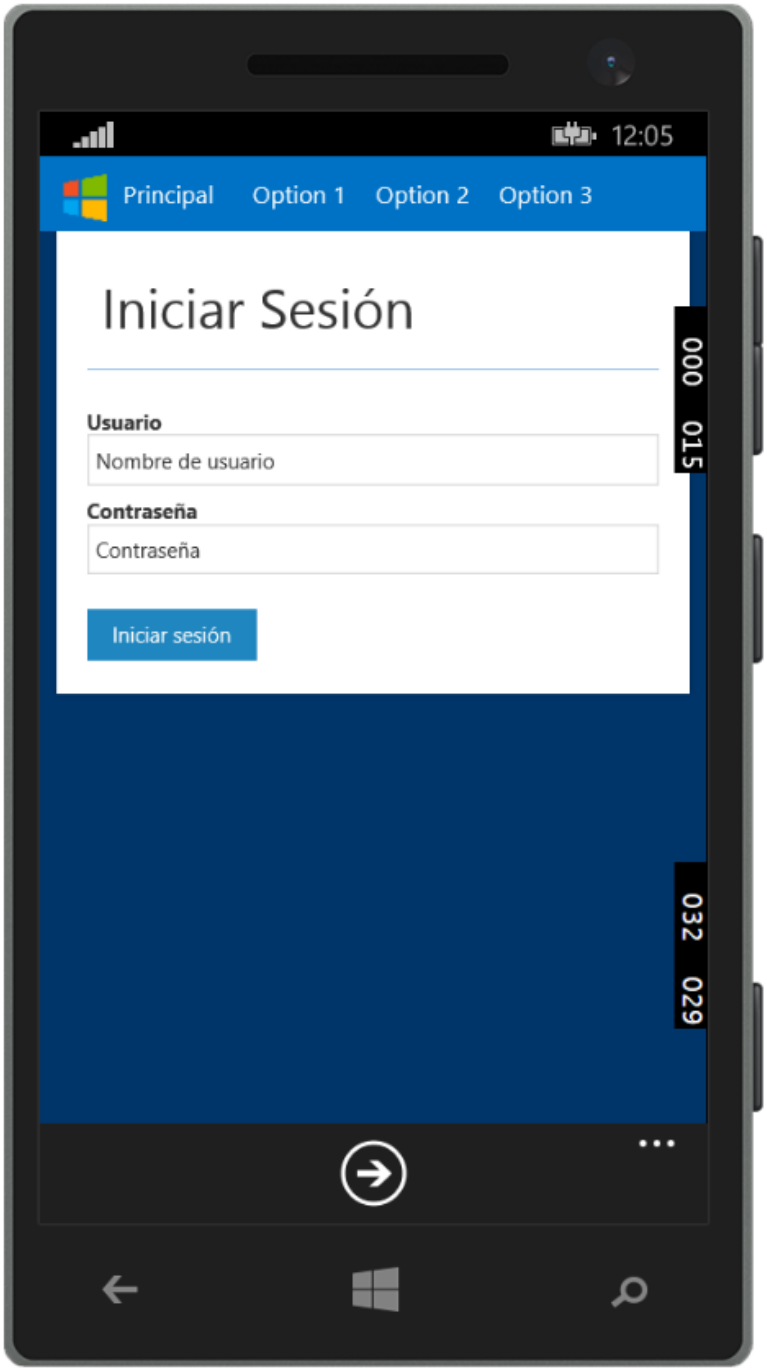


Figura 4.37. Interfaz *login* en Windows Phone®

Y posteriormente una vez que es autenticado correctamente en el login se despliega la segunda interfaz que es el dashboard, como se muestra en la figura 4.38.

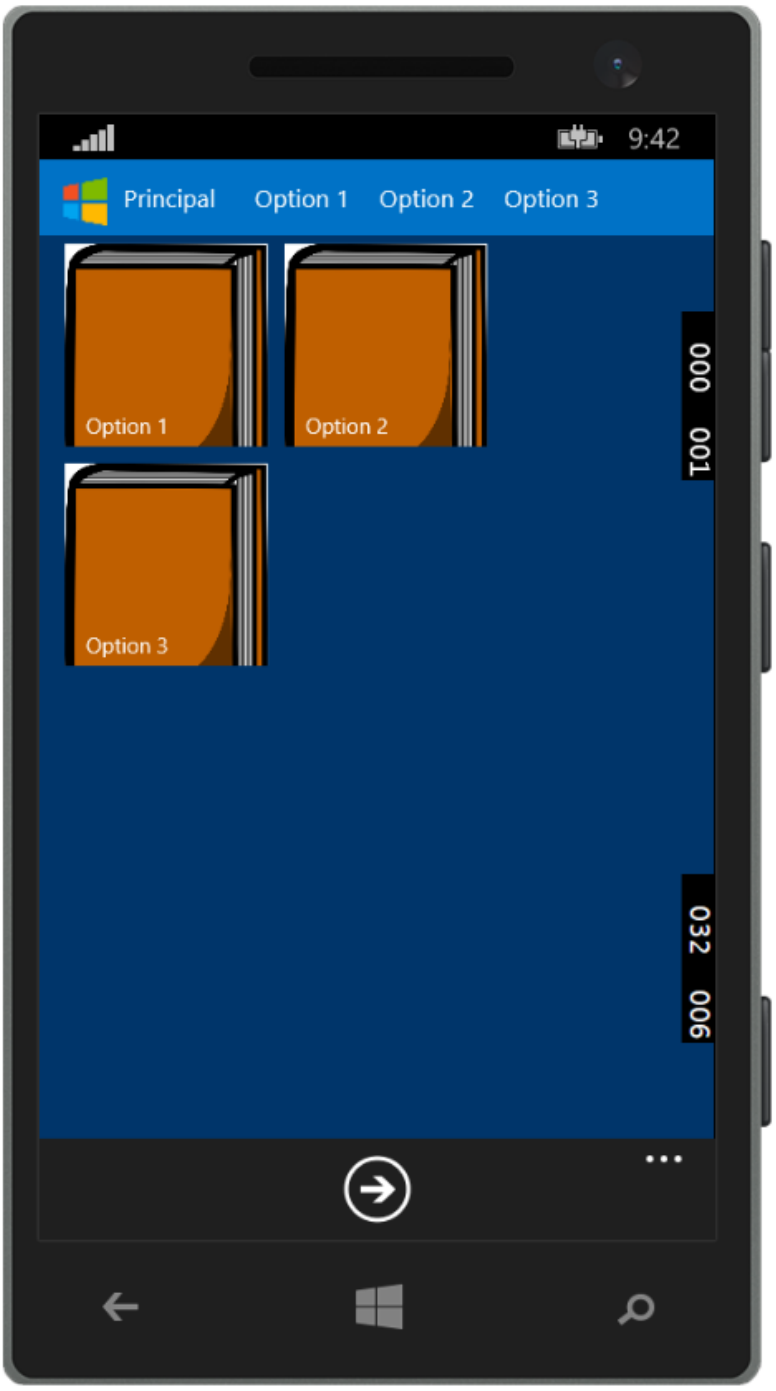


Figura 4.38. Interfaz *dashboard* en Windows Phone®

Con esto se comprueba que la aplicación fue generada correctamente para las 4 plataformas para dispositivos móviles seleccionadas.

