

PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

DISERTACIÓN PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS Y COMPUTACIÓN

"INTEGRACIÓN ENTRE LAS TECNOLOGÍAS RASPBERRY PI Y FIREBASE MEDIANTE EL DESARROLLO DE UNA APLICACIÓN MÓVIL UTILIZANDO REACT NATIVE QUE CONTROLE LA PRESENTACIÓN DE IMÁGENES EN UN MONITOR EXTERNO"

AUTOR:

ESTEBAN PATRICIO ARISTIZÁBAL SCACCO

DIRECTOR:

MTR. ANDRÉS JIMÉNEZ PACHECO

QUITO, 2018

DEDICATORIA

A mi padre Patricio, que siempre será mi inspiración y motivación en los sueños que quiero cumplir en mi vida.

A mi madre Dina, por estar conmigo cada día, preocuparse de mis problemas y apoyarme incondicionalmente.

A mi hermano Daniel, por los objetivos que nos falta cumplir y las metas que nos hemos trazado.

A mi novia Andrea, por ser mi mejor amiga, mi tranquilidad en los momentos difíciles y estar junto a mí siempre.

AGRADECIMIENTOS

A mis padres Patricio y Dina, por su esfuerzo y dedicación para que pueda tener una educación de calidad.

A mis profesores, que con su conocimiento y entrega me prepararon inmejorablemente para los retos que me faltan por afrontar en mi vida profesional.

A mis compañeros de curso, por haber recorrido juntos esta etapa de aprendizaje formando sinceros lazos de amistad y apoyo.

RESUMEN

Con el dispositivo Raspberry Pi emergiendo como una alternativa para desarrollar soluciones en el campo tecnológico, en el presente trabajo se demuestra su integración con la plataforma Firebase mediante el desarrollo de una aplicación móvil que controle la presentación de imágenes en un monitor externo. Todo esto con el objetivo de comprobar que un Raspberry Pi puede ser controlado remotamente utilizando la sincronización en tiempo real de la información alojada en Firebase con los dispositivos conectados a ella, y que en un futuro el trabajo realizado sirva como una fuente de consulta y abra las puertas al desarrollo de soluciones tecnológicas que sean comercialmente exitosas e innovadoras bajo la arquitectura del prototipo construido que involucren el uso del Raspberry Pi.

Igualmente se documenta de forma precisa el análisis de requerimientos, metodología, diseño, implementación y pruebas del prototipo desarrollado para verificar la integración entre el Raspberry Pi, Firebase y la aplicación móvil construida con React Native para el sistema operativo Android.

TABLA DE CONTENIDO

DEDICATORIA ..... **i**

AGRADECIMIENTOS ..... **ii**

RESUMEN ..... **iii**

ÍNDICE DE FIGURAS ..... **vii**

ÍNDICE DE TABLAS ..... **x**

CAPÍTULO 1. MARCO TEÓRICO **12**

1.1. Computador de placa única 12

1.2. Raspberry Pi 13

1.3. Firebase 18

1.4. React Native 21

1.5. Cuadro de tecnologías 25

CAPÍTULO 2. ANÁLISIS Y DISEÑO **27**

2.1. Análisis de requerimientos 27

2.2. Metodología 29

2.3. Diseño 39

2.3.1. Arquitectura 39

2.3.2. Estructura de datos 41

2.3.3. Interfaz de usuario 45

CAPÍTULO 3. DESARROLLO **48**

3.1. Herramientas de desarrollo 48

3.1.1. Raspberry Pi 3 Model B 48

3.1.2. Firebase Realtime Database 50

3.1.3. Node.js 50

3.1.4. React Native 51

3.1.5. Nativebase 52

3.2. Seguimiento de la iteración 53

3.3. Pruebas de la aplicación 67

CAPÍTULO 4. CONCLUSIONES Y RECOMENDACIONES 73

4.1. Conclusiones 73

4.2. Recomendaciones 75

GLOSARIO 77

BIBLIOGRAFÍA 80

ÍNDICE DE FIGURAS

Figura 1-1. SBC Raspberry Pi 3 Model B+. (Raspberry Pi Foundation, 2018)..... 12

Figura 1-2. Áreas de aplicación de los productos de Firebase (Firebase, 2018)..... 18

Figura 1-3. Compañías que utilizan React Native (Facebook, 2018) 21

Figura 2-1. Ciclo de vida del proceso Extreme Programming (Pressman & Maxim, 2015) 33

Figura 2-2. Ejemplo de un tablero organizado en columnas de estado por hacer, en progreso y completado (Stellman & Greene, 2015)..... 37

Figura 2-3. Tablero de tareas del proyecto (Aristizábal S., 2018) 38

Figura 2-4. Arquitectura del prototipo (Aristizábal S., 2018)..... 40

Figura 2-5. Diagrama Entidad Relación Base de datos NoSql (Aristizábal S., 2018)42

Figura 2-6. Objeto JSON Usuario (Aristizábal S., 2018) 44

Figura 2-7. Objeto JSON Monitor (Aristizábal S., 2018)..... 44

Figura 2-8. Objeto JSON Imagen (Aristizábal S., 2018) 44

Figura 2-9. Diseño de la interfaz de usuario para la aplicación móvil (Aristizábal S., 2018) 46

Figura 2-10. Diseño de la interfaz de usuario para el monitor externo (Aristizábal S., 2018) 47

Figura 3-1. Raspberry Pi 3 Model B (Raspberry Pi Foundation, 2018) 48

Figura 3-2. Componentes gráficos prefabricados de la librería Nativebase (Nativebase, 2018) 53

Figura 3-3. Estado del tablero de tareas del proyecto durante el proceso de desarrollo (Aristizábal S., 2018) 54

Figura 3-4. Consola del proyecto en Firebase (Aristizábal S., 2018) 55

Figura 3-5. Estructura de la base de datos NoSql en Firebase (Aristizábal S., 2018) 56

Figura 3-6. Pantalla de la aplicación de inicio del framework React Native (Aristizábal S., 2018) 57

Figura 3-7. Pantalla con el diseño de la interfaz gráfica de la aplicación terminado (Aristizábal S., 2018) 57

Figura 3-8. Credenciales del proyecto en Firebase (Aristizábal S., 2018)..... 58

Figura 3-9. Código que consume imágenes desde Firebase (Aristizábal S., 2018) ... 59

Figura 3-10. Pantalla de la aplicación mostrando un conjunto de imágenes recuperadas desde Firebase (Aristizábal S., 2018)..... 60

Figura 3-11. Código que despliega la imagen en un monitor externo conectado a un Raspberry Pi (Aristizábal S., 2018)..... 61

Figura 3-11. Comando para modificar los programas de inicio en un Raspberry Pi (Aristizábal S., 2018) 61

Figura 3-13. Valores modificados en el archivo “autostart” (Aristizábal S., 2018) .. 61

Figura 3-11. Código que inicia el servidor web local de Node.js (Aristizábal S., 2018) 62

Figura 3-11. Código que integra Firebase en la aplicación web del Raspberry Pi (Aristizábal S., 2018) 63

Figura 3-16. Código que recupera la imagen que se visualiza en el monitor externo desde Firebase (Aristizábal S., 2018)..... 64

Figura 3-17. Registro imagenActual en Firebase (Aristizábal S., 2018) 65

Figura 3-18. Código que actualiza el registro imagenActual del monitor en Firebase (Aristizábal S., 2018) 65

Figura 3-19. Registro imagenActual actualizándose en tiempo real en Firebase (Aristizábal S., 2018) 65

Figura 3-20. Código que recupera y actualiza en tiempo real la imagen que se despliega en el monitor (Aristizábal S., 2018)..... 66

Figura 3-21. Imagen de prueba visualizada en el monitor (Aristizábal S., 2018)..... 66

Figura 3-22. Resultado 1 prueba de aceptación H1 (Aristizábal S., 2018)..... 68

Figura 3-23. Resultado 2 prueba de aceptación H1 (Aristizábal S., 2018)..... 68

Figura 3-24. Resultado 1 prueba de aceptación H2 (Aristizábal S., 2018)..... 69

Figura 3-25. Resultado 2 prueba de aceptación H2 (Aristizábal S., 2018)..... 70

Figura 3-26. Resultado 1 prueba de aceptación H3 (Aristizábal S., 2018)..... 71

Figura 3-27. Resultado 2 prueba de aceptación H3 (Aristizábal S., 2018)..... 72

Figura 3-28. Resultado 3 prueba de aceptación H3 (Aristizábal S., 2018)..... 72

ÍNDICE DE TABLAS

Tabla 1-1. Diferencias entre SBCs y computadoras tradicionales (Aristizábal S., 2018) 13

Tabla 1-2. Modelos Raspberry Pi en el mercado actual (Raspberry Pi Foundation, 2018) 15

Tabla 1-3. Características de hardware modelos Raspberry Pi en el mercado actual (Raspberry Pi Foundation, 2018) 16

Tabla 1-4. Principales sistemas operativos alternos que pueden utilizarse en un Raspberry Pi (Raspberry Pi Foundation, 2018) 18

Tabla 1-5. Cuadro de tecnologías del proyecto (Aristizábal S., 2018) 26

Tabla 1-4. Cuadro comparativo metodologías ágiles (Raspberry Pi Foundation, 2018) 31

Tabla 2-1. Historias de usuario (Aristizábal S., 2018) 35

Tabla 2-2. Historias de usuario divididas en tareas (Aristizábal S., 2018) 36

Tabla 2-3. Pruebas de aceptación para las historias de usuario (Aristizábal S., 2018) 39

Tabla 2-4. Entidad Usuario (Aristizábal S., 2018) 42

Tabla 2-5. Entidad Monitor (Aristizábal S., 2018) 42

Tabla 2-6. Entidad Imagen (Aristizábal S., 2018) 43

Tabla 3-1. Características del Raspberry Pi Model 3 (Raspberry Pi Foundation, 2018) 49

CAPÍTULO 1. MARCO TEÓRICO

En este capítulo se describirán los conceptos de las tecnologías involucradas en el desarrollo del presente trabajo con el fin de dar un fundamento teórico al mismo.

1.1. Computador de placa única

En inglés *Single Board Computer* (SBC), es una computadora completamente funcional construida en una sola placa de circuito impreso, cuenta con microprocesador, memoria RAM[1], almacenamiento, periféricos de E/S[2] y otras características requeridas para ser una computadora mínimamente funcional como se muestra en la ***figura 1-1.***



Figura 1-1. SBC Raspberry Pi 3 Model B+. (Raspberry Pi Foundation, 2018)

La ***tabla 1-1*** lista las diferencias entre las SBCs y computadoras tradicionales.

SBCs	Computadoras tradicionales
Coste económico bajo	Cuestan más que una SBC
Factor de forma reducido	Factor de forma mayor
Placa base en un solo circuito	Placa base en 2 o más circuitos
Carecen de pantalla y teclado	Poseen en su mayoría pantalla y teclado

[1] RAM: Memoria de acceso aleatorio
[2] E/S: Entrada/Salida

Consumen menos energía	Consumen más energía
Componentes no pueden ser reemplazados o mejorados	Componentes pueden ser reemplazados o mejorados
Capacidad de procesamiento bajo	Mayor capacidad de procesamiento
Sistemas de enfriamiento simples	En su mayoría poseen sistemas de enfriamiento complejos
Su uso es experimental o comercial	Su uso es profesional o recreativo
Se utilizan en sistemas embebidos	No se pueden utilizar en sistemas embebidos
Recolección y envío de datos mediante sensores	Recepción y procesamiento de datos

Tabla 1-1. Diferencias entre SBCs y computadoras tradicionales (Aristizábal S., 2018)

1.2. Raspberry Pi

Es una familia de SBCs[3] desarrolladas en el Reino Unido por la Raspberry Foundation. El objetivo detrás del desarrollo del Raspberry Pi fue promover la enseñanza de computación básica en escuelas y países en desarrollo proporcionando una plataforma de computación a un bajo coste económico (Pajankar, 2017).

Desde su lanzamiento en el año 2012 hasta marzo del 2018 se han vendido 19 millones de unidades (Upton, 2018) siendo muy popular debido a que su uso ha sido expandido más allá de la enseñanza de programación en las escuelas, llegando a ser utilizada en el desarrollo de proyectos por parte de entusiastas de la computación e ingenieros, convirtiéndose actualmente en uno de las herramientas de computación más utilizadas en el mundo.

Se pueden distinguir 3 usos principales que se le pueden dar a un Raspberry Pi:

1. *Computación de uso general:* Debido a que no deja de ser una computadora mínimamente funcional puede ser utilizada para

[3] SBCs: Computadoras de placa única

entretenimiento por ejemplo ver películas, jugar videojuegos o escuchar música, como también para tareas de productividad como trabajar con documentos, hojas de cálculo, software profesional o almacenamiento de información.

2. *Plataforma de enseñanza:* Siendo este el uso para el que originalmente fue pensada, puede ser implementada en escuelas para la enseñanza de programación básica a niños debido a que viene precargada con lenguajes de programación y software orientado al desarrollo de esta capacidad.
3. *Plataforma de proyectos:* Debido a su flexibilidad de uso y específicamente a su capacidad de integrarse con componentes electrónicos como sensores, su potencial ha sido demostrado en proyectos basados en tecnología tanto comerciales como experimentales.

La ***tabla 1-2*** muestra los modelos de Raspberry Pi que se ofertan en el mercado actual. La ***tabla 1-3*** una comparación de las principales características de estos modelos.

Raspberry Pi 3 Modelo B+	
Raspberry Pi 3 Modelo B	
Raspberry Pi 2 Modelo B	

Raspberry Pi 1 Modelo B+	
Raspberry Pi 1 Modelo A+	
Raspberry Pi Zero W	
Raspberry Zero	

Tabla 1-2. Modelos Raspberry Pi en el mercado actual (Raspberry Pi Foundation, 2018)

Elaborado por: Esteban Aristizábal

	3 B+	**3 B**	**2 B**	**1 B+**	**1 A+**	**Zero W**	**Zero**
Lanzamiento	Mar 2018	Feb 2016	Feb 2015	Jul 2014	Nov 2014	Feb 2017	Nov 2015
Breve descripción	Última revisión de la tercera generación	SBC de tercera generación	SBC de segunda generación	Revisión final del modelo original	Variante de bajo coste del modelo original	SBC con bluetooth y wi-fi	SBC de bajo coste
Precio	$38	$35	$35	$25	$20	10$	$5
SOC[4]	Broadcom BCM2837B0	Broadcom BCM2837	Broadcom BCM2836	Broadcom BCM2835			

[4] SOC: Sistema en chip

CPU[5]	1.4GHz 64-bit Quad-Core ARMv8 Cortex-A53	1.2GHz 64-bit quad-core ARM Cortex-A53	900MHz 32-bit quad-core ARM Cortex-A7	700MHz single core ARM1176JZF-S		1GHz ARM1176JZF-S single core
GPU[6]	VideoCore IV					
RAM[7]	1 Gb[8]			512 Mb[9]		
Puertos USB[10]	4				1	
Ethernet	10/100/1000 (max 300) Mbps[11]	10/100 Mbps				No dispone
Video	HDMI[12]					Mini-HDMI
Audio	Jack 3.5mm, HDMI					Mini-HDMI
SD/MMC	MicroSD					
Wi-Fi	802.11ac dual band 2.4/5 GHz	802.11n	No dispone		802.11n	No dispone
Bluetooth	Bluetooth 4.2	Bluetooth 4.1	No dispone		Bluetooth 4.1	No dispone
Dimensiones	85.60 mm × 56.5 mm		85.60 mm × 56.5 mm	65 mm × 56.5 mm	65 mm × 30 mm	
Consumo	1.13A	1.34A	820mA	350mA	200mA	350mA
Alimentación	5V MicroUSB/GPIO					

Tabla 1-3. Características de hardware modelos Raspberry Pi en el mercado actual (Raspberry Pi Foundation, 2018)

Elaborado por: Esteban Aristizábal

[5] CPU: Unidad de procesamiento central
[6] GPU: Unidad de procesamiento gráfico
[7] RAM: Memoria de acceso aleatorio
[8] Gb: Gigabyte
[9] Mb: Megabyte
[10] USB: Bus universal en Serie
[11] Mbps: Megabits por segundo
[12] HDMI: Interfaz multimedia de alta definición

Como toda computadora los modelos Raspberry Pi deben usar un sistema operativo que gestione el hardware del computador y provea aplicaciones para interactuar con el dispositivo. Raspbian es el sistema operativo oficial para todos los modelos (Raspberry Pi Foundation, 2018), una distribución Linux basada en Debian precargada con programas, paquetes de software precompilado y una interfaz de usuario gráfica para el usuario. Es muy conveniente que sea una distribución basada en Linux porque al ser gratuita permite mantener los costos bajos y de código abierto para que la plataforma sea hackeable, sin embargo no se está limitado a utilizar únicamente Raspbian porque existen varios sistemas operativos capaces de ser utilizados en un Raspberry Pi. La ***tabla 1-4*** muestra los principales junto con una breve descripción de su finalidad.

