PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR
FACULTAD DE INGENIERÍA
ESCUELA DE SISTEMAS Y COMPUTACIÓN

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

TEMA:

DESARROLLO DE UNA APLICACIÓN MÓVIL PARA LA PROMOCIÓN DE EVENTOS PARA EL DEPARTAMENTO DE ACTIVIDADES COMPLEMENTARIAS DE LA PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR (PUCE). CASO DE ESTUDIO: PROYECTO CULTURAL “VOCES DESDE LA MITAD DEL MUNDO”.

AUTOR:
VÍCTOR JOSÉ SILVERIO TORRES

DIRECTOR:
JORGE ALEJANDRO ALARCÓN MENA

QUITO DM, 2021

“Passion glows within your heart
Like a furnace burning bright
Until you struggle through the dark
You'll never know the joy in life.”

Dream Theater – Illumination Theory

DEDICATORIA

Dedico este trabajo a mis padres, Diana y Néstor, por brindarme su apoyo y amor incondicional. A mi hermano, Néstor, que, entre risas y debates, me inspira a ser cada día alguien mejor. A mi novia, Stephanie, por ser la chispa de felicidad en mi vida, y la fuente de mi fuerza.

Víctor José Silverio Torres

AGRADECIMIENTOS

Quiero agradecer a todas las personas que me apoyaron en la realización de este trabajo. A mis padres, por darme la oportunidad de cumplir mis sueños. A mi director, Ing. Jorge Alarcón, por brindarme su guía a lo largo de la carrera. A mis amigos, la familia que uno escoge. A Juan Carlos Velasco, por creer en este proyecto para poder hacerlo real.

Víctor José Silverio Torres

RESUMEN

La presente disertación de grado describe el proceso de ingeniería de software seguido para la construcción de una aplicación móvil con el objetivo de facilitar la promoción de eventos y difusión de información relacionada al proyecto cultural “Voces desde la Mitad del Mundo”. Al final del estudio, se reconoce la importancia de cumplir con los procedimientos de una metodología de desarrollo de software enfocada al cambio, a fin de adaptarse correctamente al trabajo a pesar de los obstáculos encontrados en la elaboración del sistema. Para este motivo, se empleó “Extreme Programming” (XP), llevando a cabo sus seis fases: exploración, planificación, iteraciones, producción, mantenimiento y muerte. El patrón de desarrollo empleado fue una arquitectura combinada entre “Modelo-Vista-Controlador” y “sin servidor”. El producto de software ha sido construido con la ayuda de marcos de trabajo orientados al lenguaje de programación “JavaScript”, como lo son “React Native” y “Expo”.

ABSTRACT

This degree dissertation describes the software engineering process followed for building a mobile app to facilitate event promotion and information diffusion related to the cultural project “Voces desde la Mitad del Mundo”. At the end of the study, one recognizes the importance of complying with the procedures of a change-oriented software development methodology, correctly adapting despite the obstacles encountered in the development of this system. For this reason, “Extreme Programming” (XP) was used, under its six phases: exploration, planning, iterations, production, maintenance, and death. The development pattern used was a combination of “Model-View-Controller” and “serverless” architectures. The software product was built based upon “JavaScript” programming language-oriented frameworks, such as “React Native” and “Expo”.

TABLA DE CONTENIDOS

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1 INTRODUCCIÓN

1.1 Justificación

La tecnología garantiza el crecimiento y éxito de los proyectos, sea estos empresariales. Una aplicación web, una aplicación móvil, o incluso un sistema personalizado enfocado a la gestión, pueden proporcionar una ventaja considerable a causa de que facilita el trabajo de las personas dentro del proyecto y genera un impacto positivo al público objetivo, dando como resultado una rápida y eficiente propagación de la información.

El desarrollo de una aplicación móvil para la promoción de eventos resulta importante para el proyecto cultural “Voces desde la Mitad del Mundo” gracias a que facilitará y mejorará la tarea de promocionar los eventos. De la misma forma, el tema resulta un reto personal, debido a que con mis conocimientos tecnológicos aportaré al equipo de organización del festival.

Igualmente, el presente trabajo tiene un efecto positivo en cuanto a la situación de salud en Ecuador en los años 2020 y 2021, causada por el COVID-19. La aplicación móvil, permite promocionar eventos remotamente, sin contacto físico. De esta manera, es posible apuntar a una audiencia mayor sin requerir de los medios tradicionales de comunicación, respetando las medidas de confinamiento.

El abordaje al proyecto cultural permite reconocer un problema real al momento de realizar las preparaciones el Festival Coral que se realiza en julio cada año. Dicho problema se refiere a la dificultad para llegar al público con información cargada en línea y que pueda ser consultada en cualquier momento. Se pretende, por lo tanto, desarrollar una aplicación móvil para resolver estos inconvenientes presentando datos de forma organizada y rápida, con notificaciones a las personas que se subscriben a la aplicación).