4.4. Evaluación

La literatura en Ingeniería de Software ha reportado diversos métodos para evaluar las herramientas de software. En este sentido, Kitchenham, Linkman y Law (1997) propusieron tres métodos para la evaluación de software y herramientas: Métodos cuantitativos, Métodos cualitativos y Métodos híbridos.

La literatura describe el Análisis de Efectos Cualitativos y *Benchmarking* como un método híbrido considerando las diferentes formas en que los métodos / herramientas son evaluados. Los métodos híbridos tienen elementos tanto cuantitativos como cualitativos.
Siguiendo la clasificación de Kitchenham, Linkman y Law, se consideró el método de evaluación híbrida para evaluar tres herramientas de generación automática de código, incluyendo ImagIng, WebRatio (Acerbis et al. 2015), y AlexandRIA (Colombo-Mendoza et al. 2013). La evaluación tuvo como objetivo destacar y medir las características clave de cada herramienta desde perspectivas tanto cualitativas como cuantitativas. En la sección 4.3.1 se discute la evaluación cualitativa realizada para medir la usabilidad, soporte de características estándar y el cumplimiento de los principios de mejores prácticas. Para esta evaluación cualitativa, se identificaron diez características deseadas en las tres herramientas para evaluar su legitimidad y, al mismo tiempo, identificar cuál sería mejor en circunstancias específicas. Debido a que este análisis cualitativo requiere medidas subjetivas, se presentauna discusión detallada sobre los resultados obtenidos.

Por otro lado, en la sección 4.3.2 se discute la evaluación cuantitativa para medir la calidad en el uso de cada herramienta. Para esta evaluación, se propusieron dos métricas basadas en métricas definidas por el estándar ISO / IEC 9126. También, esta evaluación cuantitativa tomó como guía la evaluación reportada en (Colombo-Mendoza et al. 2014), además se midieron las características esenciales de la herramienta de generación de código identificadas por tres expertos en desarrollo de software: un diseñador de IU, un ingeniero de software y un desarrollador de aplicaciones para dispositivos móviles.

4.4.1. Evaluación Cualitativa

En esta sección, describimos las cuatro necesidades de usuario identificadas en el desarrollo de aplicaciones multiplataforma.

Necesidad (N) 1: Soporte para diversas arquitecturas de aplicaciones para dispositivos móviles

El tipo de aplicación para dispositivos móviles soportada es un aspecto crucial de cualquier herramienta de generación automática de código. Es necesario evaluar esta necesidad no sólo a partir de factores de independencia de tecnología y dominio, sino también considerando aspectos arquitectónicos. Hay tres tipos de arquitecturas de aplicaciones para dispositivos móviles: (a) aplicaciones nativas, (b) aplicaciones híbridas, construidas con tecnologías Web y envueltas en un contenedor de aplicaciones nativo específico del dispositivo, y (c) aplicaciones basadas en Web.

Característica (C) 1: Aplicaciones independientes del dominio

Característica (C) 2: Aplicaciones independientes de la plataforma

Necesidad (N) 2: Cobertura completa del ciclo de vida del desarrollo

El ciclo de vida del desarrollo incluye comúnmente: planificación, diseño, desarrollo, pruebas, publicación (o implementación) y mantenimiento. Esta necesidad o aspecto se aborda indirectamente en la necesidad descrita a continuación.

Característica (C) 1: Soporte de distribución de aplicaciones

Característica (C) 2: Control total sobre las aplicaciones

Necesidad (N) 3: Tiempo y ahorro de esfuerzo en el desarrollo

Compatibilidad con diversos lenguajes de programación y marcos de trabajo, así como instalaciones destinadas a evitar la codificación manual en la medida de lo posible.

Característica (C) 1: Uso de asistentes para realizar tareas complejas

Característica (C) 2: Generación automática de código fuente y nativo

Característica (C) 3: Soporte a diversos lenguajes de programación

Necesidad (N) 4: Herramientas de desarrollo compatibles con la usabilidad

En ISO 9241-11 (1998) se define la usabilidad como la medida en que un producto es utilizado por usuarios específicos para alcanzar objetivos específicos con eficacia, eficiencia y satisfacción en un contexto específico de uso. De acuerdo con esta definición, la usabilidad

es evaluada bajo factores de efectividad, eficiencia y satisfacción. En este sentido, Brooke (1996) propuso una escala de Likert de cinco puntos llamada escala de usabilidad simple (SUS: *Simple Usability Scale*), que da una visión global de las evaluaciones subjetivas de usabilidad. SUS permite evaluar la usabilidad de los sitios web con respecto a la satisfacción del usuario. Para evaluar la usabilidad en las tres herramientas, se propuso medir los siguientes aspectos cubiertos por el SUS: facilidad de uso, capacidad de aprendizaje y conocimientos y habilidades requeridos.

Característica (C) 1: Fácil de usar

Característica (C) 2: Fácil de aprender

Característica (C) 3: Conocimiento y habilidades de los desarrolladores

Diseño de la evaluación

En el método de evaluación híbrida propuesto, cada aspecto evaluado representaba una de las necesidades de un desarrollador de software. Además, cada aspecto (o necesidad) se compone de características clave, principalmente características de generación de código fuente, que las herramientas de desarrollo de software requieren poseer para satisfacer cada necesidad identificada. Para medir cada característica, se utilizó una escala de Likert de cinco puntos (Likert, 1932), en la que 5 representaron la mejor puntuación posible, y 1 representa la puntuación más baja posible bajo la siguiente evaluación:

1. Totalmente en desacuerdo
2. Desacuerdo
3. Neutro
4. De acuerdo
5. Totalmente de acuerdo

La evaluación general de cada herramienta de software fue la suma de todas las puntuaciones de las características.

Resultados

La tabla 4.1 y la figura 4.39 presentan las puntuaciones obtenidas a partir de la evaluación cualitativa de cada característica de la herramienta.

Tabla 4.1. Resultados de la evaluación cualitativa

Aspecto		WebRatio	AlexandRIA	ImagIng
N1	*C1*	5	2	5
	C2	5	1	5
N2	C1	5	5	5
	C2	5	5	5
N3	C1	5	5	5
	C2	5	4	5
	C3	5	1	5
N4	C1	4	4	5
	C2	4	3	5
	C3	4	3	4
SUMA		47	33	49

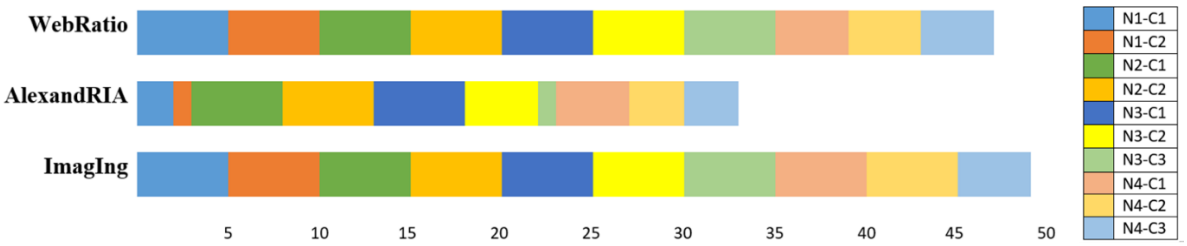


Figura 4.39. Resultados de la evaluación cualitativa

Después de la evaluación cualitativa para el aspecto N1-C1, se encontró que WebRatio e ImagIng permiten construir aplicaciones para dispositivos móviles independientes del dominio, ya que ofrecen integración con una amplia gama de servicios y son plataformas de software de propósito general. En contraste, AlexandRIA no es una plataforma de software de propósito general, ya que sólo genera aplicaciones enriquecidas de Internet (RIAs). Aunque es posible que las RIAs sean de diferentes dominios, AlexandRIA se limita a generar este tipo de aplicación. Por otro lado, sobre el aspecto N1-C2, las tres herramientas permiten a los desarrolladores utilizar estándares Web como HTML5, JavaScript y CSS3 para construir aplicaciones para dispositivos móviles basadas en Web e incluso aplicaciones para dispositivos móviles híbridas. En este sentido, las aplicaciones Web resultantes no requieren un complemento adicional del navegador Web ni un entorno de tiempo de ejecución para ejecutarse. Sin embargo, las aplicaciones para dispositivos móviles híbridas y nativas