Sistema Operativo	Descripción
UBUNTU MATE	Basado en Ubuntu, provee un ambiente de escritorio
SNAPPY UBUNTU CORE	Versión ligera de Ubuntu, diseñado para sistemas embebidos y aplicaciones IoT[13]
WINDOWS 10 IOT CORE	Versión de Windows 10 optimizada para dispositivos pequeños y aplicaciones IoT
OSMC	Reproductor de medios digitales basado en Linux
LIBREELEC	Distribución de Linux desarrollada para funcionar con Kodi[14]
PINET	Gestión centralizada de aulas dedicadas a la enseñanza con dispositivos Raspberry Pi
RISC OS	No basado en Linux, diseñado para trabajar con procesadores ARM[15]
WEATHER STATION	Sistema operativo enfocado a utilizar Oracle Weather Station
ICHIGOJAM RPI	Diseñado para aprender a programar con el lenguaje de programación BASIC[16]

[13] IoT: Internet de las cosas
[14] Kodi: Software de reproducción de medios digitales de código abierto
[15] ARM: En inglés Advanced Risc Machine, procesadores basados en arquitectura RISC
[16] BASIC: Lenguaje de programación de alto nivel enfocado en la facilidad de uso

Tabla 1-4. Principales sistemas operativos alternos que pueden utilizarse en un Raspberry Pi (Raspberry Pi Foundation, 2018)

Elaborado por: Esteban Aristizábal

1.3. Firebase

"Es una plataforma BaaS (Backend as a Service)[17] soportada por Google que brinda soluciones backend[18] para el desarrollo de aplicaciones web y móviles" (Mayur Tanna, 2018). Se enfoca en liberar de las dificultades que implica el desarrollo y manejo de las tareas que normalmente debe realizar un servidor como por ejemplo almacenamiento, APIs[19], bases de datos o hosting, y permitir al desarrollador construir software enfocado a las necesidades del cliente.

Actualmente los productos que ofrece la plataforma Firebase se dividen en 3 áreas de aplicación representadas en la ***figura 1-2***.



Figura 1-2. Áreas de aplicación de los productos de Firebase (Firebase, 2018)

Elaborado por: Esteban Aristizábal

[17] BaaS: Modelo de servicio de computación en la nube que brinda tecnologías para el desarrollo de aplicaciones web y móviles
[18] Backend: En Ingeniería de Software es considerada la capa que se aloja en un servidor y se encarga de brindar funcionalidad al programa
[19] APIs: Conjunto de funciones y procedimientos para ser utilizado por otro software

1. **Desarrollo:** Diseñadas para mejorar y facilitar el proceso de desarrollo de una aplicación web o móvil.

- *Cloud Firestore:* Base de datos NoSQL flexible, escalable y en la nube a fin de almacenar y sincronizar datos para la programación en el lado del cliente y del servidor.
- *ML[20] Kit:* SDK[21] para aplicaciones móviles que provee la tecnología de "machine learning" de Google a las plataformas iOS y Android.
- *Cloud Functions:* Permite ejecutar de forma automática el código backend en respuesta a eventos activados por las funciones de Firebase y las solicitudes HTTPS.
- *Firebase Authentication:* Ofrece autenticación de usuarios para una aplicación web y móvil mediante contraseñas, números de teléfono y proveedores de identidad como Google, Facebook, Twitter y muchos más.
- *Firebase Hosting:* Servicio de hosting de contenido web orientado a programadores en una red de distribución de contenido global.
- *Cloud Storage:* Es un servicio de almacenamiento de objetos potente, simple, rentable y escalable.
- *Realtime Database:* Base de Datos NoSQL alojada en la nube que almacena datos y sincroniza en tiempo real con cada cliente conectado.

2. **Mejorar la calidad:** Herramientas dedicadas a la recolección de estadísticas soble fallos, rendimiento y pruebas de la aplicación en diferentes dispositivos alojados en la nube.

- *Crashlytics:* Informa en tiempo real de los fallos que afectan la calidad de una app.
- *Performance Monitoring:* Servicio que obtiene estadísticas sobre el rendimiento de la aplicación en dispositivos iOS y Android.

[20] ML: Machine Learning
[21] SDK: Software Development Kit

- *Test Lab:* Hostea dipositivos iOS y Android en la nube para que puedan utilizarse para pruebas de la aplicación y observar los resultados.

3. **Crecer tu negocio:** Firebase provee un conjunto de herramientas que ayudan a incrementar las ganancias que pueden generar una aplicación y proveer retroalimentación sobre el uso de la app.

- *Analytics:* Proporciona estadísticas sobre el uso de la app y la participación de los usuarios.
- *Firebase Predicitions:* Incorpora el poder del aprendizaje automático de Google a los datos generados por la aplicación para comprender el comportamiento de los usuarios.
- *Pruebas A/B:* Facilita la ejecución, análisis y el escalamiento del producto y los experimentos de marketing. Por ejemplo permite probar cambios en la UI[22] para ver si cambian métricas como la aceptación de nuevos diseños por parte de los usuarios.
- *Cloud Messaging:* Solución de mensajería multiplataforma que permite enviar mensajes de forma segura y gratuita.
- *Remote Config:* Cambiar el comportamiento o apariencia de la app modificando variables en el lado del servidor sin necesidad de actualizar la aplicación.
- *Dynamic Link:* Vincula directamente al contenido deseado, ya sea en la app nativa o en web.
- *App Indexing:* Integra la aplicación en la búsqueda de Google, sugiere el uso de la aplicación a partir de las búsquedas del usuario.
- *Firebase Invites:* Facilita que el contenido compartido e invitaciones entre usuarios sea fácil de implementar.

[22] UI: User Interface

1.4. React Native

"Es un framework Javascript para desarrollar aplicaciones móviles nativas en iOS y Android, está basado en React, la biblioteca de Facebook para construir interfaces de usuario, pero en lugar de apuntar al navegador, apunta a plataformas móviles" (Eisenman, 2018).

Desde que fue anunciado por Facebook en el año 2015 (Occhino, 2015) se ha convertido en una de las plataformas más populares para el desarrollo de aplicaciones multiplataforma móviles, contando actualmente con una comunidad que supera los 500 contribuidores (Alpert, 2018) y más de 68 mil "estrellas" en su repositorio de Github (Github, Inc., 2018) y siendo utilizado por importantes compañías como Facebook, Instagram, Pinterest, Skype, Tesla, Bloomberg, Wix, Walmart, Uber y muchas más. Véase ***figura 1-3***.

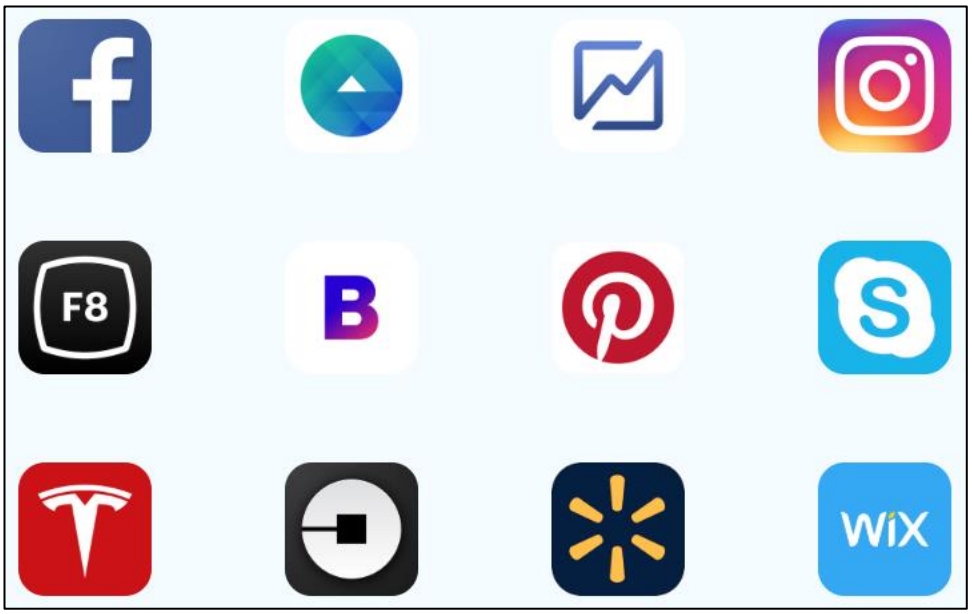


Figura 1-3. Compañías que utilizan React Native (Facebook, 2018)

Entre sus principales *ventajas* están las siguientes:

- *Desarrollo de aplicaciones iOS y Android usando Javascript:* Javascript es el lenguaje de programación más popular en la actualidad (Stack Overflow, 2018), utilizado principalmente en web, esto permite a los programadores que deseen empezar en el desarrollo móvil tener una transición rápida debido a que no deben aprender un lenguaje de programación nuevo como Objective-C o Java, sino mantener el mismo estilo de desarrollo web y aplicarlo para construir aplicaciones móviles.
- *Aplicaciones móviles verdaderamente nativas:* React Native utiliza un enfoque diferente a lo que normalmente nos ofrecen otras herramientas multiplataforma como son la combinación de HTML[23], Javascript y CSS[24] para renderizar el contenido usando "webviews", que a pesar de que puede funcionar trae consigo algunas desventajas principalmente en el rendimiento o no contar con un acceso completo a los componentes gráficos nativos de distintas plataformas. React Native logra transformar código Javascript en componentes gráficos nativos iOS y Android consiguiendo aplicaciones con un "look" and "feel" igual a una aplicación desarrollada enteramente con Objective-C o Java, además de no sacrificar rendimiento ni perder capacidades que ofrecen plataformas nativas.
- *Mejorar la experiencia del desarrollador:* En este aspecto React Native ofrece una característica que la distingue del resto de la competencia y puede ser considerada como una "killer feature"[25] que es la capacidad de permitir ver los cambios casi instantáneamente realizados en el código fuente dentro de la aplicación sin la necesidad de volver a compilar como sucede en el desarrollo tradicional nativo,

[23] HTML: HyperText Markup Language, lenguaje de marcado para la elaboración de páginas web
[24] CSS: Cascade Style Sheets, describen el estilo visual que deben tener un documento HTML
[25] Killer feature: Principal atributo de un producto que lo distingue de la competencia

esto supone una gran ventaja debido a que acorta los tiempos de desarrollo sustancialmente, reduce el impacto negativo que tiene sobre los programadores la larga espera de compilación de un proyecto y dedicar la energía a mejorar las funcionalidades del producto para que estén afin a las necesidades del cliente.

- *Reusabilidad de código:* Con React Native se puede desarrollar simultánemente para las plataformas iOS y Android llegando a obtener una compartición de código del 95%, por ejemplo la aplicación "Facebook Ads Manager" comparte un 87% del código fuente en sus versiones de iOS y Android (Eisenman, 2018) logrando reducir drásticamente los tiempos de desarrollo necesarios para conseguir una aplicación multiplataforma, además de lograrlo en un solo lenguaje de programación que es Javascript y lograr contar con equipos de trabajo que compartan recursos y conocimiento más eficazmente.
- *Gran comunidad de colaboradores y usuarios:* Al ser React Native un proyecto open source el número de contribuidores externos desde su lanzamiento no ha parado de crecer logrando tener una comunidad muy activa pendiente de mejoras que se le pueden realizar a la herramienta, corregir "bugs", solventar inquietudes de usuarios nuevos rápidamente o mantener actualizada la documentación de la herramienta. Además cabe mencionar que todo el proyecto está activamente soportado por Facebook, garantizando un ciclo de vida largo de la herramienta y la seguridad para sus usuarios de que React Native solo podrá mejorar su potencial en los próximos años.

Sin embargo React Native también tiene algunas *desventajas*:

- *Necesidad de trabajar en código nativo Objective-C o Java para determinadas características:* Debido al vasto uso que puede tener una aplicación móvil, es necesario adentrarnos en el código nativo para

implementar módulos o funcionalidades que React Native no ofrece por defecto. Esto se hace sumamente necesario en proyectos complejos que requieren especialmente trabajar con sensores del dispositivo o generar interfaces de usuario más sofisticadas.

- *Ausencia de módulos nativos:* React Native actualmente se considera una herramienta que ha alcanzado una madurez considerable, sin embargo en lo que concierne al número de componentes nativos que React Native ofrece se puede considerar como limitada o todavía subdesarrollados, esto hace que en algunas ocasiones sea necesario construir desde el principio los módulos que necesitamos o hacer uso de librerías alternas que pueden solventar nuestras necesidades pero con la desventaja de que algunas carecen de una documentación apropiada o un soporte a largo plazo.
- *Dependencia de las reglas de Facebook:* Es bueno tener en cuenta que React Native está bajo el soporte y licencia impuesta por Facebook, aunque parezca improbable que la herramienta pueda dejar de ser mantenida por la gran cantidad de usuarios y aplicaciones actuales, en un futuro es incierto los planes de Facebook con respecto a la licencia de uso y los nuevos cambios a los que se deberá adaptar React Native.

1.5. Cuadro de tecnologías

A continuación, se presenta un cuadro resumiendo las principales tecnologías con las que se va a experimentar en el presente proyecto.

Tecnología	Descripción	Licencia	Uso en el proyecto
Raspberry Pi	Computadora de placa única de bajo coste económico capaz de soportar diversos sistemas operativos y ser utilizada para la enseñanza de computación, computación de uso general o como una plataforma de proyectos.	En el caso de que se utilice para fines comerciales es necesario incluir la etiqueta "Powered by Raspberry Pi" en el paquete del producto. En cuestión de software el sistema operativo Raspbian tiene una licencia GPL[26] por lo que únicamente se deben desinstalar los programas de tipo propietarios para que pueda ser utilizado comercialmente.	Cumple la función de controlar la imagen que se está presentando en el monitor externo escuchando los cambios de información en la base de datos Firebase.
Firebase Realtime Database	Es una base de datos NoSql alojada en la nube. La información es almacenada en formato JSON[27] y sincroniza los datos en tiempo real con todos los clientes conectados.	Para proyectos experimentales la plataforma es gratuita con funcionalidades limitadas como almacenamiento reducido y un número máximo de conexiones simultáneas. Se debe subscribir a los paquetes de pago que ofrece la plataforma para utilizarla comercialmente.	Se encarga de almacenar y organizar la estructura de los datos de la aplicación en formato JSON y sincroniza en tiempo real la imagen que debe mostrarse en el monitor con el dispositivo Raspberry Pi.