1.2 Planteamiento del problema

Parafraseando a Campillo y Herrero (2015) que indica que la planificación de un evento está necesariamente relacionada con la promoción. También menciona que, para cumplir con una eficaz estrategia de promoción de marketing, es importante apuntar a un adecuado público objetivo. Uno puede estar de acuerdo que, para lograrlo, es crucial el poder compartir la información por todos los medios posibles. De esta manera, uno de los problemas más comunes es identificar los canales de información óptimos tomando en cuenta el limitado presupuesto que las organizaciones.

El Festival Coral ha logrado expandirse hacia medios como periódicos, radio, páginas web y hasta redes sociales. Sin embargo, no se ha explorado aún el aspecto tecnológico de las aplicaciones móviles como canal de difusión de la información y los respectivos beneficios que podría aportar.

De esta forma, el presente trabajo de disertación propone el desarrollo de una aplicación móvil como un nuevo medio informativo que permita cumplir con los intereses comunicativos del proyecto cultural. Esto último se refiere a apoyar en la fase de promoción del Festival Coral y a servir como un medio de consulta referente para sus respectivos eventos.

Cabe resaltar que el presente producto de software fue una propuesta realizada para el proyecto cultural, y no fue una necesidad requerida por el mismo. Esto tendrá implicaciones sobre los requerimientos de la aplicación y sobre la participación del cliente sobre la construcción de la aplicación móvil.

A partir de esto, se puede plantear la siguiente pregunta principal de investigación:

- ¿Cómo podría una aplicación móvil facilitar la tarea de promocionar eventos para el departamento de actividades complementarias de la Pontificia Universidad Católica del Ecuador, relacionados con el proyecto cultural “Voces desde la Mitad del Mundo”?

Y, bajo el mismo criterio, se plantean las siguientes preguntas de investigación secundarias:

- ¿Qué requerimientos del programa deberán ser incluidos en la aplicación móvil?
- ¿Qué herramientas tecnológicas serán las más eficientes para el desarrollo de la aplicación móvil?
- ¿Qué metodología de desarrollo de software se será la óptima en el proceso de construcción de la aplicación móvil?
- ¿Cómo efectuar el proceso de ingeniería de software según la metodología determinada?

1.3 Objetivos

1.3.1 Objetivo General

- Desarrollar e implantar una aplicación móvil para la promoción de eventos para el departamento de actividades complementarias de la Pontificia Universidad Católica del Ecuador (PUCE).

1.3.2 Objetivos Específicos

- Analizar los requerimientos del software para determinar los componentes de la aplicación móvil.
- Examinar las herramientas tecnológicas idóneas para el desarrollo de la aplicación móvil.
- Determinar la metodología de desarrollo de software óptima que se seguirá en el proceso de construcción de la aplicación móvil.
- Realizar el proceso de ingeniería de software según la metodología determinada.

1.4 Marco Teórico

Partiendo de la explicación de Luna (2016) respecto a la definición de un requerimiento es: “una condición o capacidad que debe estar presente en un sistema o componente de un sistema para satisfacer un contrato, estándar, especificación u otro documento formal”. Con esto, el autor explica que es importante generar una ingeniería de requerimientos, la cual la define como “el proceso de recopilar, analizar y verificar las necesidades del cliente para un sistema de software” (Luna, 2016).

Igualmente, es pertinente señalar que la importancia de la ingeniería de requerimientos radica en el aporte que se le da al sistema en cuestión en varios aspectos como la gestión estructurada de necesidades, la mejora en la planificación de cronogramas, disminución de costos, mejoramiento en la calidad del software, así como en la comunicación entre los miembros del equipo, y la prevención de rechazos de parte del cliente (Quintero Vélez, 2016). Este análisis deja en claro la necesidad de una ingeniería de requerimientos sobre la aplicación a desarrollar.

Esta aplicación contará con tecnología móvil la cual, según Herrera y Fennema (2011), se define como “un término genérico que describe la habilidad para usar tecnología sin ataduras, es decir, no conectada físicamente o que pertenece a entornos remotos o móviles, no estáticos”. Esto es útil para el proyecto gracias a la ventaja fundamental que se presenta a continuación:

> Otras muchas áreas, como la de turismo, sanidad, educación o cultura, pueden utilizar aplicaciones móviles que les permitan acercar su oferta de servicios a las personas e incluso interactuar con ellas y conocerlas mejor para así poder orientar mejor las estrategias operativas y de marketing al público objetivo. (Carrasco Usano, 2015)

También, es adecuado describir las herramientas que se estudiarán en el trabajo de disertación. En primer lugar, Gutiérrez (2014) afirma que un framework es “una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación”. El autor, de igual forma, comenta que un framework aporta en ámbitos tales como “acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones”.