funcionan de manera diferente. En el caso de las aplicaciones nativas generadas con ImagIng, es necesario utilizar un IDE para cada plataforma, y luego compilar y empaquetar el código, pero el código se generará para todas las plataformas soportadas. A diferencia de ImagIng, AlexandRIA sólo genera código para Adobe Flex™, lo que la hace dependiente de la plataforma.

En cuanto a N2-C1, todas las herramientas proporcionan mecanismos para la distribución de aplicaciones nativas. ImagIng proporciona el código para diferentes plataformas listas para ser importadas en el IDE correspondiente y para ser compiladas y empaquetadas para generar aplicaciones ejecutables. En cuanto a N2-C2, todas las plataformas integran herramientas de gestión que proporcionan funciones, como la administración de usuarios, lo que permite a los desarrolladores controlar las aplicaciones desplegadas. En el caso de ImagIng es posible modificar el código generado según las necesidades del desarrollador en el IDE correspondiente, sin limitaciones por parte del generador de código, ya en este punto las modificaciones que se hagan al código son responsabilidad del desarrollador.

Además, las tres propuestas integran herramientas de asistente (*wizard*) para automatizar tareas o como una forma alternativa y más rápida de realizar tareas de usuario complejas (aspecto N3-C1). Por ejemplo, WebRatio ofrece un asistente de un solo paso para seleccionar el tipo de aplicación y la configuración. ImagIng proporciona una herramienta de asistente única para las fases de diseño, desarrollo, despliegue y mantenimiento del ciclo de vida del desarrollo. Por otra parte, con respecto al aspecto N3-C2, las tres herramientas cuentan con funciones de generación de código. WebRatio utiliza el framework Apache Cordova™ para empaquetar el código fuente y generar el código para Android™ e iOS, mientras que Alexandria utiliza el framework PhoneGap™ para aplicaciones de empaquetado y genera el código fuente para Android™, iOS, BlackBerry Tablet OS™ y Windows Phone®. Por último, ImagIng genera código fuente nativo y código HTML5 para los sistemas operativos Android™, iOS, Windows Phone® y Firefox® OS y Web. En cuanto al aspecto N3-C3, las tres herramientas son independientes de la tecnología, ya que soportan más de un lenguaje de programación. WebRatio proporciona soporte para Android™ (utilizando Java) e iOS (utilizando Objective C) con Apache Cordova™. AlexandRIA proporciona soporte para HTML, JavaScript, JSP, MXML, ActionScript y PHP usando PhoneGap™. Por último,

ImagIng proporciona soporte para Android™ (Java), iOS (Swift), Windows Phone® (XAML y HTML5), y Firefox® OS y Web (HTML5).

En cuanto a N4-C1, gracias a la generación automática de código utilizando técnicas de procesamiento de imágenes, ImagIng parece ser más intuitivo que las otras dos herramientas. Sin embargo, una extensa documentación técnica podría ayudar a los desarrolladores a superar la posible complejidad detrás de WebRatio y AlexandRIA. En cuanto al aspecto N4-C2, la curva de aprendizaje de WebRatio es mayor que las curvas de aprendizaje de ImagIng y AlexandRIA. Más específicamente, tanto WebRatio como AlexandRIA proporcionan asistentes y pantallas de configuración para facilitar el desarrollo de aplicaciones; sin embargo, sólo ImagIng proporciona un solo asistente en el que sólo se proporciona una imagen, y con una configuración mínima, es posible obtener aplicaciones para todas las plataformas soportadas. En otras palabras, pocos clics son suficientes para que el usuario pueda obtener aplicaciones funcionales en ImagIng sin ser un experto en ninguna de estas plataformas. Según la escala SUS, la integración de las funcionalidades del sistema es un problema al evaluar la usabilidad. En este sentido, los resultados de la evaluación mostraron que las funciones de ImagIng están mejor integradas. Por último, con respecto al aspecto N4-C3, la evaluación cualitativa indica que tanto WebRatio como AlexandRIA requieren conocimientos de lenguajes de programación para el desarrollo exitoso de aplicaciones, mientras que ImagIng no es exclusivo para usuarios experimentados. Para desarrollar una aplicación en ImagIng, el usuario sólo necesita proporcionar una imagen de entrada. Sin embargo, esta propiedad no se considera completamente abordada, ya que los usuarios de ImagIng también necesitan importar las aplicaciones generadas a los IDEs correspondientes para empaquetar y ejecutar las aplicaciones, y esto implica cierto esfuerzo.

En conclusión, ImagIng cumple con más del 90% de las características que un generador de código fuente requiere poseer para satisfacer las necesidades de los usuarios en términos de independencia de tecnología y dominio, cobertura de ciclo de vida de desarrollo, complejidad de migración, soporte y documentación y usabilidad. ImagIng calificó a 49 de 50 en la evaluación aquí propuesta. De acuerdo con estos resultados, ImagIng no requiere conocimientos o habilidades en el desarrollo de aplicaciones, sobre todo hablando de

aplicaciones para dispositivos móviles, como es el caso de WebRatio y AlexandRIA. Además, ImagIng es más fácil de usar y más fácil de aprender.

4.4.2. Evaluación Cuantitativa

El objetivo de este método de evaluación es medir la calidad de las herramientas de generación de código fuente, WebRatio, AlexandRIA e ImagIng. Para ello, se seleccionó el modelo de calidad de software ISO / IEC 9126. En detalle, ISO / IEC 9126 es un estándar internacional para la evaluación de la calidad del software; Define un modelo compuesto por características generales de software que se descomponen adicionalmente en sub-características generales y atributos específicos. Además, ISO / IEC 9126 define tres conjuntos de métricas de software para evaluar atributos desde tres perspectivas diferentes: (1) calidad externa, (2) calidad interna y (3) calidad en uso. Gracias a su naturaleza genérica, es posible refinar el modelo de calidad ISO / IEC 9126 dependiendo del tipo de software que se va a evaluar. En este sentido, es factible utilizar el modelo de calidad ISO / IEC 9126 para evaluar los generadores de código fuente.

Diseño de la evaluación

Para la evaluación cuantitativa de las herramientas de generación de código fuente, se seleccionaron atributos con sub-características de la norma ISO / IEC 9126. Los componentes seleccionados fueron: precisión y utilización de recursos, ambos sub-característicos de la norma ISO / IEC 9126. A continuación se describen los factores de calidad evaluados y las métricas de software correspondientes.