[26] GPL: En inglés General Public License, es un tipo de licencia de software que garantiza a los usuarios finales la libertad de ejecutar, compartir, modificar o utilizer para fines comerciales el software.
[27] JSON: Es un formato ligero para el intercambio de datosale

React Native	Es un framework basado en React[28] para la construcción de aplicaciones móviles Android e iOS nativas.	Bajo la licencia MIT[29] se puede utilizar libremente para fines experimentales o comerciales. (Facebook, 2018)	Con esta herramienta se construirá la aplicación móvil encargada de mostrar un conjunto de imágenes y permitir seleccionar una para que se visualice en el monitor externo.

Tabla 1-5. Cuadro de tecnologías del proyecto (Aristizábal S., 2018)

Elaborado por: Esteban Aristizábal

[28] React: Herramienta de software Javascript para la construcción de interfaces de usuario en aplicaciones basadas en web.
[29] MIT: Licencia de software sin restricciones al usar, copiar, modificar, compartir, publicar, distribuir, o vender el software.

CAPÍTULO 2. ANÁLISIS Y DISEÑO

En este capítulo se plantea realizar un análisis de requerimientos, definir la metodología para el desarrollo del proyecto y elaborar componentes de diseño para continuar a la fase de desarrollo.

2.1. Análisis de requerimientos

"Comprender los requerimientos de un problema es una de las tareas más difíciles a las que se enfrenta un ingeniero, establece una sólida base para las etapas de diseño y construcción, y sin ella el producto tiene una alta probabilidad de que no cumpla las necesidades del cliente." (Pressman & Maxim, 2015, p. 132)

Una pobre documentación de requerimientos es la razón principal del porque un proyecto de software fracasa (Statista, 2015), conseguir desde un principio una visión clara del producto final radica en la capacidad del ingeniero en comprender en profundidad el problema que se desea resolver, reconocer acertadamente las necesidades que debe solventar el producto terminado y transformar esas necesidades en requerimientos que según el estándar IEEE 830-1998[30] deben cumplir las siguientes características:

- *Correcto:* Cada requerimiento declarado debe encontrarse en el producto.
- *Inequívoco:* Deben tener una sola interpretación.
- *Completo:* Si incluye los siguientes elementos:
    - Se han recolectado todos los requerimientos impuestos por las especificaciones del producto.
    - Permite asegurar una respuesta a toda clase de situaciones y entrada de datos.
    - Definición de todos los términos y referencias.

[30] IEEE 830-1998: Estándar sobre prácticas recomendadas para la especificación de requerimientos de software

- *Consistente:* Coherente con los demás requisitos.
- *Clasificado por importancia y/o estabilidad:* Debe poder ser identificado por su importancia y criticidad.
- *Verificable:* Solamente si existe un proceso factible en el que una persona o máquina pueda comprobar que el producto cumple el requerimiento.
- *Modificable:* Su estructura y estilo debe permitir cambios fáciles, completos y consistentes manteniendo la coherencia dentro de la especificación.
- *Trazable:* El origen del requerimiento es claro y facilita su referencia durante todo el proceso de desarrollo.

Con el concepto claro de las especificaciones que debe tener un requerimiento, corresponde realizar un análisis de las necesidades que deben ser satisfechas por el prototipo que se va a desarrollar.

Para demostrar la integración entre el dispositivo Raspberry Pi y Firebase, se ha propuesto construir una aplicación móvil que controle la presentación de imágenes en un monitor externo conectado a un dispositivo Raspberry Pi. La actualización en tiempo real de las imágenes que se presentan en el monitor se pretende lograr utilizando la plataforma Firebase, haciendo uso específicamente de su producto *Realtime Database* (Base de datos en tiempo real)[31] que nos ofrece sincronizar lo que el dispositivo Raspberry Pi está presentando en el monitor con la imagen que se escoja en la aplicación móvil.

[31] Realtime Database: Base de Datos NoSQL alojada en la nube que almacena datos y sincroniza en tiempo real con cada cliente conectado.

De esta manera se formulan los siguientes requerimientos funcionales y no funcionales:

- **Requerimientos funcionales**
  - La aplicación móvil debe desplegar un conjunto de imágenes que deben ser consumidas desde Firebase.
  - Dentro de la aplicación móvil al seleccionar una imagen, esta debe visualizarse en el monitor externo.
  - El dispositivo Raspberry Pi debe estar conectado a Firebase para controlar la imagen que debe mostrarse en el monitor.
- **Requerimientos no funcionales**
  - La aplicación móvil debe ser desarrollada con el framework React Native para la plataforma Android.
  - La imagen en el monitor externo debe mostrarse en pantalla completa.
  - El tiempo de espera para la actualización de la imagen en el monitor debe ser inmediato.

2.2. Metodología

A partir del análisis de requerimientos realizado, se puede especificar las siguientes características que tiene el proyecto:

- Se pretende construir un prototipo que demuestre la integración entre 2 o más tecnologías, mas no un producto de software terminado.
- El equipo de trabajo es reducido, específicamente de una persona.
- No se busca generar documentación excesiva.
- Se desea que el tiempo de desarrollo sea corto.
- Se han recopilado todos los requerimientos para que el producto cumpla las expectativas.

En base a lo anterior, se plantea escoger una metodología de desarrollo ágil para el proyecto. El desarrollo ágil de software se basa en un enfoque incremental e iterativo, en vez de realizar una planificación extensa en el inicio de un proyecto, las metodologías ágiles producen software funcional en un período corto de tiempo para que exista retroalimentación por parte del cliente y las funcionalidades puedan ser validadas rápidamente. Con este enfoque se logra adaptar un sistema a los cambios de requerimientos que se dan durante el desarrollo, una participación activa por parte de los interesados del proyecto o usuarios finales y la entrega de software que se ajusta a las necesidades del cliente.

A continuación, se presenta un cuadro comparativo con una selección de las metodologías ágiles que se pueden ajustar a las necesidades del proyecto.

	Scrum	**Extreme Programming**	**Kanban**
Tamaño de los proyectos	Todo tipo de proyectos	Pequeños a medianos	Todo tipo de proyectos
Tamaño de equipo	De 5 hasta 10 personas	Menos de 5 personas	De 3 a más personas
Roles	Scrum Master, dueño del producto, cliente y equipo de desarrollo	Tracker[32], cliente, programadores, tester y coach[33]	Roles no definidos
Enfoque de desarrollo	Iterativo, con fases de planificación de sprint[34], desarrollo, revisión del sprint y retrospectiva	Iterativo con fases de planeación, diseño, codificación y pruebas	Se lo adapta a ciclos de vida de otras metodologías
Manejo de tareas	Scrum Master reparte y dirige las tareas en forma de historias de usuario al equipo	Organizadas por prioridad y en un orden a ejecutarse, definidas en forma de historias de usuario	Trabajo dividido en tareas pequeñas y organizado en un tablero de control para ver su estado

[32] Tracker: Rol en la metodología Extreme Programing que se encarga del seguimiento de las tareas
[33] Coach: Rol en la metodología Extreme Programming que guía en las tareas al equipo de trabajo
[34] Sprint: Intervalo de tiempo en que se cumple una iteración de la metodología Scrum

Comunicación	Reuniones diarias	Informal	Reuniones semanales
Documentación	Relacionada al sprint y diseños necesarios	Básica, solo diseños que sirvan para la codificación	Básica
Ventajas	Visibilidad del proyecto, facilidad para implementar cambios, grandes proyectos son divididos en sprints, obtiene retroalimentación del cliente activamente	El software se adapta a cambios al producir prototipos que validan requerimientos, utiliza técnicas como pair programming o TDD[35], reducción de riesgos al crear pruebas de aceptación en la fase de planeación	Propone un mejoramiento continuo al entregar productos rápidamente para obtener retroalimentación, visibilidad de la carga de trabajo del proyecto, incrementar la eficiencia de un equipo de trabajo controlando cuellos de botella
Desventajas	Equipo de trabajo con roles definidos para implementarlo correctamente, carece de un alcance del producto definido, adoptarlo completamente requiere tiempo	Enfocado a producir código y no a un diseño detallado, no conveniente en equipos de trabajo remotos, no adecuado para proyectos complejos	No propone un ciclo de vida de desarrollo a seguir, no hay tareas definidas para cada miembro del equipo, cambios de requerimientos puede causar problemas de coordinación

Tabla 1-1. Cuadro comparativo metodologías ágiles (Raspberry Pi Foundation, 2018)

Elaborado por: Esteban Aristizábal

En base a una comparación de las metodologías ágiles que se pueden implementar en el presente trabajo, se escoge *Extreme Programming (XP)* porque reúne las siguientes características que se ajustan a las necesidades del proyecto:

[35] TDD: Test driven development, requerimientos son convertidos en específicos casos de uso para ser validados

- Prioriza la entrega de software funcional sobre documentación excesiva.
- Sigue el principio KIS (keep it simple)[36], la única función de los diseños es proveer una guía para la implementación.
- El conjunto de tareas a desarrollar necesarias para la entrega del prototipo pueden agruparse en una o más iteraciones cortas.
- Permite escribir pruebas de aceptación para verificar si el prototipo cumple los requerimientos establecidos, esto se hace muy útil porque se desea validar condiciones de integración entre las tecnologías.
- En cada iteración se puede verificar si se están cumpliendo las expectativas del proyecto.
- Es factible implementarlo en equipos reducidos hasta de una persona. (Agarwal & Umphress, 2008)

La ***figura 2-1*** ilustra el ciclo de vida que sigue la metodología XP[37] que engloba cuatro actividades clave que son: planificación, diseño, codificación y pruebas.

[36] KIS: En inglés Keep It Simple, un diseño simple siempre es preferido contra uno complejo
[37] XP: Extreme Programming

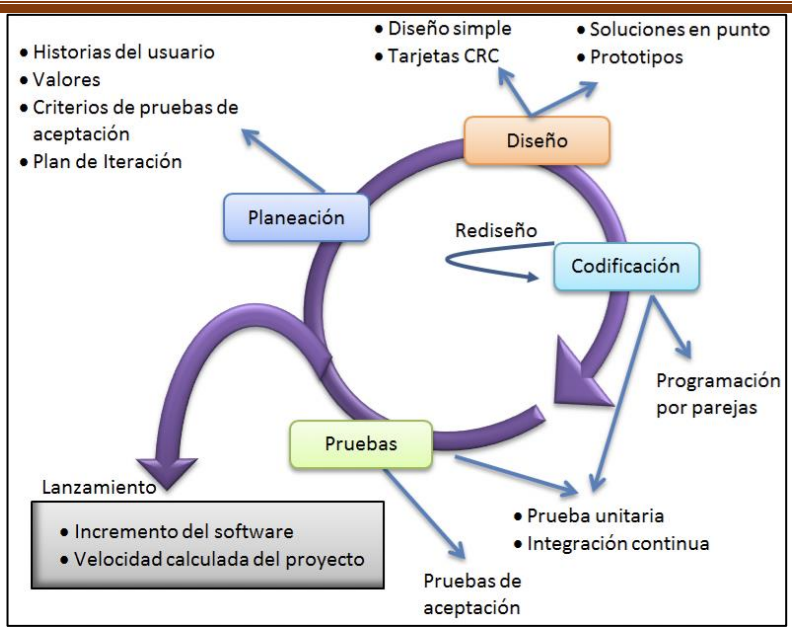


Figura 2-1. Ciclo de vida del proceso Extreme Programming (Pressman & Maxim, 2015)

1. ***Planificación:*** Empieza por una reunión que tiene como objetivo comprender el contexto de negocio del producto y sus posibles funcionalidades. Se formula la visión del producto en forma de *historias de usuario* que encapsulan la definición de un requerimiento conteniendo la suficiente información para que pueda ser estimado en tiempo y dividido en subtareas a implementar. Adicionalmente se le asigna una prioridad, una estimación del esfuerzo necesario para llevar a cabo esa historia y se plantea una prueba de aceptación, que es una condición que debe cumplir el producto para que esa historia de usuario esté completa. Por último se agrupan las historias de usuario que se desean cumplir en la siguiente iteración en un determinado orden de ejecución.
2. ***Diseño:*** El objetivo principal es conseguir diseños que sean simples y fáciles de ser interpretados para su implementación. La creación de

diseños adicionales que no van a ser utilizados es desaprobado. (Pressman & Maxim, 2015, p. 73)

3. ***Codificación:*** Después de que las historias de usuario son subdivididas en tareas que se puedan implementar y la etapa de diseño está completa, lo correcto antes de pasar a la codificación es desarrollar pruebas unitarias[38] que comprobarán automáticamente si una historia de usuario cumple su condición de aceptación, en esta fase se hace uso de la técnica de *pair programming* que tiene como fin implementar historias con 2 programadores trabajando simultáneamente.
4. ***Pruebas:*** Las pruebas unitarias creadas en la fase de codificación son utilizadas inmediatamente para tener una retroalimentación rápida y precisa de los errores que se deben corregir. Las pruebas de aceptación de las historias de usuario son comprobadas con el software construido, y se ratifica si el prototipo construido cumple las demandas o se prepara nuevas historias de usuario para una nueva iteración.

[38] Pruebas unitarias: Método que invoca un código que se quiere probar y determinar si el resultado es el esperado

De acuerdo al análisis de requerimientos elaborado en la ***sección 2.1*** se pueden proponer las siguientes historias de usuario representadas en la ***tabla 2-1***.

CÓDIGO	DESCRIPCIÓN	PRIORIDAD[39]	ESTIMACIÓN[40]
H1	Como usuario *quiero* visualizar un conjunto de imágenes en la aplicación móvil *para* encontrar la que deseo se muestre en el monitor externo.	Normal	12 horas
H2	Como usuario *quiero* que el dispositivo Raspberry Pi al iniciarse se conecte a Firebase y despliegue una imagen *para* visualizarla en el monitor externo.	Normal	8 horas
H3	Como usuario *quiero* que la aplicación móvil me permita seleccionar una imagen *para* que el dispositivo Raspberry Pi se sincronice y la despliegue en el monitor externo.	Normal	4 horas

Tabla 2-2. Historias de usuario (Aristizábal S., 2018)

Elaborado por: Esteban Aristizábal

A cada historia de usuario se la debe dividir en las tareas necesarias que se deben llevar a cabo para implementarla. Este nivel de detalle permite a los programadores tener una descripción clara de lo que deben realizar y comprender la estrategia que debe tomar en la fase de codificación.

La ***tabla 2-2*** permite visualizar el conjunto de tareas que se deben hacer para completar las historias de usuario definidas anteriormente.

[39] Se define clasificar una historia de usuario en 3 niveles de prioridad: alta, normal y baja
[40] El valor de estimación de tiempo hace referencia al número de horas necesarias para terminarla

HISTORIA DE USUARIO	CÓDIGO	DESCRIPCIÓN
H1	T1	Definir esquema de base de datos en Firebase.
	T2	Construir la interfaz gráfica de la aplicación móvil.
	T3	Conectar la aplicación móvil con Firebase.
	T4	Desplegar el conjunto de imágenes consumidas desde Firebase.
H2	T5	Programar en el Raspberry Pi la funcionalidad de mostrar una imagen en pantalla completa.
	T6	Desplegar la imagen en pantalla completa al momento que se inicie el Raspberry Pi.
	T7	Conectar el Rapsberry Pi con Firebase.
	T8	Desplegar la imagen recuperada desde Firebase en el monitor externo.
H3	T9	Al seleccionar una imagen dentro de la aplicación móvil se debe actualizar la base de datos en Firebase.
	T10	El Raspberry Pi debe actualizar la imagen que se muestra en el monitor según lo que recupere desde Firebase

Tabla 2-3. Historias de usuario divididas en tareas (Aristizábal S., 2018)

Elaborado por: Esteban Aristizábal

En las metodologías ágiles el uso de una pizarra de tareas es una práctica que ayuda al equipo de trabajo a seguir el progreso de las tareas y proveer información en tiempo real en caso de que se requiera cambios en el producto. (Stellman & Greene, 2015, p. 37) Por lo tanto al ser XP una metodología basada en conceptos ágiles se puede adoptar el uso de un tablero que permita organizar y visualizar el estado de las tareas en el transcurso del desarrollo del proyecto.