En cuanto a la actualización de contenidos de la aplicación móvil, una herramienta que podría aportar mucho es Cloud Firestore de Google Firebase. Según la documentación oficial de Google Developers (2017), Firebase es una base de datos escalable que permite mantener los datos sincronizados con agentes que trabajan a tiempo real. Se adaptan a los dispositivos móviles con una adaptación directamente desde la plataforma de Google, lo cual puede contribuir sustancialmente al desarrollo de la aplicación.

Es importante definir que “una metodología es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información” (Ortega & Camacho, 2019). Los autores también indican que “una metodología está formada por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a los desarrolladores de sistemas a elegir las técnicas más apropiadas en cada momento del proyecto” (Ortega & Camacho, 2019)

Para la creación del software en el plan de disertación, entonces, se busca un correcto proceso de ingeniería de requerimientos que, con ayuda de herramientas como un framework y una base de datos fácil de sincronizar, dé paso a una metodología de desarrollo de software para guiar el proceso de construcción de una aplicación con tecnología móvil.

1.5 Información Sobre la Organización

Para encaminar correctamente la construcción del presente proyecto, es valiosa la participación del usuario. Para ello se ha solicitado la asistencia de personas que conforman el lado del cliente. De esta manera, se contó con el apoyo de Juan Carlos Velasco, director del proyecto cultural; con Andrea Rodríguez, secretaria general del festival coral; y con Freddy Coello, jefe del diseño gráfico.

En la sección 9, referente a los Anexos, se pueden encontrar las actas de reuniones que evidencian el trabajo en asociación con el cliente.

1.6 Costos del Proyecto

Es pertinente resaltar el costo que pudo haber representado la realización del presente proyecto. Para ello se han tomado en cuenta los costos directos como el salario del equipo de desarrollo o el costo de la licencia de desarrollador de Google Play, los costos indirectos como el valor aproximado del internet y electricidad. Para este propósito, se consideraron las fechas de inicio y final del proyecto, resultando en que la duración total del proyecto fue de tres meses. A continuación, se presenta una tabla resumiendo los costos del proyecto.

Descripción	Costo Unitario (USD)	Meses	Costo Total (USD)
Salario	937,24	3	2 811,72
Cuenta de desarrollador Google Play	25	Pago único	25
Costos Indirectos	25	3	75
TOTAL			***2 911,72***

Tabla 1 Silverio, V. (2021) Costos del proyecto [Tabla].

Para el valor del salario del equipo de desarrollo, el cual consta de una sola persona, se tomó en cuenta el valor registrado en el documento de salarios mínimos sectoriales del Ministerio de Trabajo (2021), para el cargo de “Ingeniero de procesos/proyectos de telefonía móvil”, según la naturaleza del presente proyecto. Asimismo, como se indica en Appypie (2020), la licencia para tener una cuenta de desarrollador en Google Play tiene un valor de 25 USD.

Como se puede apreciar en la Tabla 1, por lo tanto, el costo aproximado del proyecto, en los tres meses de trabajo, es de 2 911, 72 USD.

2 CAPÍTULO 1: DETERMINACIÓN DE LA METODOLOGÍA DE DESARROLLO DE SOFTWARE

En un primer momento, se debe establecer una guía para la construcción de la aplicación propuesta en el presente trabajo de titulación, para lo cual fue preciso determinar la metodología de desarrollo de software. Este capítulo trata la examinación de las variantes de estas (ágiles o tradicionales), como también tres marcos de referencia tentativos en base a criterios definidos, a fin de poder precisar el mejor acercamiento para esta aplicación móvil en específico. Al final del apartado se encuentra una comparativa entre las metodologías junto con la conclusión respectiva.

El proceso de construcción del software comienza con la identificación de la metodología y del ciclo de vida o fases, puesto que se siguen las directrices explicadas a profundidad en este capítulo.

2.1 Las Metodologías de Desarrollo de Software

Los autores Maida y Pacienzia (2015) exploran a profundidad el concepto de Metodología de Desarrollo de Software, explicando que se trata de una agrupación de técnicas y métodos interrelacionados con el fin de abordar las diferentes fases de un proyecto de desarrollo. En otras palabras, definen que “una metodología de desarrollo de software es un marco de trabajo que se usa para estructurar, planificar y controlar el proceso de desarrollo de sistemas de información”.

Por esta razón, es imperativo realizar un análisis objetivo y completo sobre la metodología que mejor se adapte al proyecto propuesto. Con esto, se pretende que el marco de referencia proporcione una guía simple pero efectiva en aspectos como las fases de desarrollo, actividades a realizar y qué documentación se debe generar, con el objetivo de generar un producto de calidad en tiempos factibles.

Por el contrario, si un proyecto de software no sigue las directrices de una metodología de desarrollo, o si se escogiese una metodología no óptima, existe una baja probabilidad de éxito en el proyecto, como lo señalan Maida y Pacienzia (2015).

2.2 Metodologías Ágiles Contra Metodologías Tradicionales

Uno de los primeros aspectos a considerar en el análisis de las metodologías de desarrollo, es la determinación de la modalidad de la metodología con la que se va a trabajar. Es decir, escoger si la presente investigación se basa en las metodologías ágiles o las tradicionales. Esta sección muestra la diferencia entre las dos para continuar con un estudio más a fondo de lo que conlleva cada metodología.