Precisión del servicio (*Service accuracy*): La norma ISO / IEC 9126 define "precisión" como un factor de calidad de software que afecta a la característica de "funcionalidad". Aquí, la funcionalidad se interpreta como el conjunto de funciones y propiedades del software que satisfacen las necesidades indicadas. Por lo tanto, la "precisión" se define como la corrección del software con respecto a las necesidades o requerimientos implícitos. Puesto que ImagIng implementa un enfoque de generación de código automático basado en un proceso de desarrollo de software utilizando técnicas de procesamiento de imágenes, se utilizó la precisión del servicio para medir la corrección de las funciones generadas automáticamente

con respecto a los requisitos del usuario. La precisión del servicio es un factor de calidad de las aplicaciones de ImagIng.

Utilización de los recursos de hardware: Según el modelo de calidad descrito por la norma ISO / IEC 9126, la "utilización de los recursos" es un factor de calidad del software. Formalmente, es un componente de "eficiencia" dentro de la característica de calidad, y se interpreta como la capacidad del software para ejecutar tareas requeridas con la menor cantidad de recursos.

Para medir los factores de calidad antes mencionados, se emplearon tres métricas de software basadas en métricas internas de calidad de ISO / IEC 9126. La Tabla 4.2 muestra estas métricas. Sin embargo, en esta evaluación cuantitativa sólo se tuvo en cuenta WebRatio e ImagIng, ya que AlexandRIA no está disponible comercialmente.

Tabla 4.2. Métricas de calidad para la evaluación cuantitativa

Factor de calidad	Métrica	Descripción
Precisión del servicio	Relación defecto-característica	Relación entre el número de defectos descubiertos y el número de características desarrolladas.
Utilización de los recursos de hardware	Uso de RAM	Cantidad promedio de memoria utilizada por el servicio.
	Uso de CPU	Porcentaje promedio de CPU utilizado por el servicio.
	Uso de ancho de banda	Promedio de ancho de banda utilizado por el servicio.

Como se muestra en el caso estudio, la herramienta ImagIng genera el código fuente para diversas aplicaciones Web y móviles; Sin embargo, para aplicar el mismo contexto en WebRatio e ImagIng, se realizó la evaluación en el desarrollo de una interfaz de inicio de sesión (*login*), ya que la función de inicio de sesión se utiliza en casi cualquier aplicación.

Resultados

En el escenario de evaluación del inicio de sesión, los usuarios de WebRatio necesitan descargar la herramienta "WebRatio Mobile Platform" en Community Edition e instalarla en el equipo. Después de instalarla y abrir la herramienta se crea el espacio de trabajo y está

listo para usar, se selecciona un nuevo proyecto móvil, al abrir un nuevo proyecto, WebRatio pregunta si el usuario desea agregar algún PDIU, entre los cuales está el *login*, que es el que se seleccionó. En este punto es posible realizar modificaciones al proyecto de WebRatio, aunque esta etapa resulta algo confusa, ya que WebRatio ofrece un asistente para construir el proyecto, pero no se proporciona soporte adicional. Una vez que el desarrollador está satisfecho con el proyecto, hacen clic en el botón "Generar y ejecutar", y la aplicación se despliega en un emulador Web, también provee la opción de empaquetar el código utilizando el marco de trabajo Apache Cordova™. A continuación, el emulador ofrece dos opciones para acceder a la aplicación para dispositivos móviles: a través del código QR, que se escanea con una aplicación previamente instalada en el dispositivo móvil o introduciendo una dirección IP proporcionada por el propio emulador. En cuanto a ImagIng, el usuario accede al sitio Web de ImagIng y proporcionar una imagen que represente una interfaz de usuario de inicio de sesión. A continuación, el usuario selecciona las plataformas para las que se generará el código y se realizarán las configuraciones requeridas. Finalmente, se descarga el código generado, que se importa al IDE correspondiente para generar aplicaciones ejecutables en los diferentes dispositivos móviles.

Para estimar el esfuerzo requerido en el proceso de desarrollo de aplicaciones bajo el escenario de inicio de sesión usando ImagIng y WebRatio, sólo se evaluaron los principales pasos realizados manualmente en ambos procesos de desarrollo, se omiten pasos que son solo informativos. En el caso de la WebRatio, se consideraron los siguientes pasos: (1) Configurar el entorno IDE de WebRatio Mobile Platform para el desarrollo de aplicaciones Web móviles, (2) Construir un nuevo proyecto para dispositivos móviles, (3) Editar el código fuente para agregar la funcionalidad requerida (*login*), (4) Optimizar la página resultante para dispositivos móviles, (5) Emulador de Access Web, (6) Acceder a la aplicación para dispositivos móviles a través de un código QR o dirección IP. Por otro lado, se evaluaron los siguientes pasos de la herramienta ImagIng (1) Cargar una imagen que representa una interfaz de usuario de inicio de sesión, (2) Seleccionar plataformas, (3) Establecer configuración de desarrollo e implementación, (4) Descargar el código generado. La Tabla 4.3 resume los resultados obtenidos.

Tabla 4.3. Resultados de la evaluación de la precisión de servicios

Herramienta	Número de pasos	Tiempo requerido
WebRatio	6	~6 min
ImagIng	4	~3 min

Como se observa, el proceso de desarrollo de aplicaciones basado en WebRatio consta de seis pasos principales, mientras que el proceso de desarrollo de aplicaciones basado en ImagIng se compone de cuatro pasos principales. Además, el tiempo necesario para completar el proceso de desarrollo es de unos seis minutos en WebRatio y tres minutos en ImagIng aproximadamente. No se consideró la fase de mantenimiento debido a la naturaleza del escenario de evaluación.

La exactitud de las solicitudes en el escenario de evaluación se midió desde el punto de vista del consumidor. De acuerdo con los requisitos establecidos como parte del escenario de evaluación, se consideraron las siguientes tres funcionalidades: (1) ejecución de la aplicación, (2) inicio de sesión del usuario y (3) autenticación del usuario. La Tabla 4.4 resume los resultados de la evaluación.

Tabla 4.4. Resultados de la estimación del tiempo y esfuerzo necesarios para desarrollar una nueva aplicación

Herramienta	Número de funcionalidades desarrolladas	Número de funcionalidades requeridas	Proporción
WebRatio	3	3	3/3
ImagIng	3	3	3/3

Ambas herramientas cumplen las métricas establecidas, ya que no se encontraron errores. Dado que WebRatio es una aplicación de escritorio e ImagIng es una aplicación basada en Web, el rendimiento y los recursos de hardware necesarios varían. Por lo tanto, se realizaron las siguientes pruebas para determinar la utilización de recursos en cada herramienta. La Tabla 4.5 resume los resultados obtenidos.

Tabla 4.5. Resultados de la estimación de la utilización de los recursos de hardware

Herramienta	CPU	RAM	Ancho de banda
WebRatio	911 ms	11.5 MB	225 KB
ImagIng	447 ms	6.45 MB	177 KB

De acuerdo con los resultados resumidos anteriormente, generar una aplicación en ImagIng utiliza menos RAM y CPU (tiempo de ejecución) que en WebRatio. Los resultados de utilización de CPU cubren el tiempo empleado en la ejecución, mientras que los resultados de utilización de RAM muestran la cantidad de RAM consumida en las tareas de carga, secuencias de comandos y representación. Por último, los resultados de utilización del ancho de banda muestran el tamaño total de los archivos transferidos a través de la red, incluidos los archivos CSS, HTML y JS y las imágenes. Es necesario tener en cuenta que WebRatio es una aplicación de escritorio que utiliza un *browser* para ejecutar el emulador de aplicaciones.