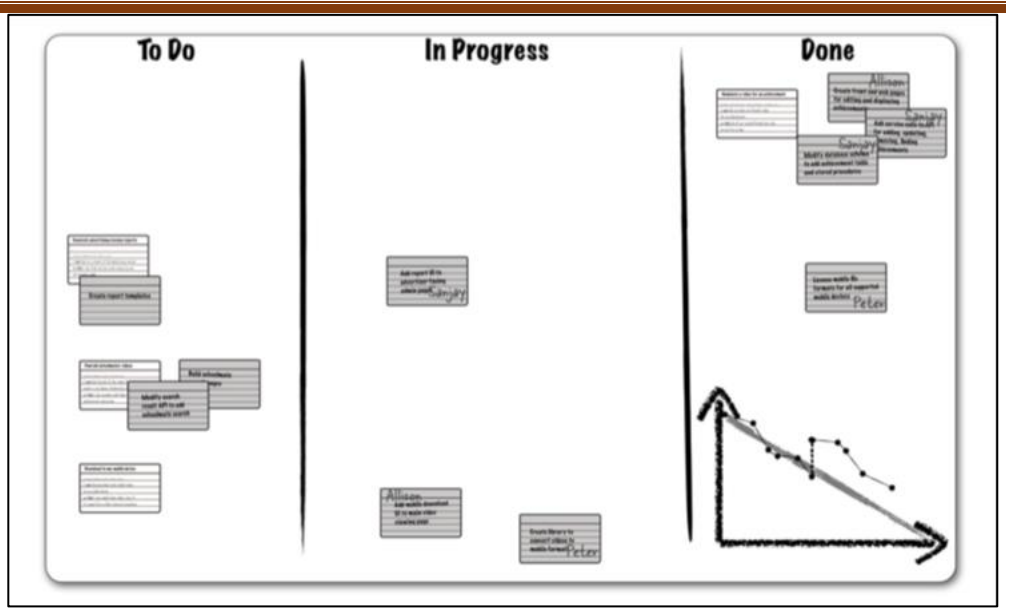


Figura 2-2. Ejemplo de un tablero organizado en columnas de estado por hacer, en progreso y completado (Stellman & Greene, 2015)

Para el presente proyecto la organización de las columnas de estado serán:

1. Por hacer: Son todas las tareas que faltan por terminarse dentro del proyecto.
2. En progreso: En esta columna se encuentran las tareas que se estarán desarrollando.
3. Completadas: Aquí se visualizarán todas las tareas que se hayan terminado y validado su funcionalidad.

La ***figura 2-3*** presenta el tablero de tareas que se va a utilizar en el presente proyecto, se observa que las tareas están ubicadas en el orden que van a ser ejecutadas y con una codificación que corresponde a la historia de usuario que pertenece y el código de tarea asignado.

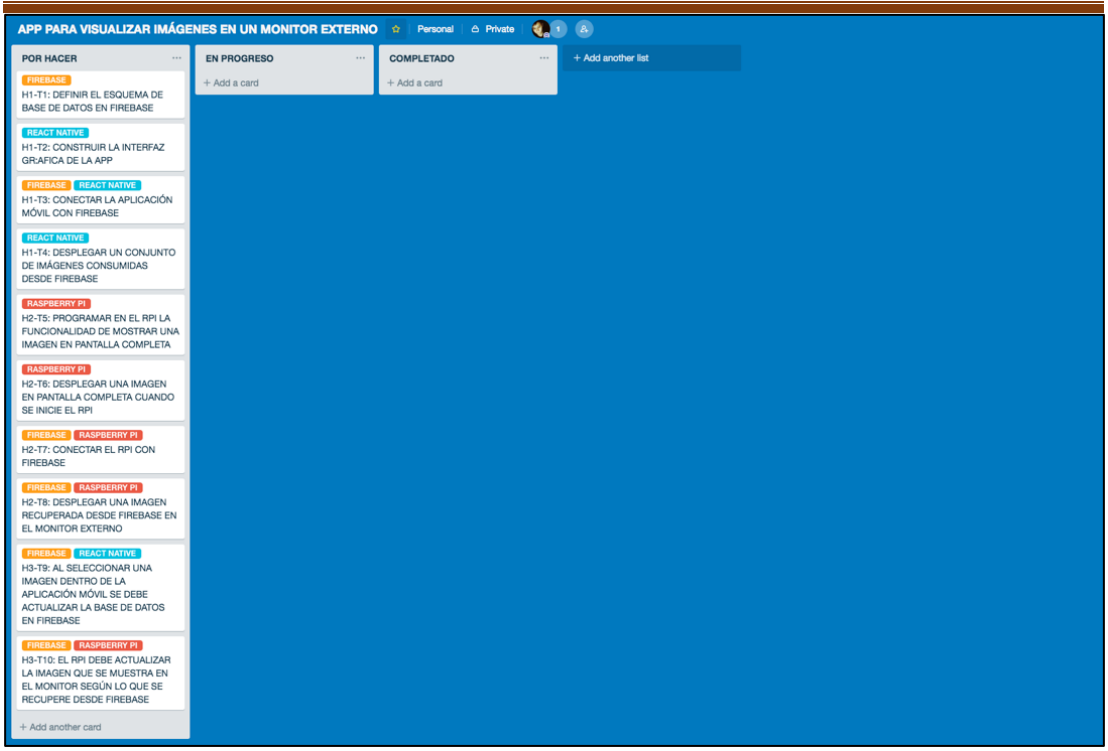


Figura 2-3. Tablero de tareas del proyecto (Aristizábal S., 2018)

Elaborado por: Esteban Aristizábal

Por último se deben definir las *pruebas de aceptación* para cada historia de usuario que se van a tener que cumplir para considerar que el prototipo esté terminado.

CÓDIGO	DESCRIPCIÓN	PRUEBA DE ACEPTACIÓN
H1	Como usuario *quiero* visualizar un conjunto de imágenes en la aplicación móvil *para* encontrar la que deseo se muestre en el monitor externo.	La aplicación móvil debe permitir buscar y mostrar un conjunto de imágenes recuperadas desde Firebase en forma de lista.
H2	Como usuario *quiero* que el dispositivo Raspberry Pi al iniciarse se conecte a Firebase y despliegue una imagen *para* visualizarla en el monitor externo.	Una vez que el Raspberry Pi se inicie debe mostrar en pantalla completa una imagen recuperada desde Firebase.

H3	Como usuario *quiero* que la aplicación móvil me permita seleccionar una imagen *para* que el dispositivo Raspberry Pi se sincronice y la despliegue en el monitor externo.	La aplicación móvil debe permitir seleccionar una imagen e inmediatamente esta debe ser actualizada y mostrada por el monitor externo conectado el Raspberry Pi.

Tabla 2-4. Pruebas de aceptación para las historias de usuario (Aristizábal S., 2018)

Elaborado por: Esteban Aristizábal

Se planea terminar todas las historias de usuario necesarias para culminar el prototipo en una *iteración*, se obtendrá un producto funcional que demuestre la integración entre el dispositivo Raspberry Pi y la tecnología Firebase mediante el desarrollo de una aplicación móvil al culminar un ciclo de desarrollo de la metodología XP.

Con esto la etapa de planificación en la metodología XP escogida para este proyecto se considera terminada y corresponde continuar a la fase de diseño.

2.3. Diseño

La fase de diseño nos permite modelar el sistema o producto que se va a construir, establece una línea base que guía la etapa de implementación y provee detalladamente la arquitectura, estructuras de datos, interfaces y componentes que van a formar parte del sistema (Pressman & Maxim, 2015).

2.3.1. Arquitectura

"La arquitectura de software es la estructura de organización de los componentes de un programa (módulos), la forma en la que éstos interactúan y la estructura de datos que utilizan." (Pressman & Maxim, 2015, p. 226)

La ***figura 2-4*** representa la arquitectura que soportará el prototipo:

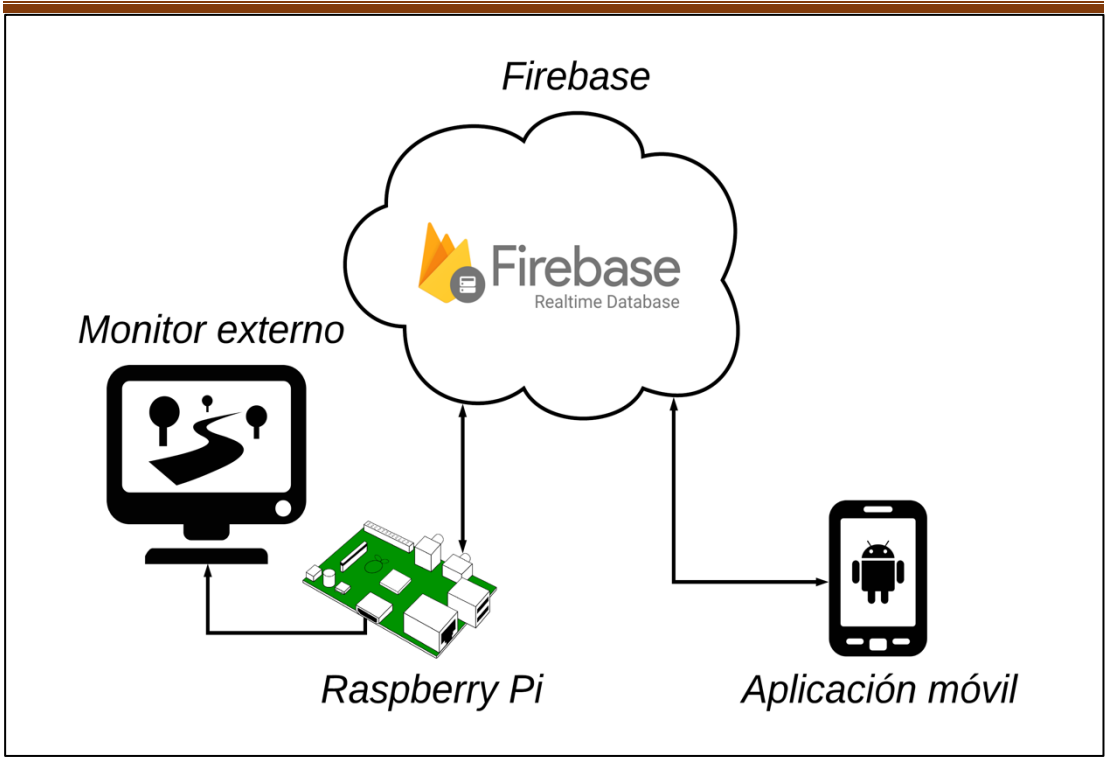


Figura 2-4. Arquitectura del prototipo (Aristizábal S., 2018)

Elaborado por: Esteban Aristizábal

Se distinguen 3 capas que forman parte de la estructura del sistema:

a) *Firebase Realtime Database:* Actúa como el medio de enlace entre la aplicación móvil y el Raspberry Pi, se encarga de almacenar y sincronizar las imágenes que se visualizan en el monitor.

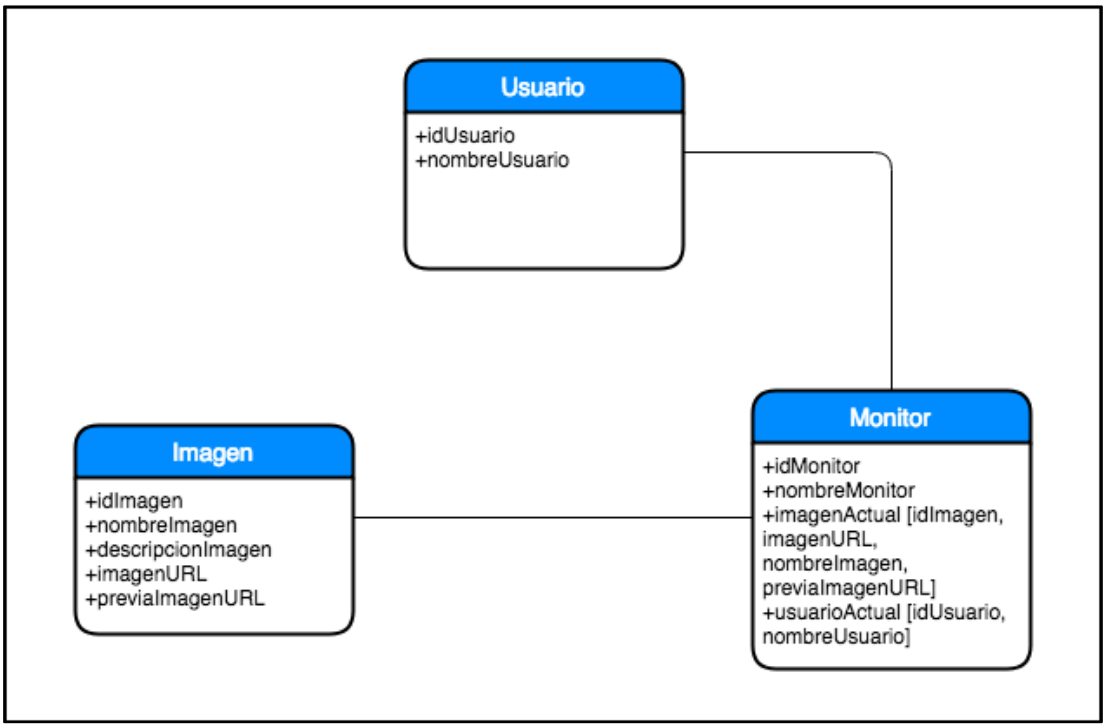
b) Aplicación Móvil: Interactúa con el usuario presentando imágenes consumidas desde Firebase permitiéndo seleccionar la que se presentará en el monitor.

c) Raspberry Pi: Controla la imagen que se muestra en el monitor y la actualiza en caso que el usuario escoja una nueva imagen desde la aplicación.

2.3.2. Estructura de datos

"El diseño del software siempre debe comenzar con el análisis de los datos, pues son el fundamento de todos los demás elementos del diseño. Una vez obtenido el fundamento, se obtiene la arquitectura. Sólo entonces deben aplicarse otros trabajos de diseño." (Pressman & Maxim, 2015, p. 226)

Firebase Realtime Database es una base de datos NoSql alojada en la nube. La información es almacenada en formato JSON[41] y sincroniza los datos en tiempo real con todos los clientes conectados. (Firebase, 2018) Al ser una base de datos NoSql permite especificar una estructura de datos con la que podamos trabajar, para lograr eso es conveniente modelar primeramente un diagrama de Entidad Relación a fin de definir entidades, relaciones y atributos del sistema.



[41] JSON: Es un formato ligero para el intercambio de datos

Figura 2-5. Diagrama Entidad Relación Base de datos NoSql (Aristizábal S., 2018)

Entidades

- **Usuario:** Es la persona que interactúa con la aplicación y selecciona las imágenes que se deben mostrar en el monitor.

Campo	Tipo de dato	Descripción
idUsuario	Int	Identificador único de usuario
nombreUsuario	String	Nombre de usuario

Tabla 2-5. Entidad Usuario (Aristizábal S., 2018)

Elaborado por: Esteban Aristizábal

- **Monitor:** Muestra una imagen escogida por el usuario.

Campo	Tipo de dato	Descripción
idMonitor	Int	Identificador único del monitor
nombreMonitor	String	Nombre del monitor, por ejemplo “monitor en la sala”
imagenActual	Imagen	Se relaciona con la imagen que debe visualizarse en el monitor
usuarioActual	Usuario	Se relaciona al usuario que está conectado al monitor

Tabla 2-6. Entidad Monitor (Aristizábal S., 2018)

Elaborado por: Esteban Aristizábal

- **Imagen:** Contiene las propiedades de una imagen para que pueda ser visualizada en el monitor.