2.2.1 Metodologías Tradicionales

Como lo explican Molina et al. (2018), en su comparativa sobre metodologías ágiles y tradicionales, los marcos de referencia tradicionales nacieron en la década de los 60 con el propósito de poner un fin a la desorganización causada por el desarrollo de software en grandes cantidades. Proponía una forma de estructurar y definir la ingeniería de software.

Sin embargo, como mencionan Maida y Pacienzia (2015), las metodologías de este conjunto tienden a ser más rigurosas, poco flexibles y pesadas, debido a que "centran su atención en llevar una documentación exhaustiva de todo el proyecto, la planificación y control del mismo, en especificaciones precisas de requisitos y modelado y en cumplir con un plan de trabajo". Los autores también indican que la fase inicial del proyecto es una de las más importantes, ya que se definen los "roles, actividades, artefactos, herramientas y notaciones".

2.2.2 Metodologías Ágiles

En los años 90, como consecuencia de la dificultad de la aplicación de las metodologías tradicionales, aparecen los marcos de referencia ágiles (o ligeros), como señala García (2015). El autor también expone que esta metodología trata de mejorar la estimación en costes, duración y alcance, de forma que pueda ser adaptable a los cambios y se vuelva una metodología más eficiente, ya que se produce menos documentación.

Según Molina et al. (2018) y García (2015), las metodologías ágiles se adaptan mejor a los cambios gracias a su nuevo enfoque. Ya no se trata de una visión orientada a los procesos, sino a las personas, haciendo que el marco de referencia tenga una comunicación constante con el cliente.

2.2.3 Comparación de las metodologías

En la determinación de la modalidad de metodología de software que se usó para el desarrollo del proyecto, se tomaron como base las comparativas de García (2015), Maida y Pacienzia (2015) y Molina et al. (2018). Se consideraron los aspectos más importantes que cada uno de los autores resalta y se utilizaron aquellos que fueran pertinentes para el presente trabajo.

Dentro de estos criterios se encuentra, en orden alfabético, la importancia de la arquitectura de software[1], la cantidad de artefactos[2], la adaptabilidad a los cambios, el modelo de ciclo de vida en el que se basa, la modalidad de los contratos, el tipo de cultura organizativa (jerarquía), el enfoque del desarrollo de software, la cantidad de documentación, el número

[1] **Arquitectura de Software:** Es "un grupo de abstracciones y patrones que brindan un esquema de referencia útil para el desarrollo de software dentro de un sistema informático." (Sarasty España, 2015)
[2] **Artefactos:** "productos tangibles del proyecto que son producidos, modificados y usados por las actividades." (Maida y Pacienzia, 2015)

de entregas, el estilo de desarrollo, la orientación de la gestión del proyecto, las metas de los proyectos siguiendo cierta modalidad, el modelo que se emplea, la forma de organización del equipo, la relación con el cliente, las características de los requerimientos, la cantidad de roles en el equipo de desarrollo, y tamaño del equipo de desarrollo.

Aspecto	Metodologías Tradicionales	Metodologías Ágiles
Arquitectura de Software	La arquitectura es esencial	No tan enfocado a la arquitectura
Artefactos	Mayor cantidad de artefactos	Menor cantidad de artefactos
Cambios	Resistencia a los cambios	Facilidad para realizar cambios
Ciclo de Vida	Secuencial (cascada, espiral)	Iterativo o evolutivo
Contrato	Contrato prefijado	Contrato flexible
Cultura Organizativa	Jerarquía fuertemente establecida	Menos jerárquica
Desarrollo de Software	Solución universal y segura	Solución según las necesidades del proyecto
Documentación	Extensa, detallada y con conocimiento específico	Reducida, ligera y con conocimiento tácito
Entregas	Una sola entrega (al finalizar)	Varias entregas constantes de los avances en el software
Estilo de Desarrollo	Predictivo (anticipativo)	Adaptativo
Gestión	Orientado a procesos	Orientado al cliente
Metas	Optimización	Adaptabilidad
Modelo	Pesado, rígido y sobredimensionado	Flexible: Se hace lo que se necesita
Organización del Equipo	Equipos preestablecidos	Equipos autoorganizados
Organización del Proyecto	Un solo proyecto completo	El proyecto está dividido en varios subproyectos
Relación con el cliente	Poca relación con el cliente	Mucha relación con el cliente (es parte del equipo de desarrollo)
Requerimientos	Conocidos, estables y predefinidos	Desconocidos, cambiantes y definidos durante el proyecto

Aspecto	Metodologías Tradicionales	Metodologías Ágiles
Roles	Muchos roles	Pocos roles
Tamaño del Equipo	Grupos grandes	Grupos pequeños y medianos (menos de 10 personas)

Tabla 2 Silverio, V. (2021) Cuadro comparativo entre las metodologías tradicionales y las metodologías ágiles [Tabla].