Capítulo 5: Conclusiones

5.1. Conclusiones

ImagIng representa un nuevo enfoque para el desarrollo automático de aplicaciones multiplataforma mediante técnicas de procesamiento de imágenes. ImagIng es fácil de aprender e intuitivo, ya que no requiere modelado y no requiere mucho tiempo en la fase de diseño. El único requerimiento de ImagIng es una imagen de entrada que representa una interfaz de usuario y con esto es capaz de generar el código fuente para las aplicaciones que se desplegarán a través de diferentes plataformas para dispositivos móviles y Web. Los resultados muestran que, como herramienta para el desarrollo automático de aplicaciones multiplataforma, ImagIng reduce enormemente el tiempo de desarrollo de aplicaciones, disminuyendo así los costos de producción. En pocos pasos, los usuarios obtienen todas las interfaces necesarias para desarrollar una aplicación funcional, la cual también es posible implementar en todas las plataformas disponibles en la propia herramienta.

ImagIng cambia el esquema de desarrollo de software en la ingeniería de software, ya que acelera el ciclo de vida de desarrollo de software, al menos en la etapa de planificación. En este sentido, los usuarios dibujan directamente las interfaces a mano alzada, mismas que luego se generan de forma rápida y automática utilizando ImagIng. Además, debido a las diferencias entre las imágenes de entrada que representan un mismo PDIU, se hace uso de técnicas de inteligencia artificial para mejorar la funcionalidad de ImagIng; específicamente, se basa en las capacidades de una red neural artificial para reconocer tales diferencias y encontrar los PDIUs representados en las imágenes procesadas.

Para ejemplificar la funcionalidad y las capacidades de ImagIng, se realizó una evaluación híbrida, que objetivamente analizó ImagIng y dos herramientas de desarrollo de aplicaciones multiplataforma similares. La evaluación tuvo por objetivo evaluar las características clave de las tres herramientas y mostró que ImagIng presentaba ventajas atractivas con respecto a herramientas de desarrollo de aplicaciones multiplataforma similares. Tales ventajas se describen en términos de usabilidad, capacidad de aprendizaje y una menor utilización de recursos computacionales.

Por lo tanto el utilizar la herramienta ImagIng representa diversas ventajas con respecto a los procesos tradicionales, lo cual garantiza disminución del tiempo de desarrollo y de errores en el código de las aplicaciones de software, esto porque el código generado por la herramienta ImagIng ya se evaluó y aprobó anteriormente, y solo es reutilizable cada vez que requiera desarrollarse un componente o PDIU en particular.

Cabe mencionar la importancia de la escalabilidad de la herramienta ImagIng, ya que por el momento son 5 los PDIU soportados pero es posible agregar nuevos PDIU para incrementar las capacidades de la herramienta y su éxito de manera comercial.

5.2. Trabajo a futuro

Como trabajo futuro, se buscará agregar más PDIUs, tales como pestañas de navegación, paginación, galerías y autocompletado de campos, por mencionar algunos. Agregando nuevos PDIUs se incrementará exponencialmente el uso y lo beneficios de utilizar la herramienta ImagIng, es decir con un repositorio amplio de PDIU será posible generar automáticamente el código fuente de las interfaces de usuario de prácticamente cualquier aplicación tanto para Web como para dispositivos móviles.

Asimismo, se pretende trabajar en la composición de los PDIUs y mejorar el reconocimiento de los elementos. En otras palabras, se trata de mejorar la capacidad de ImagIng para identificar interfaces complejas conformadas por dos o más PDIUs en una misma imagen.

En cuanto a las plataformas de software soportadas, se espera ampliar las opciones de ImagIng a BlackBerry o Ubuntu Touch, por mencionar algunas. A pesar de que fueron incluidas las plataformas de software o sistemas operativos móviles con la mayor cuota de mercado, en el futuro se espera incluir tantos de estos como sea posible y no dejar a ninguno relegado.

Las opciones para continuar con el trabajo de la herramienta ImagIng son muy variadas ya que existen diversas áreas de oportunidad para esta herramienta, por lo cual se busca continuar el desarrollo y mejorar la herramienta continuamente para evitar así que quede obsoleta en el futuro.

Productos académicos

A continuación se presentan los trabajos derivados de este proyecto de investigación.

Libro

Alor-Hernandez, G., **Rosales-Morales, V. Y.,** & Colombo-Mendoza, L. O. (2015). Frameworks, Methodologies, and Tools for Developing Rich Internet Applications (pp. 1-366). Hershey, PA: IGI Global. doi:10.4018/978-1-4666-6437-1.

Artículos JCR

Viviana Yarel Rosales-Morales, Giner Alor-Hernández, Jorge Luis García-Alcaráz, Ramón Zatarain-Cabada, María Lucía Barrón-Estrada. An analysis of tools for automatic software development and automatic code generation. Revista Facultad de Ingeniería. No. 77. Pp 75-87. ISSN: 0120-6230. 2013 ISI Impact **Factor: 0.070.** DOI: http://dx.doi.org/10.17533/udea.redin.n77a10

Viviana Yarel Rosales-Morales, Laura Nely Sánchez-Morales, Giner Alor-Hernández, Jorge Luis García-Alcaraz, José Luis Sánchez-Cervantes, Lisbeth Rodriguez-Mazahua. ImagIngDev: A New Approach for Developing Automatic Cross-Platform Mobile Applications Using Image Processing Techniques. **Estatus: Enviado**

Capítulo de libro

Sánchez-Morales, Laura Nely; Alor-Hernández, Giner; Miranda-Luna, Rosebet; **Rosales-Morales, Viviana Yarel**; Cortes-Camarillo, Cesar Augusto; Generation of User Interfaces for mobile applications using neuronal networks. Management and Industrial Engineering. Springer Verlag. ISSN: 2365-0532.

Conferencias Internacionales

Laura Sánchez Morales, **Viviana Yarel Rosales-Morales**, Giner Alor-Hernández, Rubén Posada-Gómez, Hilarión Muñoz Contreras, Ulises Juárez-Martínez. Módulo de generación de aplicaciones multi-dispositivo a partir del procesamiento de imágenes. Research in Computing Science 91: 81-94. ISSN: 1870-4069. **Indexing: LATINDEX, PERIODICA, DBLP.**

Cesar Augusto Cortes-Camarillo, **Viviana Yarel Rosales-Morales**, Laura Nely Sanchez-Morales, Giner Alor-Hernández, Lisbeth Rodríguez-Mazahua. Atila: A UIDPs-based educational application generator for mobile devices. 27th International Conference on Electronics, Communications and Computers. IEEE Press.

Registro de Derecho de Autor

Alor-Hernández, G., **Rosales-Morales V. Y.** & Sánchez-Ramírez C. (2015). ImagIng: Un generador de aplicaciones Web usando técnicas de procesamiento de imágenes. Estatus: Registrado. INDAUTOR. **Número de Registro: 03-2015-052013184200-01**

Proyecto de Investigación

Clave: 5599.15-P. Desarrollo de una herramienta para la generación de software multi-dispositivo a partir del reconocimiento de patrones en imágenes. Convocatoria 2015 de Proyectos de Investigación Científica, Aplicada, Desarrollo Tecnológico e Innovación. TecNM. Estatus: Financiado

Desarrollo de una herramienta para la generación de aplicaciones educativas multi-plataforma y multi-dispositivo usando patrones de diseño de interfaces de usuario y técnicas de inteligencia artificial. Convocatoria de Apoyo a Proyectos de Investigación Científica, Aplicada, Desarrollo Tecnológico e Innovación 2017. TecNM. Estatus: Aprobado.