Campo	Tipo de dato	Descripción
idImagen	Int	Identificador único de la imagen
nombreImagen	String	Nombre de la imagen, por ejemplo “paisaje”

descripcionImagen	String	Corta descripción de la imagen, por ejemplo "fotografía de la ciudad"
imagenURL	String	Link web que apunta a la imagen
previaImagenURL	String	Link web que apunta a la imagen en un tamaño reducido para su vista previa dentro de la aplicación

Tabla 2-7. Entidad Imagen (Aristizábal S., 2018)

Elaborado por: Esteban Aristizábal

Relaciones

- Un monitor puede estar asociado a un solo usuario.
- Un monitor puede estar asociado a una sola imagen que es la que se está mostrando en el monitor.

En el modelamiento de una base de datos NoSql la redundancia de datos no genera inconveniente debido a que toda la información es almacenada en un solo documento en formato JSON donde el tamaño físico que puede alcanzar no compromete la capacidad del sistema. Por esta razón es válido *desnormalizar* agregando o duplicando datos redundantes donde se piensa que ayudarán a incrementar el rendimiento de la búsqueda de información, en una base de datos NoSql se procede a incrustar en una entidad la información que tiene de sus relaciones con otras entidades.

A continuación, se modelan en objetos JSON las entidades para que se puedan estructurar en el esquema NoSql de Firebase:

- **Usuario**

```
{
  "idUsuario": "1",
  "nombreUsuario" : "Esteban Aristizabal"
}
```

Figura 2-6. Objeto JSON Usuario (Aristizábal S., 2018)

Elaborado por: Esteban Aristizábal

- **Monitor**

Para la desnormalización en la entidad monitor se requirió incrustar la información relacionada a la imagen que se está mostrando en el monitor y el usuario actual asociado al monitor. Ahora, al recuperar la información del monitor se lo hace en una única operación porque los datos están almacenados juntos.



```
{
  "idMonitor": "1",
  "imagenActual" : {
    "idImagen": "1",
    "imagenURL" : "https://imagen.com/paisaje",
    "nombreImagen" : "Paisaje de Quito",
    "previaImagenURL" : "https://imagen.com/paisaje"
  },
  "nombreMonitor" : "Monitor de la sala",
  "usuarioActual" : {
    "idUsuario": "1",
    "nombreUsuario" : "Esteban"
  }
}
```


Figura 2-7. Objeto JSON Monitor (Aristizábal S., 2018)

Elaborado por: Esteban Aristizábal

- **Imagen**



```
{
    "imagenURL" : "https://imagen.paisaje.jpg",
    "nombreImagen" : "Paisaje de Quito",
    "descripcionImagen" : "Fotografía tomada desde el panecillo",
    "previaImagenURL" : "https://imagen.paisaje.jpg"
}
```


Figura 2-8. Objeto JSON Imagen (Aristizábal S., 2018)

Elaborado por: Esteban Aristizábal

2.3.3. Interfaz de usuario

"El diseño de interfaz de usuario crea un medio eficaz de comunicación entre los seres humanos y la computadora. Siguiendo un conjunto de principios de diseño de la interfaz, el diseño identifica los objetos y acciones de ésta y luego crea una plantilla de pantalla que constituye la base del prototipo de la interfaz de usuario." (Pressman & Maxim, 2015, p. 318)

La interfaz de usuario moldea la percepción que el usuario tiene del software por lo tanto tiene que estar bien hecha, debe cumplir 3 reglas doradas (Mandel, 1997):

- *Dejar el control al usuario:* La interfaz de usuario debe ser construida para reaccionar a las necesidades del usuario y ayudarle a resolver las dificultades que se presenten, se debe lograr que el usuario sienta que puede controlar la interfaz y no que se le están imponiendo restricciones o limitaciones en su uso.
- *Reducir la carga de memoria del usuario:* Mientras más pasos debe realizar el usuario dentro de la aplicación las probabilidades de que cometa un error aumentan. Simplificar el número de interacciones con el sistema para llegar a una solución logra en el usuario una percepción de que el software es fácil e intuitivo de manejar.
- *Hacer que la interfaz sea consistente:* Las reglas de diseño impuestas en el sistema deben respetarse en todas las pantallas y la navegabilidad en todo el sistema debe ser coherente, es decir lograr que el usuario pase de una tarea a otra tarea fácilmente y llevar al usuario donde quiere ir.

La ***figura 2-9*** representa el diseño de pantalla que permite al usuario visualizar un conjunto de imágenes y seleccionar una para visualizarla en el monitor externo, para el diseño de esta interfaz se ha preferido mantener la simplicidad, mostrando todas las imágenes disponibles en forma de lista donde cada elemento contiene una

miniatura de la imagen, su título con una corta descripción y un botón que al presionarlo muestra la imagen seleccionada en el monitor externo.

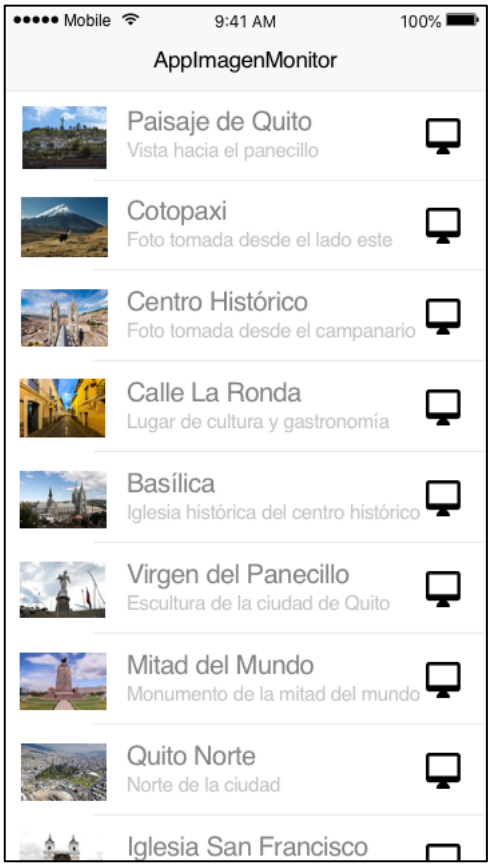


Figura 2-9. Diseño de la interfaz de usuario para la aplicación móvil (Aristizábal S., 2018)

Elaborado por: Esteban Aristizábal

En lo que corresponde a lo que se visualizará en el monitor externo conectado al Raspberry Pi será en todo momento una imagen en pantalla completa. La ***figura 2-10*** muestra el diseño de la aplicación que se ejecutará en el Raspberry Pi.



Figura 2-10. Diseño de la interfaz de usuario para el monitor externo (Aristizábal S., 2018)

CAPÍTULO 3. DESARROLLO

3.1. Herramientas de desarrollo

3.1.1. Raspberry Pi 3 Model B

Computadora de placa única perteneciente a la familia de dispositivos Raspberry Pi de tercera generación, desarrollada por la Raspberry Foundation y lanzada en el año 2016 es el primer modelo de Raspberry Pi que cuenta con un módulo de red inalámbrico y bluetooth integrado. Para el desarrollo de este proyecto se ajusta a las necesidades debido a que es obligatorio que el dispositivo tenga una conexión permanente a internet para conectarse con la plataforma Firebase, además de poseer un conector HDMI[42] para su conexión a un monitor externo.

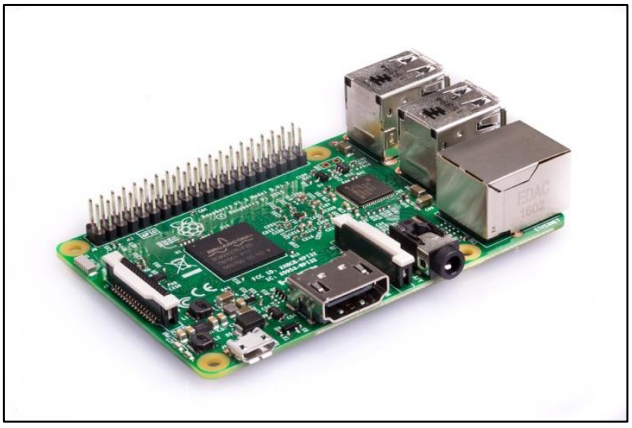


Figura 3-1. Raspberry Pi 3 Model B (Raspberry Pi Foundation, 2018)

Lanzamiento	Feb 2016
Precio	$35
SOC	Broadcom BCM2837
CPU	1.2GHz 64-bit quad-core ARM Cortex-A53
GPU	VideoCore IV
RAM	1 Gb
Puertos USB	4

[42] HDMI: Interfaz para la transmission de audio y video entre dispositivos

Ethernet	10/100 Mbps
Video	HDMI
Audio	Jack 3.5mm, HDMI
SD/MMC	MicroSD
Wi-Fi	802.11n
Bluetooth	Bluetooth 4.1
Dimensiones	85.60 mm × 56.5 mm
Consumo	1.34A
Alimentación	5V MicroUSB/GPIO

Tabla 3-1. Características del Raspberry Pi Model 3 (Raspberry Pi Foundation, 2018)

Elaborado por: Esteban Aristizábal

El sistema operativo para el Raspberry Pi es Raspbian en su versión estable con fecha de publicación 2018-06-29. (Raspbian, 2018) Es adecuado el uso de este sistema operativo para la programación en el Raspberry Pi porque posee un soporte oficial por parte de la Raspberry Pi Foundation, viene con software precargado y provee una interfaz gráfica para interactuar con el dispositivo más fácilmente.

El dispositivo Raspberry Pi se acopla a las necesidades del proyecto por la capacidad de soportar software como en una computadora tradicional, esto la distingue de otros dispositivos como el Arduino[43] que únicamente permite ejecutar un programa al mismo tiempo y suele ser utilizado para aplicaciones que involucren tareas repetitivas con dispositivos electrónicos específicos. Con el Rapberry Pi se pueden instalar programas o librerías de desarrollo sin limitaciones, esto se hace verdaderamente útil para que pueda integrarse con plataformas de internet como Firebase y soportar aplicaciones de software construidas en cualquier arquitectura.

[43] Arduino: Placa de circuito programable para construir aplicaciones de electrónica

3.1.2. Firebase Realtime Database

Este servicio de la plataforma Firebase almacena información de la aplicación en la nube usando una base de datos NoSql en formato JSON y sincroniza esa información con todas las instancias de la aplicación que estén conectadas. Proporciona librerías que se pueden instalar y usar en las plataformas iOS, Android, Web o Flutter[44]. (Firebase, 2018) Estas librerías permiten acceder a la base de datos utilizando un REST API[45] y actualizan la información en tiempo real dentro de la aplicación creando conexiones HTTP que reciben notificaciones del servidor cuando el estado de los datos cambian.

En el Raspberry Pi las librerías de Firebase cumplen la función de notificar al programa que debe cambiar la imagen que se está mostrando en el monitor de acuerdo a un dato que cambia de estado en Firebase cuando el usuario dentro de la aplicación móvil selecciona una nueva imagen.

3.1.3. Node.js

Es un framework de desarrollo que incluye todo lo necesario para ejecutar un programa escrito en Javascript, está construido sobre el motor V8[46] de Google y utiliza una arquitectura orientada a eventos con un I/O[47] de datos asíncrono que lo convierte en un entorno ligero y eficiente. (Patel, 2018)

Con Node.js se pueden desarrollar servicios y aplicaciones tanto del lado del servidor como en el del cliente dejando atrás la separación de estas dos capas donde normalmente se usan lenguajes de programación diferentes, esto supone un gran beneficio porque permite desarrollar aplicaciones enteramente en Javascript y compartir código implementado en el frontend con el backend y viceversa. Además

[44] Flutter: Framework para el desarrollo de aplicaciones móviles multiplataforma creado por Google
[45] REST API: Interfaz de software que se accede mediante peticiones HTTP como GET, PUT, POST o DELETE
[46] V8: Motor de software que convierte código Javascript en código de bajo nivel para que el microprocesador pueda entender
[47] I/O: Se refiere a cualquier proceso que involucre entrada o salida de datos, desde leer un archivo hasta realizar peticiones HTTP

Node.js hace uso del manejador de paquetes "npm"[48] que contiene librerías desarrolladas por la comunidad que se le pueden agregar a la aplicación para acelerar la codificación y obtener funcionalidades que han sido construidas por otros programadores eficientemente.

3.1.4. React Native

A pesar de que existen una multitud de opciones para el desarrollo de aplicaciones móviles multiplataforma actualmente como Ionic, Framework 7, Phonegap, Xamarin y muchos más, el uso del framework React Native para el presente proyecto se justifica por las siguientes razones:

- Es un framework basado en Javascript, por lo que permite una mejor integración con la base de datos de Firebase que almacena la información en formato JSON.
- Actualmente React Native se posiciona como una de las herramientas más utilizadas por equipos de desarrollo en todo el mundo. (g2crowd, 2018) Por lo que aprender a utilizarla es atractivo para en un futuro desarrollar aplicaciones móviles profesionalmente.
- Una de las ventajas que tiene React Native es la de generarle una buena experiencia al desarrollador en la fase de codificación, eliminando los molestos tiempos de espera del refresco de los cambios de una aplicación y acortando drásticamente el tiempo de desarrollo.

El uso del framework React Native para el desarrollo de la aplicación móvil del presente proyecto está enfocado en la plataforma Android. Por lo tanto se deben seguir los pasos necesarios que sugiere la documentación oficial para instalar un ambiente de desarrollo adecuado para construir aplicaciones Android en la computadora.

[48] npm: Node Package Manager

En cuanto a las aplicaciones de software que se utilizarán para acelerar la codificación del proyecto se pueden utilizar desde sencillos editores de texto a sofisticados IDEs[49], pero debido a que el lenguaje principal que se usa es Javascript que no requiere de instalación de software previo para que pueda funcionar, escoger el editor de texto Atom[50] para la fase de implementación es acertado en especial porque ofrece una buena compatibilidad con React Native, además de ser liviano, fácil de usar y provee herramientas como autoidentación de código o detección de errores de tipeo. (Atom Editor, 2018)

3.1.5. Nativebase

Es una librería gratuita que ofrece componentes gráficos genéricos para la construcción de aplicaciones móviles para las plataformas de iOS y Android con el framework React Native. El uso de esta librería acelera la productividad en la fase de codificación del proyecto porque para la creación de la interfaz gráfica de la aplicación móvil se pueden agregar componentes de UI[51] prefabricados que mantienen un estilo visual nativo en las dos plataformas, esto supone una gran ventaja porque no se deben crear interfaces de usuario desde cero ni personalizar estilos visuales para cada plataforma que en aplicaciones complejas puede significar mucho tiempo y requerir ayuda externa de diseñadores gráficos. Además, Nativebase tiene una gran cantidad de componentes gráficos disponibles con una buena documentación y un acabado profesional.

[49] IDEs: Aplicaciónes de software que provee facilidades para el desarrollo de programas de computación
[50] Atom: Editor de texto de código abierto desarrollado por Github
[51] UI: Interfaz de usuario

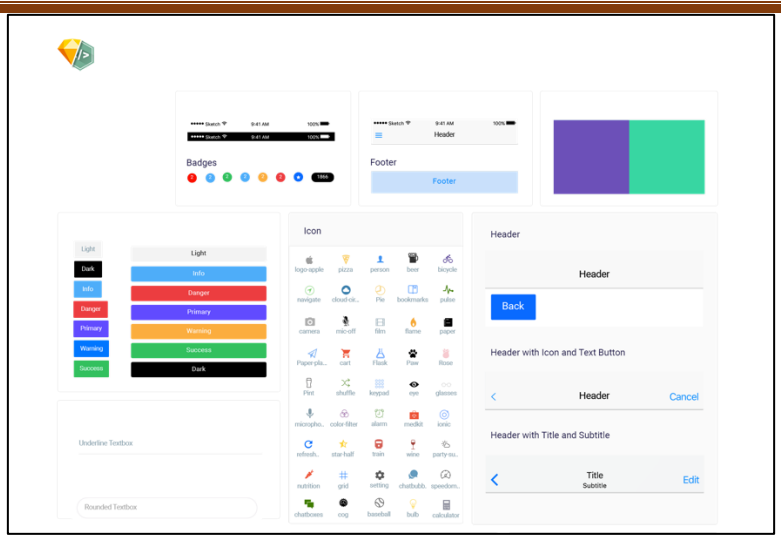


Figura 3-2. Componentes gráficos prefabricados de la librería Nativebase (Nativebase, 2018)

3.2. Seguimiento de la iteración

Se planificó concluir todas las historias de usuario definidas en la ***tabla 2-1*** en una iteración, a continuación se detalla el proceso de desarrollo del prototipo con especial énfasis en como se logró la integración entre el dispositivo Raspberry Pi y la plataforma Firebase.