En la Tabla 2, se presenta el cuadro comparativo entre las metodologías tradicionales y las metodologías ágiles, con la respectiva descripción de lo que implica cada aspecto descrito anteriormente, construido a partir de la información recopilada de los autores mencionados anteriormente. Esta comparación sirve como sustento para la selección fundamentada de la modalidad de metodología que se utilizó en el proceso de construcción del software.

2.2.4 Aplicación al Proyecto

Una vez ya se han definido los aspectos de comparación entre las modalidades de las metodologías de desarrollo de software, y con las diferencias clarificadas en base a cada uno de estos criterios, es preciso adaptar el análisis hacia el proyecto a desarrollar. Para esto, se utilizaron los mismos criterios de comparación de la Tabla 2, mostrando cuál es más pertinente para el desarrollo de la aplicación móvil. Cabe recalcar que se consideraron todos los aspectos igual de importantes y, por tanto, la ponderación de cada uno fue la misma. Así, la modalidad que tenía la mayor cantidad de características relevantes al proyecto fue la que determinó qué tipo de metodología se usó en el mismo.

Ahora, considérese la Tabla 3, que contiene la opción escogida de entre las dos modalidades de metodologías de desarrollo de software, por cada uno de los aspectos de comparación, con la respectiva justificación de dicha selección. Al final del cuadro, se encuentra el total de características preferidas por cada tipo de metodología.

Aspecto	Metodología Tradicional	Metodología Ágil	Explicación
Arquitectura de Software		X	Al tratarse de una aplicación móvil, la arquitectura de la aplicación no es el aspecto primordial en la construcción del software.
Artefactos		X	Los desarrollos de aplicaciones web no cuentan con grandes cantidades de artefactos. Los framework se encargan de reducir ese número.

Aspecto	Metodología Tradicional	Metodología Ágil	Explicación
Cambios	X		Una definición anticipada de los requerimientos y la poca relación con el cliente, supondría que el proyecto no estuviese sujeto a muchos cambios.
Ciclo de Vida	X		El proyecto podría adaptarse a un ciclo de vida secuencial, a causa de los pocos cambios y pocas entregas que se conceptualizaron en un primer momento.
Contrato		X	Es necesario un contrato flexible (o no tradicional), debido a que es un proyecto propuesto y no una contratación de un servicio.
Cultura Organizativa		X	Gracias a la baja cantidad de desarrolladores involucrados en el equipo, una organización poco jerárquica es más útil.
Desarrollo de Software		X	Al definir los requerimientos con anticipación, se puede generar una solución diseñada a las necesidades del proyecto, para optimizar tiempo y recursos que se gastarían en una solución universal.
Documentación		X	Un proyecto con documentación extendida requiere de más tiempo asignado. Con el tiempo limitado del presente trabajo, es más adecuada una documentación ligera.
Entregas	X		Debido a que el proyecto fue propuesto, y no una necesidad establecida, no habría mucho interés en varias entregas constantes.
Estilo de Desarrollo	X		Los requerimientos, al ser predefinidos, suponen un estilo de desarrollo predictivo.
Gestión	X		Una relación limitada con el cliente fomenta que la gestión del proyecto sea enfocada al proceso a ser desarrollado, en lugar de centrada en el cliente.
Metas		X	El paradigma móvil de programación tiene un enfoque hacia la adaptabilidad con varios dispositivos, como se hablará en el capítulo 3.

Aspecto	Metodología Tradicional	Metodología Ágil	Explicación
Modelo		X	Con un desarrollo enfocado a una solución específica al proyecto, es oportuno mantener un modelo flexible.
Organización del Equipo		X	Los equipos autoorganizados son ventajosos para equipos pequeños de trabajo, debido a que no hay muchas personas para una jerarquía establecida.
Organización del Proyecto	X		Se pensaron pocas entregas de software, y debido a que el trabajo no es muy grande, es útil mantener al software organizado en un solo proyecto completo.
Relación con el cliente	X		Tomando en cuenta que el proyecto fue un trabajo propuesto, y no una necesidad establecida, es posible que el cliente no esté muy sumergido en el proyecto.
Requerimientos	X		Dado a que el cliente no propuso el proyecto, se debieron establecer los requerimientos con anterioridad. No aportaría mucho al proyecto el definirlos durante la construcción de la aplicación.
Roles		X	A causa de que el equipo de desarrollo tuvo pocas personas involucradas, es necesario tener pocos roles a ser repartidos.
Tamaño del Equipo		X	El software por construir no es muy grande y no hay mucha necesidad de grupos grandes de más de 10 personas.
TOTAL	8	**11**	

Tabla 3 Silverio, V (2021) Selección de la metodología por cada aspecto de la comparación [Tabla].