Referencias

Abdullah, S., Rahman, A., & Abas, Z. (2016). Multilayer Perceptron Neural Network in Classifying Gender using Fingerprint Global Level Features. *Journal of Science and …*, *9*(March). http://doi.org/10.17485/ijst/2016/v9i9/84889

Acerbis, R., Bongio, A., Brambilla, M., & Butti, S. (2015). Model-Driven Development of Cross-Platform Mobile Applications with Web Ratio and IFML. *Mobile Software Engineering and Systems (MOBILESoft), 2015 2nd ACM International Conference on*. http://doi.org/10.1109/MobileSoft.2015.49

Alor-Hernández, G., Rosales-Morales, V. Y., & Colombo-Mendoza, L. O. (2014). *Frameworks, Methodologies, and Tools for Developing Rich Internet Applications* (1st ed.). Hershey, PA, USA: IGI Global.

Amalfitano, D., Fasolino, A. R., & Tramontana, P. (2011). A GUI Crawling-Based Technique for Android Mobile Application Testing. *Software Testing, Verification and Validation Workshops (ICSTW), 2011 IEEE Fourth International Conference on*. http://doi.org/10.1109/ICSTW.2011.77

Amalfitano, D., Fasolino, A. R., Tramontana, P., De Carmine, S., & Memon, A. M. (2012). Using GUI Ripping for Automated Testing of Android Applications. In *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering* (pp. 258–261). New York, NY, USA: ACM. http://doi.org/10.1145/2351676.2351717

Amalfitano, D., Fasolino, A. R., Tramontana, P., Ta, B. D., & Memon, A. M. (2015). MobiGUITAR: Automated Model-Based Testing of Mobile Apps. *IEEE Software*. http://doi.org/10.1109/MS.2014.55

Balagtas-Fernandez, F. T., & Hussmann, H. (2008). Model-Driven Development of Mobile Applications. In *Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering* (pp. 509–512). Washington, DC, USA: IEEE Computer Society. http://doi.org/10.1109/ASE.2008.94

Bauer, F. L. (Ed.). (1975). Software Engineering, An Advanced Course, Reprint of the First Edition [February 21 - March 3, 1972]. London, UK, UK: Springer-Verlag.

Beck, K. (2003). Test-Driven Development By Example. *Rivers*, *2*(c), 176. http://doi.org/10.5381/jot.2003.2.2.r1

Bell, B. R. (1998). Code generation from object models. *Embedded Systems Programming*. 11 (3) 74 – 88.

Benouda, H., Essbai, R., Azizi, M., & Moussaoui, M. (2016). Modeling and code generation of android applications using acceleo. *International Journal of Software Engineering and Its Applications*, *10*(3), 83–94. http://doi.org/10.14257/ijseia.2016.10.3.08

Boehm, B. W. (1976). Software Engineering. *IEEE Trans. Comput.*, *25*(12), 1226–1241. http://doi.org/10.1109/TC.1976.1674590

Boley, H., Paschke, A., & Shafiq, O. (2010). RuleML 1.0: The Overarching Specification of Web Rules. In M. Dean, J. Hall, A. Rotolo, & S. Tabet (Eds.), *Semantic Web Rules: International Symposium, RuleML 2010, Washington, DC, USA, October 21-23, 2010. Proceedings* (pp. 162–178). Berlin, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/978-3-642-16289-3_15

Brooke, J. (1996). SUS-A quick and dirty usability scale. *Usability evaluation in industry*, *189*(194), 4-7.

Cassani, V., Gianelli, S., Matera, M., Medana, R., Quintarelli, E., Tanca, L., & Zaccaria, V. (2017). On the Role of Context in the Design of Mobile Mashups BT - Rapid Mashup Development Tools: Second International Rapid Mashup Challenge, RMC 2016, Lugano, Switzerland, June 6, 2016, Revised Selected Papers. In F. Daniel & M. Gaedke (Eds.), (pp. 108–128). Cham: Springer International Publishing. http://doi.org/10.1007/978-3-319-53174-8_7

Charland, A., & Leroux, B. (2011). Mobile Application Development: Web vs. Native. *Commun. ACM*, *54*(5), 49–53. http://doi.org/10.1145/1941487.1941504

Choi, Y., Yang, J. S., & Jeong, J. (2009). Application framework for multi platform mobile application software development. *Advanced Communication Technology, 2009. ICACT 2009. 11th International Conference on*.

Cimitile, M., Risi, M., & Tortora, G. (2011). Automatic Generation of Multi Platform Web Map Mobile Applications. In *DMS* (pp. 84–89). Citeseer.

Cireşan, D. C., Meier, U., Gambardella, L. M., & Schmidhuber, J. (2012). Deep Big Multilayer Perceptrons for Digit Recognition. In G. Montavon, G. B. Orr, & K.-R. Müller (Eds.), *Neural Networks: Tricks of the Trade: Second Edition* (pp. 581–598). Berlin, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/978-3-642-35289-8_31

Coad, P., Lefebvre, E., & De Luca, J. (1999). *Java Modeling in Color with UML: Enterprise Components and Process*. Prentice Hall PTR. Retrieved from https://books.google.com.mx/books?id=fo0_AQAAIAAJ

Colombo-Mendoza, L. O., Alor-Hernández, G., Rodríguez-González, A., & Colomo-Palacios, R. (2013). Alexandria: A Visual Tool for Generating Multi-device Rich Internet Applications. *J. Web Eng.*, *12*(3–4), 317–359. Retrieved from http://dl.acm.org/citation.cfm?id=2535629.2535637

Colombo-Mendoza, L. O., Alor-Hernández, G., Rodríguez-González, A., & Valencia-García, R. (2014). MobiCloUP!: a PaaS for cloud services-based mobile applications. *Automated Software Engineering*, *21*(3), 391-437.

Cuevas, E., Zaldivar, D., & Rojas, R. (2003). Computer Vision using MATLAB and the Toolbox of Image Processing Technical Report B-05-09, (December 2003), 1–27.

Dalmasso, I., Datta, S. K., Bonnet, C., & Nikaein, N. (2013). Survey, comparison and evaluation of cross platform mobile application development tools. *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*. http://doi.org/10.1109/IWCMC.2013.6583580

Dash, Y., Kumar, S., & Patle, V. K. (2016). A Novel Data Mining Scheme for Smartphone Activity Recognition by Accelerometer Sensor. In S. Das, T. Pal, S. Kar, C. S. Satapathy, & K. J. Mandal (Eds.), *Proceedings of the 4th International Conference on Frontiers in Intelligent Computing: Theory and Applications (FICTA) 2015* (pp. 131–140). New Delhi: Springer India. http://doi.org/10.1007/978-81-322-2695-6_12

De la Rosa, R. (2007). Procesamiento de Imágenes Digitales." In X Congreso Nacional en Informática y Computación del Instituto Tecnológico de Puebla.

Díez, R. P., Gómez, A. G., & de Abajo Martínez, N. (2001). *Introducción a la inteligencia artificial: sistemas expertos, redes neuronales artificiales y computación evolutiva*.