Las historias de usuario que se proyectaron completar son las siguientes:

H1: Como usuario *quiero* visualizar un conjunto de imágenes en la aplicación móvil *para* encontrar la que deseo se muestre en el monitor externo.

H2: Como usuario *quiero* que el dispositivo Raspberry Pi al iniciarse se conecte a Firebase y despliegue una imagen *para* visualizarla en el monitor externo.

H3: Como usuario *quiero* que la aplicación móvil me permita seleccionar una imagen *para* que el dispositivo Raspberry Pi se sincronice y la despliegue en el monitor externo.

En base a las historias de usuario planteadas se fijó un número de tareas a completarse para satisfacer los requerimientos del prototipo, estas tareas se las realizó en un determinado de ejecución y con la ayuda visual del tablero de tareas para tener en cuenta en el estado que estaban. Véase ***figura 3-3***.

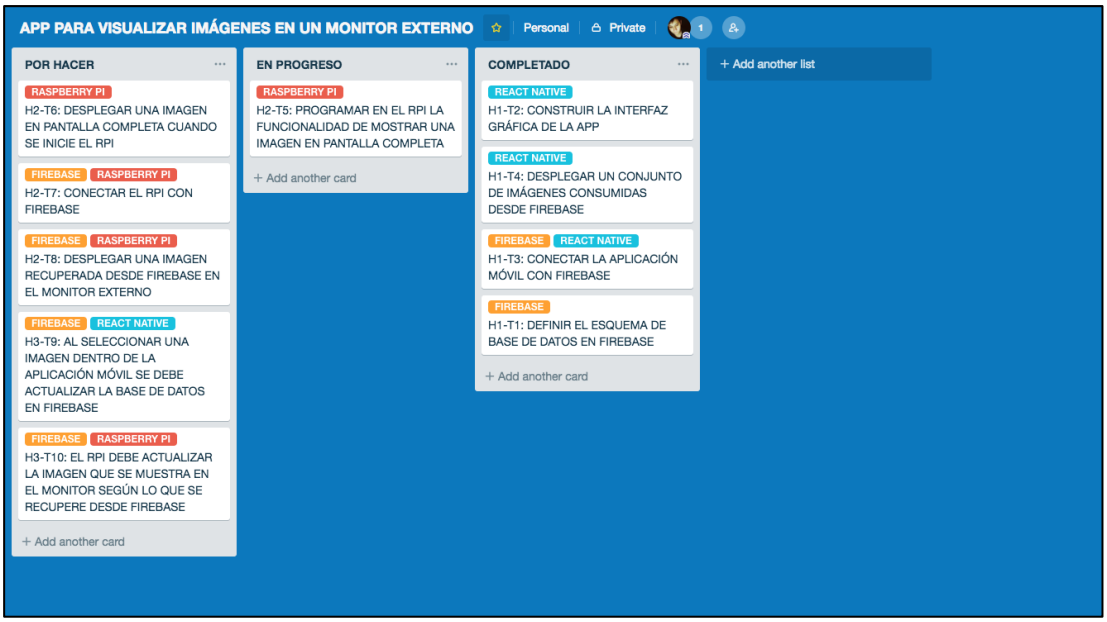


Figura 3-3. Estado del tablero de tareas del proyecto durante el proceso de desarrollo (Aristizábal S., 2018)

Aquí se hace referencia a cada una de ellas detallando la implementación que tuvo que llevarse a cabo para completarlas.

H1-T1: Definir el esquema de base de datos en Firebase.

Antes de pasar a la codificación de la aplicación móvil fue necesario estructurar la base de datos NoSql en Firebase con la que se iba a trabajar para contar con información que pueda ser consumida por la aplicación móvil y el Raspberry Pi. Se procedió a crear una cuenta en la plataforma Firebase para que podamos añadir un proyecto, para el desarrollo de este prototipo se utilizó su producto *Firebase Realtime Database* pero la flexibilidad de la plataforma permite utilizar todos los distintos

productos que ofrece juntos o en separado para el mismo proyecto. La ***figura 3-4*** muestra la consola de nuestro proyecto en Firebase.

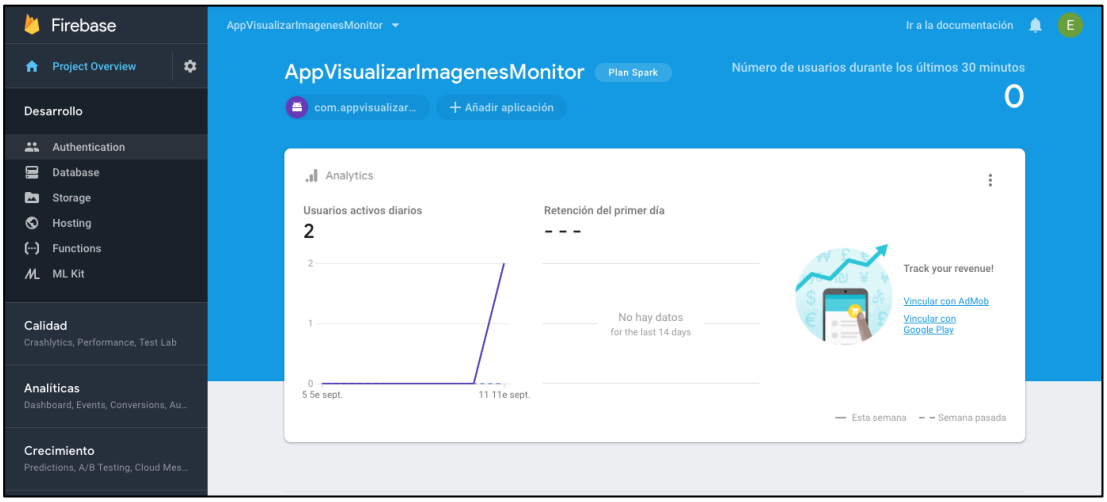


Figura 3-4. Consola del proyecto en Firebase (Aristizábal S., 2018)

Para definir el esquema de base de datos se tomó en cuenta la estructura definida en la ***sección 2.3.2.*** como se muestra en la ***figura 3-5***.

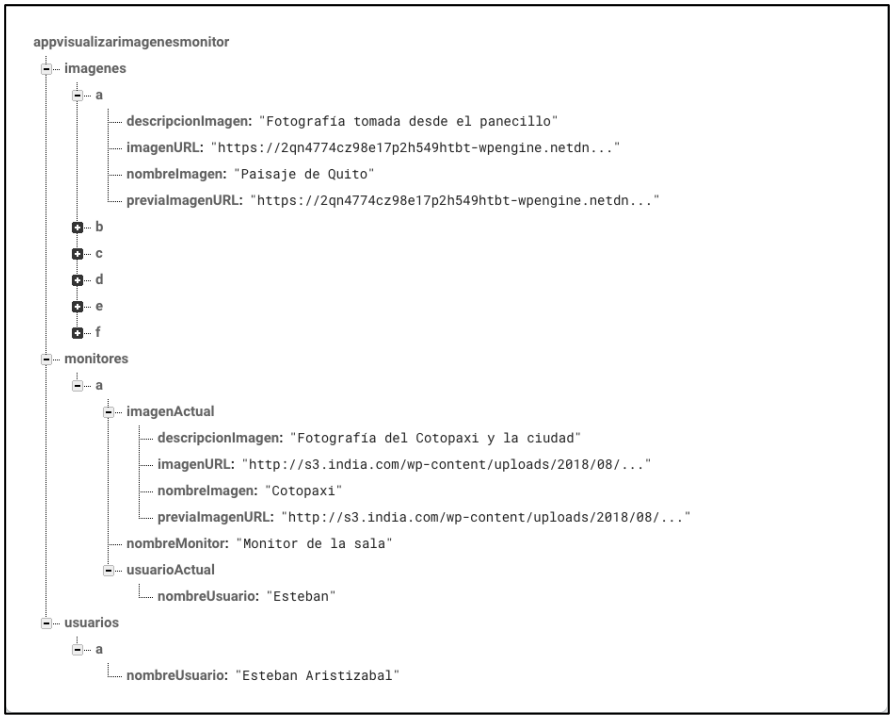


Figura 3-5. Estructura de la base de datos NoSql en Firebase (Aristizábal S., 2018)

En ella se observan las 3 entidades definidas que son imágenes, monitores y usuarios en formato JSON donde cada una de ellas cuenta con registros ingresados con un identificador único e información de prueba.

H1-T2: Construir la interfaz gráfica de la aplicación móvil.

Primeramente se configuró el ambiente de desarrollo en la computadora necesario para la construcción de aplicaciones Android según la documentación del framework React Native. (React Native, 2018) Específicamente para la plataforma Android fue necesario instalar y configurar el software *Android Sdk* que es un conjunto de herramientas para desarrollar aplicaciones Android, debido a la buena documentación esta parte no fue de mucha dificultad y al finalizar se pudo ejecutar la aplicación de inicio del framework React Native como se observa en la ***figura 3-6***.

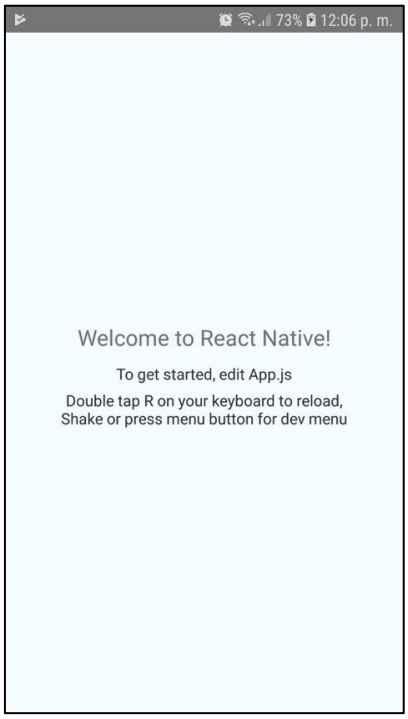


Figura 3-6. Pantalla de la aplicación de inicio del framework React Native (Aristizábal S., 2018)

Antes de comenzar fue necesario instalar la librería de componentes gráficos Nativebase para contar con estilos visuales predefinidos y acortar el tiempo de desarrollo. En la instalación de esta librería se hizo uso del manejador de paquetes "npm" por lo que fue rápido y sencillo. (Nativebase, 2018)

Basándose en el diseño de interfaces realizado en la ***sección 2.3.3.*** se procedió a construir la UI utilizando Nativebase, se hizo uso de los componentes gráficos Container, Card, CardItem, Thumbnail, Header, Text y Title de esta librería como se muestra en la ***figura 3-7***.



Figura 3-7. Pantalla con el diseño de la interfaz gráfica de la aplicación terminado (Aristizábal S., 2018)

H1-T3: Conectar la aplicación móvil con Firebase.

Debido que el framework React Native se basa en Javascript y está destinado a producir aplicaciones móviles, el ambiente de desarrollo que proporciona es muy similar a la web, tal es el caso que para integrar Firebase con la aplicación móvil se utilizó la guía que proporciona la documentación para un ambiente con *Node.js*. (Firebase, 2018) Firebase se instala como un paquete y se lo puede importar en cualquier clase dentro del proyecto sin complicaciones.

Un paso importante dentro de la configuración de Firebase fue utilizar las credenciales de nuestro proyecto en la plataforma que se muestran en la ***figura 3-8***.

```
apiKey: "AIzaSyBEMPE0Fq8_5CzlXj754KyQuybm8lN23L0",
authDomain: "appvisualizarimagenesmonitor.firebaseapp.com",
databaseURL: "https://appvisualizarimagenesmonitor.firebaseio.com",
projectId: "appvisualizarimagenesmonitor",
storageBucket: "appvisualizarimagenesmonitor.appspot.com",
messagingSenderId: "1064280677074"
```

Figura 3-8. Credenciales del proyecto en Firebase (Aristizábal S., 2018)

H1-T4: Desplegar un conjunto de imágenes consumidas desde Firebase.

En este punto fue muy importante comprender la lógica de programación que propone React Native para la construcción de interfaces gráficas que se basa en la declaración de componentes que son bloques de código que representan una pieza visual dentro de la UI. Estos componentes se encargan de desplegar, actualizar y consumir información en base a las interacciones del usuario. Algo muy importante que se debe mencionar es la naturaleza *reactiva* que tienen estos componentes que es el refresco de la información instantáneamente que se muestra al usuario sin necesidad de recargar la aplicación o de eventos predefinidos, esto es muy conveniente para aplicaciones en tiempo real como la que se está desarrollando y especialmente con Firebase logrando una interacción fluida y natural.

Para el consumo de imágenes fue necesario seguir la documentación que proporciona Firebase para la lectura y escritura en su base de datos. (Firebase, 2018)

```
export function getGalleryContentOn(callback) {
  db.ref('imagenes').on('value', (snapshot) => {
    callback(snapshot.val());
  });
}
```

Figura 3-9. Código que consume imágenes desde Firebase (Aristizábal S., 2018)

El código anterior tiene la funcionalidad de conectarse y leer las imágenes que se encuentran en Firebase, pero también de mantener una conexión con la base de datos y en caso de que un registro dentro de la entidad monitores cambie el método expuesto notifica a la aplicación móvil del cambio y por lo tanto las imágenes que se muestran en la aplicación cambian en tiempo real.

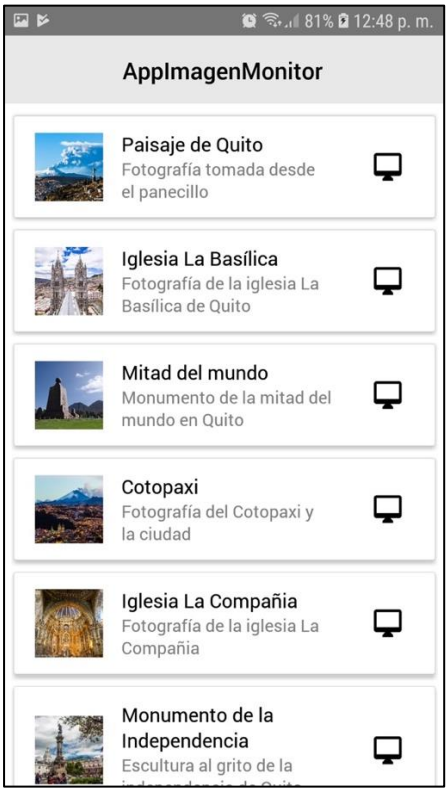


Figura 3-10. Pantalla de la aplicación mostrando un conjunto de imágenes recuperadas desde Firebase (Aristizábal S., 2018)

H2-T5: Programar en el Raspberry Pi la funcionalidad de mostrar una imagen en pantalla completa.