Basándose en los resultados de la Tabla 3, se puede inferir que la opción óptima para el proyecto es hacer uso de las metodologías ágiles, por su flexibilidad, ligereza y recurso humano reducido. También se puede apreciar que la diferencia entre las dos modalidades no fue muy grande y, por tanto, se podría también hacer uso de una metodología tradicional. Cabe destacar que la principal razón por la cual se selecciona más veces una metodología ágil que una tradicional, es por limitantes en los recursos (como el tiempo o capital humano).

De modo que, si se contara con mayor holgura en cuanto a dichos medios, la diferencia entre los dos tipos de metodologías sería aún menor.

De la misma forma, es importante mencionar que, aplicar una metodología ágil no trae inconvenientes en los criterios de la Tabla 3 relacionados a la modalidad tradicional. El hecho de que el presente trabajo fue una solución propuesta, y no una necesidad identificada por la organización, sugiere que se defina con anterioridad los requisitos de la aplicación para que el cliente esté al tanto de lo que se irá a desarrollar, provocando que el proyecto tienda a ser reacio al cambio, existan pocas entregas o se tenga un estilo predictivo de desarrollo, características que empatan con las metodologías tradicionales. Sin embargo, no significa que el cliente vaya a tener una relación distante con la solución propuesta. Por el contrario, se puede motivar a sostener una comunicación activa con la organización, como lo recomienda la modalidad de metodologías ágiles, haciendo que:

- Los requerimientos se puedan definir durante el proyecto,
- Se dé la posibilidad a cambios en el software,
- Se trabaje con un estilo de desarrollo más adaptativo a dichos cambios,
- Se emplee un ciclo de vida evolutivo para un modelo adaptativo y, en consecuencia,
- Se organice el proyecto en subdivisiones con varias entregas de los avances.

De este modo, pese a que algunos criterios se apegaban más a una modalidad tradicional de desarrollo, el proyecto fácilmente se puede adaptar completamente a una metodología ágil sin ninguna desventaja. Así, por lo tanto, se concluye que el marco de referencia que se adapta mejor al presente proyecto es referente a una metodología ágil.

2.3 Criterios de Análisis y Comparación

Similarmente a la comparación entre las modalidades ágiles y las tradicionales, es preciso señalar los criterios que se emplearon para analizar cada una de las metodologías de desarrollo. De la misma forma, estas características también fueron aspectos de comparación para poder identificar el mejor marco de referencia en el contexto del presente proyecto.

Para poder determinar qué criterios entran en el análisis, se basó en una reflexión de la necesidad de sus características según la naturaleza de la aplicación, así como, en ciertos aspectos que se emplearon en la comparación entre metodologías ágiles y metodologías tradicionales. De entre todos los elementos, se tomaron aquellos que son susceptibles a comparación. Sin embargo, nótese que no todos los criterios cuentan con el mismo nivel de relevancia para el proyecto, por lo que se les ha asignado un valor diferenciador.

A continuación, se exponen las 9 perspectivas con las que se analizó a las metodologías estudiadas, explicando qué se entiende por cada una, por qué son aspectos importantes en el estudio, y qué ponderación tiene cada uno en la comparación. Cabe aclarar

que, para la representación del peso, se utilizarán valores desde 1 hasta 3; donde “1” simboliza poca relevancia para el proyecto, y “3” significa mucha relevancia.

1. **Aprendizaje:** Un criterio que se refiere a qué tan complejo es dominar la metodología de desarrollo. Es una característica relevante debido al tiempo limitado que se posee para el desarrollo de la aplicación y es de vital importancia poder contar con un marco de referencia fácil de aprender. Sin embargo, se conoce de antemano que el aprendizaje de la metodología será complejo por contar con un equipo de desarrollo limitado. Así, el aspecto tendrá una ponderación de 1 en la escala de relevancia.
2. **Cantidad de Documentación Oficial sobre la Metodología:** Este aspecto representa qué tanta documentación sobre la metodología existe como una guía en la cual basarse. Aquí podría entrar también una comunidad, a manera de foro, que aclara dudas. Es relevante puesto que es esencial poder apoyarse en expertos para la aplicación de la metodología. Así, se tendrá una ponderación de 3 en la escala.
3. **Cantidad de Personas en el Equipo:** Este criterio analiza cuántas personas es necesario tener en el equipo de trabajo, así como qué roles debe desenvolver cada miembro. Algunas metodologías se enfocan en grupos grandes, mientras que otras fueron creadas para equipos pequeños. El número reducido de personas involucradas en el equipo de desarrollo del presente trabajo requiere de una metodología enfocada a grupos reducidos. Por esta razón, se pondera con un valor de 2 en la escala.
4. **Complejidad del Proyecto:** Algunas metodologías se construyen para poder organizar proyectos grandes, mientras que otras se enfocan en proyectos pequeños. En el caso de la aplicación que se quiere construir, se considera un proyecto de mediano alcance, por lo que se requerirá una metodología que soporte dicho tamaño. De esta forma, se otorga una valoración de 2 en la escala.
5. **Documentación Necesaria:** Pese a que las metodologías ágiles utilizan menos documentación que las tradicionales, es importante generar ciertos documentos según lo recomiende el marco de referencia. Es preciso escoger una metodología que no exija mucha documentación (por el limitado tiempo de desarrollo), pero que logre cumplir con el enfoque propuesto. Para analizar esta sección, se estudia toda la documentación necesaria por cada fase que esté definida en la metodología. Por lo cual, se asigna una ponderación de 2 en la escala.
6. **Enfoque al Paradigma Móvil:** Esta característica se refiere a qué tan enfocada está la metodología hacia la construcción de aplicaciones móviles. Dado a que se construirá un software basado en este paradigma, es fundamental que la metodología proporcione una guía para este tipo de aplicaciones. De esta manera, se califica con una relevancia de 3 en la escala.