Servicio de Publicaciones, Universidad de Oviedo. Retrieved from https://books.google.com.mx/books?id=RKqLMCw3IUkC

Dinh, H. T., Lee, C., Niyato, D., & Wang, P. (2013). A survey of mobile cloud computing: Architecture, applications, and approaches. *Wireless Communications and Mobile Computing*, *13*(18), 1587–1611. http://doi.org/10.1002/wcm.1203

Dunkel, J., & Bruns, R. (2007). Model-driven Architecture for Mobile Applications. In *Proceedings of the 10th International Conference on Business Information Systems* (pp. 464–477). Berlin, Heidelberg: Springer-Verlag. Retrieved from http://dl.acm.org/citation.cfm?id=1759779.1759822

El-Henawy, I., & Ahmed, K. (2016). Accelerating convergence of backpropagation for multilayer perceptron neural networks: a case study on character bit-mapped pixel image to ASCII conversion. *International Journal of Computers and Applications*, 1–10. http://doi.org/10.1080/1206212X.2016.1188560

Enard, Q., Stoicescu, M., Balland, E., Consel, C., Duchien, L., Fabre, J.-C., & Roy, M. (2013). Design-driven Development Methodology for Resilient Computing. In *Proceedings of the 16th International ACM Sigsoft Symposium on Component-based Software Engineering* (pp. 59–64). New York, NY, USA: ACM. http://doi.org/10.1145/2465449.2465458

Estrada Cota, I., Carreño León, M. A., Sandoval Bringas, J. A., Leyva Carrillo, A. A., & Hernández Cosío, J. (2016). Gestor de solicitudes de mantenimiento utilizando un sistema web desarrollado a través de la aplicación de la metodología ágil FDD. *Pistas Educativas; Vol. 36, Núm. 114 (2015): CITEC 2015*. Retrieved from http://www.itcelaya.edu.mx/ojs/index.php/pistas/article/view/293

Evans, E. (2004). *Domain-driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley. Retrieved from https://books.google.com.mx/books?id=xColAAPGubgC

Firaus, A., Ghani, I., & Yasin, N. I. M. (2013). Developing Secure Websites Using Feature Driven Development (FDD): A Case Study. *Journal of Clean Energy Technologies*, *1*(4), 322–326. http://doi.org/10.7763/JOCET.2013.V1.73

Garcia-Salgado, B. P., Ponomaryov, V. I., & Robles-Gonzalez, M. A. (2016). Parallel multilayer perceptron neural network used for hyperspectral image classification (Vol. 9897, p. 98970K–98970K–13). Retrieved from http://dx.doi.org/10.1117/12.2227329

Gavalas, D., & Economou, D. (2011). Development Platforms for Mobile Applications: Status and Trends. *IEEE Software*. http://doi.org/10.1109/MS.2010.155

Gonzalez, R. C., & Woods, R. E. (2006). *Digital Image Processing (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Guide, M. U. S. (1998). The mathworks. Natick, MA, 5, 333.

Gutiérrez, A. I. B., Díaz, S. M., & Torres, J. L. G. (2014). *Visión estereoscópica por computadora con Matlab y OpenCV*. Lulu.com. Retrieved from https://books.google.com.mx/books?id=_yK_CQAAQBAJ

Heitkötter, H., Majchrzak, T. A., & Kuchen, H. (2013). Cross-platform Model-driven Development of Mobile Applications with Md2. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing* (pp. 526–533). New York, NY, USA: ACM. http://doi.org/10.1145/2480362.2480464

Heitkötter, H., Hanschke, S., & Majchrzak, T. A. (2013). Evaluating Cross-Platform Development Approaches for Mobile Applications. In J. Cordeiro & K.-H. Krempels (Eds.), *Web Information Systems and Technologies: 8th International Conference, WEBIST 2012, Porto, Portugal, April 18-21, 2012, Revised Selected Papers* (pp. 120–138). Berlin, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/978-3-642-36608-6_8

Heitkötter, H., Hanschke, S., & Majchrzak, T. a. (2013). Comparing Cross-Platform Development Approaches for Mobile Applications. *Web Information Systems and Technologies*, *140*, 120–138. http://doi.org/10.1007/978-3-642-36608-6

Herrington, J. (2003). Code Generation in Action. *Greenwich: Manning Publications*.

Hu, C., & Neamtiu, I. (2011). Automating GUI Testing for Android Applications. In *Proceedings of the 6th International Workshop on Automation of Software Test* (pp. 77–83). New York, NY, USA: ACM. http://doi.org/10.1145/1982595.1982612

IEEE Standard Glossary of Software Engineering Terminology. (1990). *IEEE Std 610.12-1990*. http://doi.org/10.1109/IEEESTD.1990.101064

International Standard Organization: ISO 9241-11:1998 (1998). Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs)—*Part 11: Guidance on Usability*. ISO, Geneva.

James, B., & Lalonde, L. (2015). Domain-Driven Design. In *Pro XAML with C#: Application Development Strategies* (pp. 27–36). Berkeley, CA: Apress. http://doi.org/10.1007/978-1-4302-6775-1_3

Kabáč, M., & Consel, C. (2015). Orchestrating Masses of Sensors: A Design-driven Development Approach. In *Proceedings of the 2015 ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences* (pp. 117–120). New York, NY, USA: ACM. http://doi.org/10.1145/2814204.2814226

Kalay, Y. E. (1999). Performance-based design. *Automation in Construction*, *8*(4), 395–409. http://doi.org/http://dx.doi.org/10.1016/S0926-5805(98)00086-7

Kent Beck (2012). "Why does Kent Beck refer to the "rediscovery" of test-driven development?". Retrieved December 1, 2016.

Kernighan, B. W., & Plauger, P. J. (1981). Software tools in Pascal. Addison-Wesley Longman Publishing Co., Inc.

Kim, H., Choi, B., & Yoon, S. (2009). Performance Testing Based on Test-driven Development for Mobile Applications. In *Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication* (pp. 612–617). New York, NY, USA: ACM. http://doi.org/10.1145/1516241.1516349

Kitchenham, B., Linkman, S., & Law, D. (1997). DESMET: a methodology for evaluating software engineering methods and tools. *Computing Control Engineering Journal*, *8*(3), 120 –126. doi:10.1049/ccej:19970304

Kupiainen, E., Mäntylä, M. V, & Itkonen, J. (2015). Using metrics in Agile and Lean Software Development – A systematic literature review of industrial studies. *Information and Software Technology*, *62*, 143–163. http://doi.org/http://dx.doi.org/10.1016/j.infsof.2015.02.005

Likert, R. (1932). A technique for the measurement of attitudes, *Archives of psychology*, vol. 22, no. 140, pp. 5-55.

López, R. F., & Fernández, J. M. F. (2008). *Las Redes Neuronales Artificiales*. Netbiblo. Retrieved from https://books.google.com.mx/books?id=X0uLwi1Ap4QC

Mao, Z. M. (2016). Diagnosing Mobile Apps' Quality of Experience: Challenges and Promising Directions. *IEEE Internet Computing*. http://doi.org/10.1109/MIC.2016.1

Mathworks. (2016). Image Processing Toolbox. Retrieved from: http://www.mathworks.de/products/image/

Miravet, P., Marín, I., Ortín, F., & Rionda, A. (2009). DIMAG: A Framework for Automatic Generation of Mobile Applications for Multiple Platforms. In *Proceedings of the 6th International Conference on Mobile Technology, Application & Systems* (p. 23:1--23:8). New York, NY, USA: ACM. http://doi.org/10.1145/1710035.1710058

Neil, T. (2014). *Mobile Design Pattern Gallery: UI Patterns for Smartphone Apps*. O'Reilly Media, Inc.