Comenzando con la programación necesaria en el dispositivo Raspberry Pi primeramente fue analizar el conjunto de alternativas disponibles que permitan mostrar una imagen en pantalla completa programáticamente. Una de las alternativas que se analizó fue el uso de un programa hecho enteramente en *Python*[52], este lenguaje de programación es también compatible con Firebase y permite lograr la funcionalidad que se deseaba pero la razón por la que fue descartado fue que implicaba familiarizarse con un lenguaje de programación nuevo, además de que Python es muy limitado para la construcción de interfaces gráficas sofisticadas. La alternativa que se escogió para la implementación de esta funcionalidad fue utilizar un enfoque de desarrollo web, es decir todo lo que respecta a la programación necesaria para el despliegue de la imagen y la integración con Firebase se lo hizo con lenguajes de programación web que fueron HTML, Javascript y CSS, esto facilitó en gran medida la codificación y la posibilidad de aumentarle en un futuro las funcionalidades al prototipo programando como normalmente se haría para una aplicación web.

Para lograr la funcionalidad se construyó una aplicación web que muestre una imagen en toda la proporción del navegador y se activó la opción mostrar en pantalla completa del programa.

[52] Python: Lenguaje de programación interpretado de alto nivel para aplicaciones de software

```
<html>
<head>
<title>My Title</title>
<style type="text/css">
html {
  background-color: black;
}
</style>
</head>
<body>
  <div style='position:absolute;z-index:0;left:0;top:0;width:100%;height:100%'>
    <img style='width:100%;height:100%'>
  </div>
</body>
</html>
```

Figura 3-11. Código que despliega la imagen en un monitor externo conectado a un Raspberry Pi (Aristizábal S., 2018)

H2-T6: Desplegar una imagen en pantalla completa cuando se inicie el Raspberry Pi

Se configuró al Raspberry Pi para que al momento de iniciarse ejecute una rutina que obligue al navegador web a iniciarse en pantalla completa. Esto requirió modificar un archivo de configuración del sistema que dicta los programas que deben iniciarse por defecto, este procedimiento se lo hizo en una terminal de sistema y fue de fácil configuración.

```
sudo nano /home/pi/.config/lxsession/LXDE-pi/autostart
```

Figura 3-12. Comando para modificar los programas de inicio en un Raspberry Pi (Aristizábal S., 2018)

```
@xset s off
@xset -dpms
@xset s noblank
@chromium-browser --kiosk http://google.com/  # load chromium after boot and point to
```

Figura 3-13. Valores modificados en el archivo "autostart" (Aristizábal S., 2018)

Adicionalmente a esto fue necesario configurar un servidor web local que maneje el contenido web que debemos mostrar, por esta razón se instaló el framework Node.js que ofrece la opción de configurar uno para nuestras aplicaciones. Después de

instalar Node.js en el Raspberry Pi (instructables, 2018) se procedió a crear un archivo con nombre *server.js* en el mismo directorio de la aplicación web que se encargue de levantar el servidor web local.

```
var express = require('express');
var app = express();
var path    = require("path");
app.get('/', function (req, res) {
    res.sendFile(path.join(__dirname+'/index.html'));
})

var server = app.listen(8081, function () {
   var host = server.address().address
   var port = server.address().port

   console.log("Servidor web local iniciado", host, port)
})
```

Figura 3-14. Código que inicia el servidor web local de Node.js (Aristizábal S., 2018)

Después de esto únicamente fue necesario añadir el comando *sudo node /home/pi/Documents/visualizarmonitorpi/server.js* en el archivo de configuración del sistema que contiene los comandos que se ejecutan al iniciar el Raspberry Pi. Finalmente fue necesario modificar la dirección del sitio web en el archivo *autostart* con el siguiente nuevo valor *@chromium-browser --kiosk --disable-session-crashed-bubble --disable-infobars --incognito http://127.0.0.1:8081* para que al iniciarse el navegador se redirija al servidor web local.

H2-T7: Conectar el Raspberry Pi con Firebase

Debido a que la programación en el Raspberry Pi es enfocado a la web la integración de Firebase se la realizó siguiendo la documentación que proporciona Firebase para web. (Firebase, 2018)

```
<html>
<head>
<title>My Title</title>
<script src="https://www.gstatic.com/firebasejs/5.4.1/firebase.js"></script>
<script>
  var config = {
    apiKey: "AIzaSyBEMPE0Fq8_5CzlXj754KyQuybm8lN23L0",
    authDomain: "appvisualizarimagenesmonitor.firebaseapp.com",
    databaseURL: "https://appvisualizarimagenesmonitor.firebaseio.com",
    projectId: "appvisualizarimagenesmonitor",
    storageBucket: "appvisualizarimagenesmonitor.appspot.com",
    messagingSenderId: "1064280677074"
  };
  firebase.initializeApp(config);
</script>
<style type="text/css">
html {
  background-color: black;
}
</style>
</head>
<body>
  <div style='position:absolute;z-index:0;left:0;top:0;width:100%;height:100%'>
    <img style='width:100%;height:100%'>
</div>
</body>
</html>
```

Figura 3-15. Código que integra Firebase en la aplicación web del Raspberry Pi (Aristizábal S., 2018)

H2-T8: Desplegar una imagen recuperada desde Firebase en el monitor externo.

Para implementar esta funcionalidad fue necesario programar un método que recupere la imagen desde Firebase que actualmente está asociada al monitor para mostrarla en la aplicación web y también que escuche si hay un cambio en ese registro para actualizar el monitor con una nueva imagen.

```
<html>
<head>
<title>My Title</title>
<script src="https://www.gstatic.com/firebasejs/5.4.1/firebase.js"></script>
<script>
  var config = {
    apiKey: "AIzaSyBEMPE0Fq8_5CzlXj754KyQuybm8lN23L0",
    authDomain: "appvisualizarimagenesmonitor.firebaseapp.com",
    databaseURL: "https://appvisualizarimagenesmonitor.firebaseio.com",
    projectId: "appvisualizarimagenesmonitor",
    storageBucket: "appvisualizarimagenesmonitor.appspot.com",
    messagingSenderId: "1064280677074"
  };
  firebase.initializeApp(config);
  var db = firebase.database();
  var imagenActual = db.ref('monitores/a/imagenActual');
  imagenActual.on('value', function(snapshot) {
   var imagen = snapshot.val();
   var tag = document.getElementById('myimage').src=imagen.imagenURL;
 });
</script>
<style type="text/css">
html {
  background-color: black;
}
</style>
</head>
<body>
  <div style='position:absolute;z-index:0;left:0;top:0;width:100%;height:100%'>
    <img style='width:100%;height:100%'>
</div>
</body>
</html>
```

Figura 3-16. Código que recupera la imagen que se visualiza en el monitor externo desde Firebase (Aristizábal S., 2018)

H2-T9: Al seleccionar una imagen dentro de la aplicación móvil se debe actualizar la base de datos en Firebase.

La aplicación web en el Raspberry Pi actualiza una nueva imagen en base al cambio de un registro existente en Firebase, específicamente el registro *imagenActual* del monitor como se puede ver en la figura **3-17,**



Figura 3-17. Registro imagenActual en Firebase (Aristizábal S., 2018)

Por lo tanto en la aplicación móvil se implementó el evento que al seleccionar una imagen por parte del usuario modifique el registro *imagenActual* del monitor en Firebase para que la aplicación web en el Raspberry Pi escuche y actualice la imagen.

```
export function updateGalleryItemToDisplayInUserFrame(galleryItem, userFrameKey) {
  db.ref('monitores/a/imagenActual').update({
    imagenURL: galleryItem.imagenURL,
    previaImagenURL: galleryItem.previaImagenURL,
    nombreImagen: galleryItem.nombreImagen,
    descripcionImagen: galleryItem.descripcionImagen
  });
}
```

Figura 3-18. Código que actualiza el registro imagenActual del monitor en Firebase (Aristizábal S., 2018)



Figura 3-19. Registro imagenActual actualizándose en tiempo real en Firebase (Aristizábal S., 2018)

H2-T10: El Raspberry Pi debe actualizar la imagen que se muestra en el monitor según lo que se recupere desde Firebase.

Para completar la integración se debe codificar el método que se observa en la ***figura 3-20*** en la aplicación web del Raspberry Pi, cumple la función de recuperar desde Firebase los datos de la imagen del registro *imagenActual* y actualizar en tiempo real el código HTML que despliega la imagen cada vez que hay una actualización desde la aplicación móvil.

```
imagenActual.on('value', function(snapshot) {
 var imagen = snapshot.val();
 var tag = document.getElementById('myimage').src=imagen.imagenURL;
});
```

Figura 3-20. Código que recupera y actualiza en tiempo real la imagen que se despliega en el monitor (Aristizábal S., 2018)

Finalmente en la ***figura 3-21*** se puede visualizar una imagen recuperada desde Firebase en pantalla completa en un monitor conectado al Raspberry Pi.

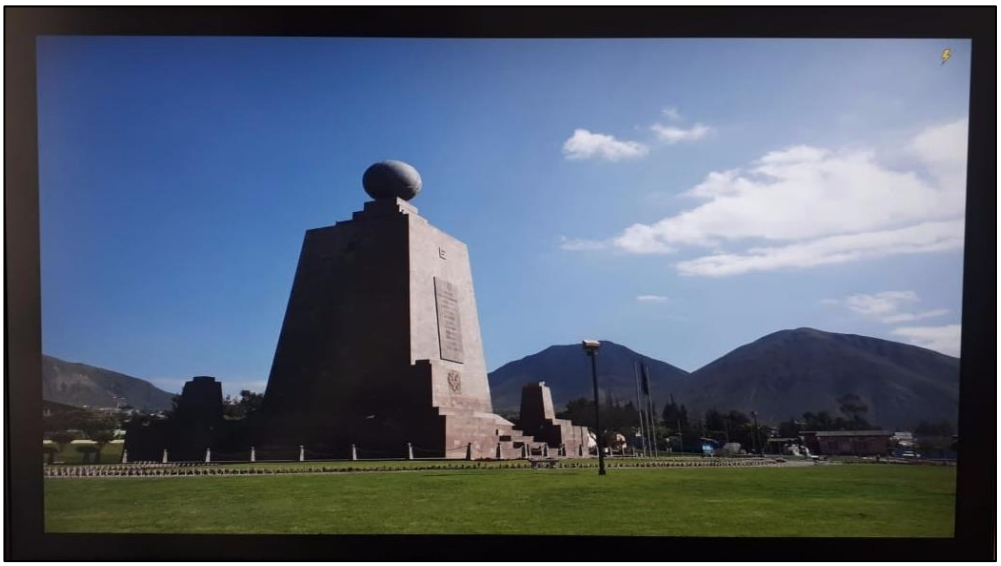


Figura 3-21. Imagen de prueba visualizada en el monitor (Aristizábal S., 2018)

Con esto se han completado las tres historias definidas al inicio de esta sección y corresponde continuar a las pruebas de la aplicación para verificar que el prototipo construido satisface las condiciones y resuelve el problema planteado.

3.3. Pruebas de la aplicación

En la metodología Extreme Programming (XP) las pruebas de aceptación se derivan de las historias de usuarios que han sido implementadas como parte de un lanzamiento de software. (Pressman & Maxim, 2015, p. 75)

A continuación se presentan los resultados de las pruebas de aceptación definidas en la ***tabla 2-3*** aplicadas al prototipo desarrollado junto con un comentario de lo obtenido.

- **Prueba de aceptación H1**

Historia de usuario: Como usuario *quiero* visualizar un conjunto de imágenes en la aplicación móvil *para* encontrar la que deseo se muestre en el monitor externo.

Prueba de aceptación: La aplicación móvil debe permitir buscar y mostrar un conjunto de imágenes recuperadas desde Firebase en forma de lista.

Resultado: **Satisfactorio**

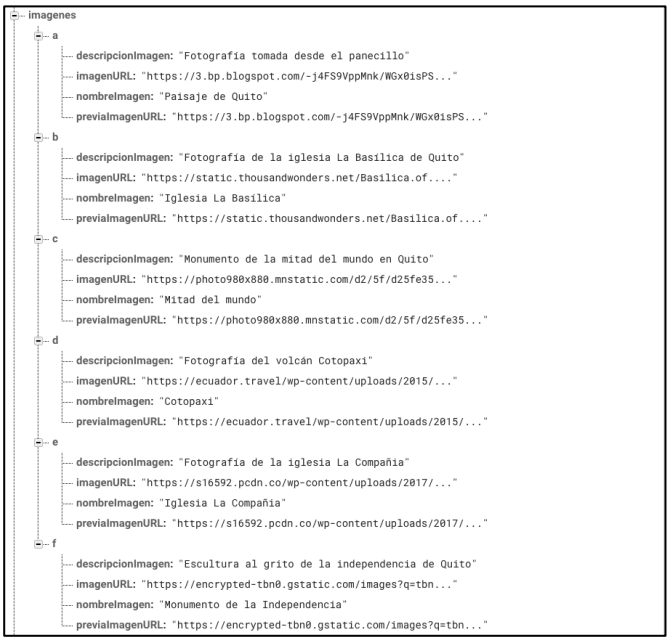


Figura 3-22. Resultado 1 prueba de aceptación H1 (Aristizábal S., 2018)

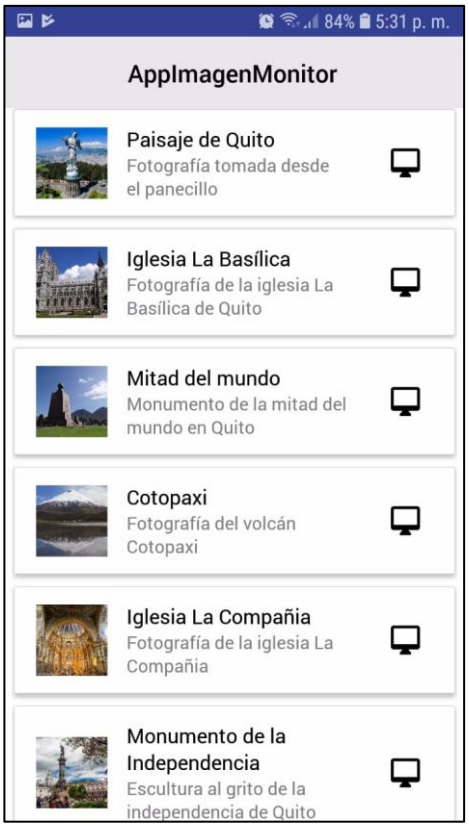


Figura 3-23. Resultado 2 prueba de aceptación H1 (Aristizábal S., 2018)

Comentario: La ***figura 3-23*** representa una captura de la aplicación móvil desplegando todas las imágenes recuperadas desde Firebase. Se pueden comprobar que las imágenes obtenidas son correctas con la ***figura 3-22*** que muestran los registros de imágenes en la base de datos Firebase. Se evidenció el buen rendimiento que ofrece React Native para el desarrollo de aplicaciones móviles al ver que el tiempo carga de imágenes dentro de la aplicación fue instantáneo y al desplazarse entre las imágenes fue muy fluido y preciso.

- **Prueba de aceptación H2**

Historia de usuario: Como usuario *quiero* que el dispositivo Raspberry Pi al iniciarse se conecte a Firebase y despliegue una imagen *para* visualizarla en el monitor externo.

Prueba de aceptación: Una vez que el Raspberry Pi se inicie debe mostrar en pantalla completa una imagen recuperada desde Firebase.