7. **Enfoque General:** Este criterio trata sobre la idea global en la que cierta metodología se va a centrar. Es importante que el propósito general del proyecto se alinee con aspecto principal de la metodología, por lo que se considera un valor de 3 en la escala de relevancia.
8. **Requerimientos:** Este aspecto se enfoca en cómo cada metodología maneja la ingeniería de requerimientos, y qué tanta importancia le da. Dado que se pretende trabajar muy de cerca con el cliente, es importante que la metodología presente una guía dedicada a los requerimientos. De esta manera, también se otorga un valor de 3 en la escala de relevancia.
9. **Tiempo de Desarrollo Promedio:** Finalmente, el aspecto de tiempo de desarrollo se enfoca en explicar cuánto se suele tardar un equipo siguiendo cierta metodología. Es importante dado al limitado recurso del tiempo para desarrollar la aplicación. Por esta razón, se asigna un valor de 3 en la escala. Para el análisis de este criterio, se hace una estimación muy general (3 iteraciones), como un valor promedio de lo que tomaría un software de similar tamaño al presente proyecto.

2.4 Análisis Individual

Una vez identificados los criterios de estudio, se puede empezar con el análisis individual de las metodologías. Antes que nada, se adjunta una breve descripción del marco de referencia en observación, su origen y una exploración de las fases definidas. Por tal motivo, se adjunta la Tabla 4 que resume la información de las metodologías estudiadas. El propósito de este análisis es entender la idea principal de cada metodología antes de mostrar la comparación entre ellas, donde se indica cuál es la guía que se usó en el resto del proyecto.

Para este estudio, por lo tanto, se han seleccionado tres metodologías ágiles de desarrollo de software, siendo esta una cantidad propicia para el análisis: no es muy pequeña como para sesgar los datos, ni muy grande como para perder el objetivo del trabajo.

Primeramente, se escogió una metodología enfocada a la construcción de aplicaciones móviles: **Mobile-D**. Posteriormente, las otras metodologías corresponderían a las dos más empleadas actualmente, ya que su amplio uso puede ser aprovechado en el proyecto, a razón de que pueden cumplir con muchas de las características de análisis que se establecieron en el apartado anterior. En este sentido, tanto Rosselló (2015) como Zumba y León (2018) coinciden en que **XP** (“eXtreme Programming”) y **Scrum** son las dos metodologías que entran en esta categoría de ser las más usadas.

2.4.1 Scrum

2.4.1.1 Descripción y origen

Los autores originales de Scrum, Schwaber y Sutherland (2020), en su guía actualizada sobre la metodología, definen a Scrum como "un marco de trabajo liviano que ayuda a las personas, equipos y organizaciones a generar valor a través de soluciones adaptativas a problemas complejos". Los autores explican que se trata de un conjunto de procesos, técnicas y métodos que son intencionalmente incompletas, con el fin de que los equipos armen el marco de referencia conforme se requiera.

En los años 90, Ken Schwaber y Jeff Sutherland crean la metodología como una respuesta ante los problemas que traían las metodologías tradicionales en los proyectos de software, dándole un sentido más centrado hacia el usuario y la calidad del software (Anwer et al., 2017).

2.4.1.2 Fases

Scrum no establece formalmente un conjunto de fases para cumplir con la metodología. En su lugar, se definen actividades generales y una estructura de trabajo de cómo debería hacerse. Sin embargo, basados en esta última, varios autores intentan definir secciones para juntar las actividades que, para fin de este trabajo, se pueden considerar como fases.

Los autores Anwer et al. (2017), por ejemplo, establecen que las actividades se pueden dividir en tres grupos:

- **Pre-juego:** es la fase encargada de proporcionar la visión del proyecto, estimación de tiempo y de costes, y un diseño de la arquitectura del software; así como de la construcción del Product Backlog[3].
- **Juego:** es la etapa iterativa de desarrollo.
- **Post juego:** es la etapa de liberación del software[4], donde se realizan pruebas y se construyen manuales.