Núñez-Valdez, E. R., García-Díaz, V., Lovelle, J. M. C., Achaerandio, Y. S., & González-Crespo, R. (2016). A model-driven approach to generate and deploy videogames on multiple platforms. *Journal of Ambient Intelligence and Humanized Computing*, 1–13. http://doi.org/10.1007/s12652-016-0404-1

Palmer, S. R., & Felsing, J. M. (2002). *A Practical Guide to Feature-driven Development*. Prentice Hall PTR. Retrieved from https://books.google.com.mx/books?id=NhlFAAAAYAAJ

Possatto, M. A., & Lucrédio, D. (2015). Automatically propagating changes from reference implementations to code generation templates. *Information and Software Technology*, *67*, 65–78. http://doi.org/http://dx.doi.org/10.1016/j.infsof.2015.06.009

Qi, Y.-L., Zhang, G.-S., & Li, Y.-L. (2016). A New Method to Generate Semantic Templates Based on Multilayer Perceptron. In A. Hussain (Ed.), *Electronics, Communications and Networks V: Proceedings of the 5th International Conference on Electronics, Communications and Networks (CECNet 2015)* (pp. 35–41). Singapore: Springer Singapore. http://doi.org/10.1007/978-981-10-0740-8_5

Raona.com. (2017). ¿App nativa, web o híbrida?. [online] Available at: http://www.raona.com/es/Solutions/Template/163/App-nativa-web-o-h%C3%ADbrida- [Accessed 3 Jan. 2017].

Rincón Nigro, M., Hidrobo Torres, F., & Aguilar Castro, J. (2010). Generación automática de código a partir de máquinas de estado finito. *Computación Y Sistemas*, *14*(4), 405–421.

Rocha Silva, T., Hak, J.-L., & Winckler, M. (2016). An Approach for Multi-Artifact Testing Through an Ontological Perspective for Behavior-Driven Development. *Complex Systems Informatics and Modeling Quarterly*, (7), 81–107. http://doi.org/10.7250/csimq.2016-7.05

Rocha Silva, T., Hak, J.-L., & Winckler, M. (2016). Testing Prototypes and Final User Interfaces Through an Ontological Perspective for Behavior-Driven Development. In C. Bogdan, J. Gulliksen, S. Sauer, P. Forbrig, M. Winckler, C. Johnson, … F. Kis (Eds.), *Human-Centered and Error-Resilient Systems Development: IFIP WG 13.2/13.5 Joint Working Conference, 6th International Conference on Human-Centered Software Engineering, HCSE 2016, and 8th International Conference on Human Error, Safety, and System Develop* (pp. 86–107). Cham: Springer International Publishing. http://doi.org/10.1007/978-3-319-44902-9_7

Satyavathy, G., Priya, P. S., & Chanthini, S. (2017). Mobile Application Development. Automation and Autonomous System, 9(1), 6-10.

Schmidt, D. C. (2006). Model-Driven Engineering. *IEEE Computer*, *39*(2), 25–31. http://doi.org/10.1109/MC.2006.58

Sheibley, M. (2013). *Mobile Patterns*. Online. Available: http://www.mobile-patterns.com. Accessed 10 Septiembre 2016.

Soares, S. A., Brandão, M., Cortés, M. I., & Freire, E. S. S. (2015). Dribbling complexity in model driven development using Naked Objects, domain driven design, and software design patterns. *Computing Conference (CLEI), 2015 Latin American*. http://doi.org/10.1109/CLEI.2015.7360022

Solis, C., & Wang, X. (2011). A Study of the Characteristics of Behaviour Driven Development. *2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications*. http://doi.org/10.1109/SEAA.2011.76

Sommerville, I., & Sawyer, P. (1997). Requirements engineering: a good practice guide. John Wiley & Sons, Inc.

Sommerville, I. (2004). Software Engineering. International computer science series.

Srihari, S. N., & Franke, K. (2008). *Computational Forensics: Second International Workshop, IWCF 2008, Washington, DC, USA, August 7-8, 2008, Proceedings* (Vol. 5158). Springer.

Tidwell, J. (2010). *Designing Interfaces: Patterns for Effective Interaction Design*. O'Reilly Media. Retrieved from https://books.google.com.mx/books?id=5gvOU9X0fu0C

Toxboe, A. (2016). *UI Patterns User Interface Design pattern Library.* Online. Available: http://ui-patterns.com. Accessed 10 Septiembre 2016.

Truyen, F. (2006). The Fast Guide to Model Driven Architecture The Basics of Model Driven Architecture. *Practice*, 12. Retrieved from http://www.omg.org/mda/mda_files/Cephas_MDA_Fast_Guide.pdf

UI-Patterns. (2016). User Interface Design Pattern Library [Online]. Available: http://ui-patterns.com/patterns. Accessed 10 Septiembre 2016.

Umuhoza, E., Ed-douibi, H., Brambilla, M., Cabot, J., & Bongio, A. (2015). Automatic Code Generation for Cross-platform, Multi-device Mobile Apps: Some Reflections from an Industrial Experience. *Proceedings of the 3rd International Workshop on Mobile Development Lifecycle*, (OCTOBER), 37–44. http://doi.org/10.1145/2846661.2846666

UNITiD. (2016). *Android Patterns*. Online. Available: http://unitid.nl/androidpatterns. Accessed 10 Septiembre 2016.

Vallon, R., Wenzel, L., E. Brüggemann, M., & Grechenig, T. (2015). An Agile and Lean Process Model for Mobile App Development: Case Study into Austrian Industry. *Journal of Software*, *10*(11), 1245–1264. http://doi.org/10.17706/jsw.10.11.1245-1264

Vaquero-Melchor, D., Garmendia, A., Guerra, E., & Lara, J. De. (2017). Domain-Specific Modelling using Mobile Devices.

Vaupel, S., Taentzer, G., Gerlach, R., & Guckert, M. (2016). Model-driven development of mobile applications for Android and iOS supporting role-based app variability. *Software & Systems Modeling*, 1–29. http://doi.org/10.1007/s10270-016-0559-4

Völter, M., Stahl, T., Bettin, J., Haase, A., & Helsen, S. (2013). *Model-driven software development: technology, engineering, management*. John Wiley & Sons.

Wirfs-Brock, R., & Wilkerson, B. (1989). Object-oriented Design: A Responsibility-driven Approach. *SIGPLAN Not.*, *24*(10), 71–75. http://doi.org/10.1145/74878.74885

Wirfs-Brock, R., & McKean, A. (2003). *Object Design: Roles, Responsibilities, and Collaborations*. Addison-Wesley. Retrieved from https://books.google.com.mx/books?id=vUF72vN5MY8C

Yang, W., Prasad, M. R., & Xie, T. (2013). A Grey-Box Approach for Automated GUI-Model Generation of Mobile Applications. In V. Cortellessa & D. Varró (Eds.), *Fundamental Approaches to Software Engineering: 16th International Conference, FASE 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings* (pp. 250–265). Berlin, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/978-3-642-37057-1_19

Yasuhiro Monden (1998), Toyota Production System, An Integrated Approach to Just-In-Time, Third edition, Norcross, GA: Engineering & Management Press, ISBN 0-412-83930-X.

Zhu, H., Liu, C., Ge, Y., Xiong, H., & Chen, E. (2015). Popularity Modeling for Mobile Apps: A Sequential Approach. *IEEE Transactions on Cybernetics*. http://doi.org/10.1109/TCYB.2014.2349954