Resultado: **Satisfactorio**

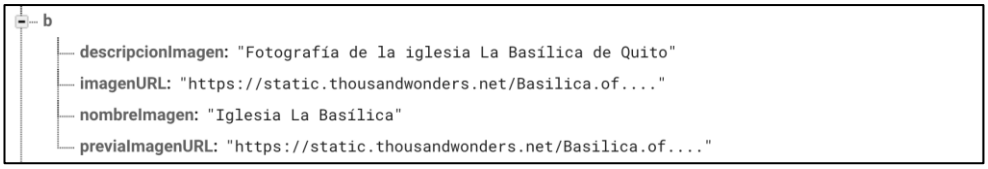


Figura 3-24. Resultado 1 prueba de aceptación H2 (Aristizábal S., 2018)



Figura 3-25. Resultado 2 prueba de aceptación H2 (Aristizábal S., 2018)

Comentario: La imagen que se muestra en la ***figura 3-25*** en el monitor externo conectado al Raspberry Pi corresponde con la que se encuentra almacenada en Firebase como se observa en la ***figura 3-24***, una vez que el Raspberry Pi inició el tiempo estimado que tardo en cargar la imagen en pantalla completa fue de 5 segundos por lo que se propone en un futuro mejorar este aspecto.

- **Prueba de aceptación H3**

Historia de usuario: Como usuario *quiero* que la aplicación móvil me permita seleccionar una imagen *para* que el dispositivo Raspberry Pi se sincronice y la despliegue en el monitor externo.

Prueba de aceptación: La aplicación móvil debe permitir seleccionar una imagen e inmediatamente esta debe ser actualizada y mostrada por el monitor externo conectado el Raspberry Pi.

Resultado: **Satisfactorio**

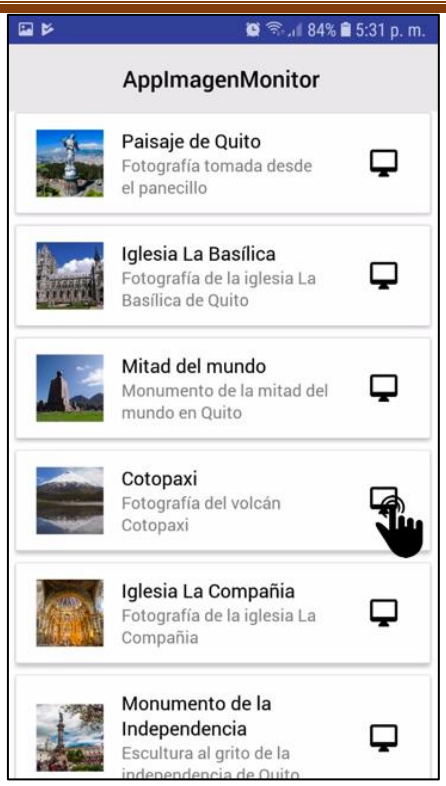


Figura 3-26. Resultado 1 prueba de aceptación H3 (Aristizábal S., 2018)

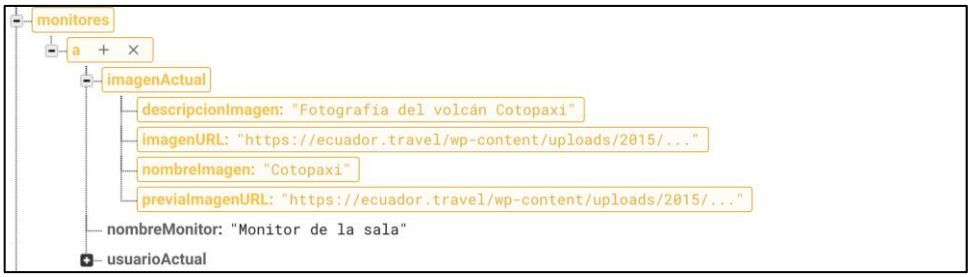


Figura 3-27. Resultado 2 prueba de aceptación H3 (Aristizábal S., 2018)



Figura 3-28. Resultado 3 prueba de aceptación H3 (Aristizábal S., 2018)

Comentario: Esta prueba de aceptación fue la más importante porque comprobó la integración completa entre el Raspberry Pi, Firebase y la aplicación móvil, el procedimiento que se siguió para ejecutarla fue primeramente dentro de la aplicación móvil seleccionar la imagen con el nombre de "Cotopaxi" como se muestra en la ***figura 3-26***, después capturar el momento en que la base de datos en tiempo real de Firebase actualiza el registro *imagenActual* del monitor que asociamos como se puede ver en la ***figura 3-27***, y por último se tomó una fotografía del monitor en el momento que la imagen se actualizó por la que la seleccionamos como lo representa la ***figura 3-28***. El tiempo que se tardó en sincronizar el Raspberry Pi la imagen en el monitor fue de aproximadamente 1 segundo, a pesar de que este tiempo es corto se recomienda hacer el uso de un componente gráfico que permita al usuario saber que se está cargando una nueva imagen.

CAPÍTULO 4. CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

- Con el desarrollo de este trabajo se comprueba que un dispositivo Raspberry Pi puede ser controlado remotamente utilizando Firebase, por lo tanto, la forma de implementación utilizada es válida para la construcción de prototipos más complejos que se basen en la arquitectura planteada.
- El potencial de aplicaciones que pueden ser construidas utilizando el Raspberry Pi con Firebase son muy amplias, específicamente en sistemas de automatización que requieran el uso de sensores que captan y envían datos en tiempo real, donde Firebase tiene una buena compatibilidad por su característica de sincronizar la información de su base de datos a todos los dispositivos conectados instantánemaente.
- Se pueden explorar usos comerciales del prototipo por ejemplo pantallas de uso decorativo que sustituyan a los cuadros comunes donde el usuario pueda elegir obras de arte que se muestren en la pantalla desde la aplicación móvil.
- La metodología ágil Extreme Programming (XP) se ajustó correctamente a las necesidades de este proyecto porque se logró entregar un prototipo funcional en corto tiempo, se mantuvo simple la documentación y se hicieron solo los diseños necesarios para entender como implementar la solución.
- En la fase de planificación de la metodología XP se plantearon historias de usuario que se dividieron en las tareas necesarias a llevarse a cabo para completarlas. En la fase de codificación fue en donde se obtuvo los mayores beneficios de tener un número de tareas definidas a implementarse, logrando que el proceso de desarrollo sea organizado y en un orden específico.

- Tener las pruebas de aceptación de las historias de usuario previamente definidas fue ventajoso para tener una idea de que es lo que se quería lograr y tener una referencia si el prototipo estaba cumpliendo los requisitos a la medida que se iban concluyendo las tareas.
- La codificación de la aplicación móvil con el framework React Native tuvo una curva de aprendizaje complicada en un inicio porque se introducen conceptos avanzados como son la construcción de la interfaz gráfica utilizando componentes o el estado reactivo que tienen los datos dentro de la aplicación, pero una vez que se comprendió la lógica de programación que dicta el framework se facilitó en gran medida el desarrollo de la aplicación y se logró evidenciar el potencial que tiene esta herramienta de producir aplicaciones multiplataforma casi indistinguibles de las desarrolladas nativamente más eficientemente.
- El uso de la librería Nativebase para acelerar la construcción de la interfaz gráfica de la aplicación fue acertado porque se importaron los componentes UI que se necesitaron al proyecto y no se invirtió tiempo en crear nuevos componentes gráficos como botones, títulos o listas.

4.2. Recomendaciones

- En muchas ocasiones al desarrollar prototipos de software es común preferir comenzar con la codificación y olvidarse de realizar un correcto análisis de requerimientos y diseño del producto primero, este enfoque no trae ningún beneficio y resulta mejor entender el problema que se va a resolver profundamente antes de pasar a la implementación.
- Una de las restricciones a la hora de escoger la metodología de este proyecto es que lo iba a realizar enteramente una persona, a pesar de que casi todas las metodologías son enfocadas a implementarse en equipos de trabajo, lo recomendable es buscar una que se acople más a las características del proyecto y ajustarla a las capacidades de la persona.
- Al iniciarse en el desarrollo de aplicaciones móviles con React Native no es recomendable comenzar desde cero independientemente porque tiene una curva de aprendizaje complicada en un principio, lo correcto es primero aprender en base a recursos didácticos como tutoriales o cursos online la lógica de programación que dicta el framework.
- Si no se desea utilizar un teclado, monitor y mouse para la programación en el Raspberry Pi existe la posibilidad de acceder remotamente desde una computadora al dispositivo reduciendo los costos de desarrollo y facilitando la programación.
- Para trabajar con la base de datos NoSql de Firebase es beneficioso comprender como funcionan este tipo de bases de datos y el modelamiento de información en ellas.
- Investigar el uso del lenguaje de programación *Python* en el Raspberry Pi porque ofrece una buena integración con componentes electrónicos como sensores y muchas de las plataformas de “big data”, minería de datos y automatización se basan en este lenguaje de programación.

- Investigar el uso del lenguaje de programación *R*[53] en el Raspberry Pi porque puede ser utilizado en aplicaciones basadas en la recolección y envío de datos hechos por el dispositivo, como también en proyectos que involucren minería de datos y automatización.
- Se recomienda que en la carrera de Ingeniería en Sistemas y Computación de la PUCE[54] se estudie más detalladamente el funcionamiento y modelamiento en las bases de datos NoSQL porque actualmente son muy utilizadas en aplicaciones de software enfocadas a la minería de datos o análisis de la "big data".
- Igualmente se recomienda la experimentación con el dispositivo Raspberry Pi en el transcurso de la carrera porque puede abrir puertas al desarrollo de proyectos basados en el IoT[55] que están obteniendo popularidad actualmente.

[53] R: Es un entorno y lenguaje de programación con un enfoque de análisis estadístico y gráfico
[54] PUCE: Pontificia Universidad Católica del Ecuador
[55] IoT: Internet de las cosas

GLOSARIO

A

API

Conjunto de librerías y funciones construidas para que puedan ser utilizadas por otro software. · 50

B

Backend

Capa de una aplicación de software alojada en el servidor. · 50

Bluetooth

Tecnología de comunicación entre dispositivos de corto alcance · 16

C

CPU

Circuito electrónico en una computadora que ejecuta las instrucciones de un programa de software. · 16

CSS

Lenguaje que define la apariencia de un documento orientado a la web · 60

F

Framework

Herramienta de software que provee funcionalidades genéricas para la construcción de sistemas · 51

Frontend

Capa de una aplicación de software que mantiene la interacción con el usuario. · 50

G

GPU

Circuito electrónico especializado en la creación de elementos visuales en un computador. · 16

H

Hardware

Partes físicas tangibles de un sistema informático. · 17

K

Kanban

Sistema de manejo de tareas dentro de un proyecto basado en mejorar la eficiencia de un equipo de trabajo. · 30

L

Linux

Sistema operativo de código abierto basado en Unix que puede ser instalado en computadoras de uso general · 17

P

Python

Lenguaje de programación interpretado de alto nivel para aplicaciones de software · 60

R

RAM

Random Access Memory en inglés, se utiliza como memoria de trabajo de computadoras y otros dispositivos para el sistema operativo. · 12

S

Scrum

Metodología de desarrollo de software basado en conceptos ágiles y una entrega de software funcional en intervalos de tiempo definido. · 30

Sistemas Embebidos

Sistemas de computación diseñados solamente para realizar una o más funciones específicas. · 13

SOC

En inglés system on chip, circuito integrado que integra todos los módulos que componen un computador. · 15, 48

Software

Soporte lógico de un sistema informático, que comprende un conjunto de programas qeu deben realizar específicas tareas. · 17

V

V8

Motor de software desarrollado por Google que convierte código Javascript en código de bajo nivel para que el microprocesador pueda entender. · 50

W

Webview

Contenido web renderizado en una vista nativa móvil · 22

BIBLIOGRAFÍA

Agarwal, R., & Umphress, D. (Enero de 2008). Extreme programming for a single person team. Auburn, Alabama, United States.

Alpert, S. (14 de Junio de 2018). *State of React Native 2018.* Obtenido de React Native: https://facebook.github.io/react-native/blog/2018/06/14/state-of-react-native-2018

Apache Hadoop. (2017). *What Is Apache Hadoop?* Recuperado el 05 de Mayo de 2017, de http://hadoop.apache.org

Aristizábal S., E. P. (2018). *Integración entre las tecnologías Raspberry Pi y Firebase mediante el desarrollo de una aplicación móvil utilizando React Native que controle la presentación de imágenes en un monitor externo.* Quito: PUCE.

Atom Editor. (2018). *Atom Editor*. Obtenido de Atom Editor: https://atom.io/

Eisenman, B. (2018). *Learning React Native: Building Native Mobile Apps with JavaScript.* Sebastopol: O'Reilly Media, Inc.

Facebook. (2018). *facebook/react-native.* Obtenido de Github: https://github.com/facebook/react-native/blob/master/LICENSE

Facebook. (6 de Septiembre de 2018). *React Native: A framework for building native apps using React*. Obtenido de React Native: https://facebook.github.io/react-native/

Firebase. (2018). *Add Firebase to your JavaScript Project.* Obtenido de Firebase: https://firebase.google.com/docs/web/setup?authuser=0

Firebase. (2018). *Firebase*. Obtenido de Firebase: https://firebase.google.com/

Firebase. (2018). *Read and Write Data on the Web.* Obtenido de Firebase: https://firebase.google.com/docs/database/web/read-and-write?authuser=0

g2crowd. (2018). *Best Mobile Development Frameworks Software.* Obtenido de g2crowd: https://www.g2crowd.com/categories/mobile-development-frameworks

Github, Inc. (6 de Septiembre de 2018). *facebook/react-native*. Obtenido de Github: https://github.com/facebook/react-native

instructables. (2018). *HOW TO BUILD A WEBSITE ON A RASPBERRY PI, WITH NODE.JS, EXPRESS, AND MONGODB...PART 1* . Obtenido de instructables: https://www.instructables.com/id/How-to-Build-a-Website-on-a-Raspberry-Pi-With-Node/

Mandel, T. (1997). *The elements of user interface design.* New York: John Wiley & Sons, Inc.

Mayur Tanna, H. (2018). *Serverless Web Applications with React and Firebase: Develop real-time applications for web and mobile platforms.* Birmingham: Packt Publishing.

McDonald, C. (26 de Octubre de 2017). *Data Modeling Guidelines for NoSQL JSON Document Databases*. Obtenido de MAPR: https://mapr.com/blog/data-modeling-guidelines-nosql-json-document-databases/

Nativebase. (2018). *Nativebase*. Obtenido de Nativebase: https://nativebase.io/

Occhino, T. (26 de Marzo de 2015). *React Native: Bringing modern web techniques to mobile.* Obtenido de Facebook Code: https://code.fb.com/android/react-native-bringing-modern-web-techniques-to-mobile/

Pajankar, A. (2017). *Raspberry Pi Image Processing Programming: Develop Real-Life Examples with Python, Pillow, and SciPy.* Nashik: Apress.

Patel, P. (18 de Abril de 2018). *What exactly is Node.js?* Obtenido de Medium: https://medium.freecodecamp.org/what-exactly-is-node-js-ae36e97449f5

Pressman, R., & Maxim, B. (2015). *Software Engineering: A Practitioner's Approach.* New York: Mc Graw Hill Education.

Raspberry Pi Foundation. (2018). *Raspberry Pi*. Recuperado el 2 de Mayo de 2017, de https://www.raspberrypi.org

Raspbian. (2018). *Raspbian*. Obtenido de Raspbian: raspbian.org

React Native. (2018). *Getting Started.* Obtenido de React Native: https://facebook.github.io/react-native/docs/getting-started.html

Stack Overflow. (2018). *Stack Overflow Developer Survery 2018*. Obtenido de Stack Overflow: https://insights.stackoverflow.com/survey/2018/

Statista. (2015). *Leading reasons for software project failure according to developers worldwide, as of 2015.* Obtenido de Statista: https://www.statista.com/statistics/627648/worldwide-software-developer-survey-project-failure/

Stellman, A., & Greene, J. (2015). *Learning Agile: Understanding Scrum, XP, LEAN and Kanban.* Sebastopol: O'Reilly Media.

Upton, E. (4 de Mayo de 2018). *Raspberry Pi 3 Model B+ on Sale at $35.* Obtenido de Raspberry Pi Blog: https://www.raspberrypi.org/blog/raspberry-pi-3-model-bplus-sale-now-35/