Una clasificación más amplia es la de SCRUMstudy (2013), el cuerpo de acreditación oficial para Scrum y certificaciones Ágiles, que indica que las actividades de Scrum se pueden definir en cinco grupos: iniciación (construcción del Product Backlog), planificación y estimación, implementación, revisión y retrospectiva, y liberación.

Bajo este concepto, se puede concluir que Scrum es una metodología iterativa que pasa por la planificación, diseño, programación, pruebas y liberación. Sin embargo, lo más

[3] **Product Backlog:** "Lista de objetivos/requisitos priorizada del producto" (Maida y Pacienzia, 2015)
[4] **Liberación del Software:** "Es la distribución de la versión final de una aplicación [de software]" (TechTarget Contributor, 2008)

importante por definir en esta sección es cómo se compone un Sprint, que representaría cada iteración que personifica estas fases antes mencionadas (Schwaber y Sutherland, 2020).

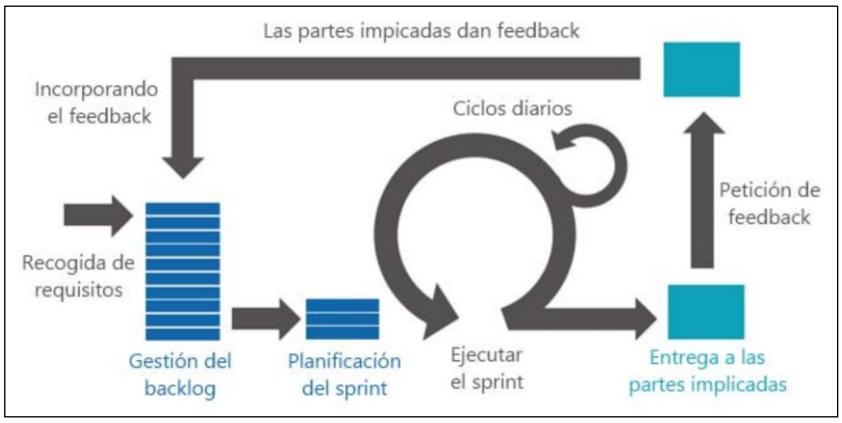


Ilustración 1 Maida, E. G., & Pacienzia, J. (2015). "Proceso de Scrum" [Figura]. Recuperado de https://repositorio.uca.edu.ar/handle/123456789/522

Como se puede apreciar en la Ilustración 1, un sprint es un proceso cíclico que se compone de seis secciones principales. Como parte del pre-juego, se debió establecer correctamente el Product Backlog que permite comenzar con la primera base de la "Planificación del Sprint" que, a su vez, contiene dos partes: la reunión con el cliente sobre qué requisitos se irán a desarrollar; y el diseño de todo el Sprint, es decir, cómo se irá a desarrollar los requisitos.

La segunda sección corresponde a la "Desarrollo del Sprint" que se enfoca en las diferentes iteraciones del desarrollo que culminan con la entrega completa de algún requisito definido en el Backlog. Esta sección, a su vez, contiene la tercera sub-fase: "Scrum Diario", que representan reuniones diarias de 10 a 15 minutos con todo el equipo de desarrollo, donde se discute el trabajo del día anterior, el progreso que se irá a hacer ese día, y la resolución de dudas respecto a cualquier inconveniente que haya tenido algún miembro. Esto se hace con el fin de poder mejorar la comunicación con el equipo y tener una idea clara del avance del proyecto hasta el momento (Anwer et al., 2017).

La cuarta sección corresponde a la "Revisión del Sprint". Es una reunión con el cliente en la cual se presentan los avances realizados para que este último pueda determinar si se han cumplido o no con los requisitos acordados. Con esta información, se prosigue a la quinta sección, la "Retrospectiva del Sprint", donde se analizan las causas del resultado que se obtuvo en la revisión del sprint, de manera que se pueda solucionar cualquier obstáculo en el equipo. (Maida y Pacienzia, 2015)

Finalmente, una vez reestructurado el equipo, y con la retroalimentación de parte del cliente, se puede llegar a la última sección que corresponde al "Refinamiento del Product

Backlog", cerrando el ciclo del Sprint. Aquí se añaden, eliminan, o re priorizan requisitos u objetivos definidos en el Backlog. Con esta información se puede volver a la sección primera en la nueva iteración hasta completar el proyecto.

2.4.1.3 Estudio de los criterios establecidos

Seguidamente, se encuentra el análisis de la metodología Scrum según los criterios definidos:

1. **Aprendizaje:** Requiere de un alto nivel de formación. Sangama (2020) indica que "no es una modalidad de gestión propia de grupos junior o que apenas estén en proceso de formación."
2. **Cantidad de Documentación Oficial sobre la Metodología:** Los autores mantienen actualizada constantemente la página para la divulgación de información sobre la metodología. De igual manera, existen capacitaciones y certificaciones.
3. **Cantidad de Personas en el Equipo:** Los equipos en Scrum tienen un tamaño de 5 a 10 personas, divididas en 3 diferentes roles: 1 Dueño del producto (representante del cliente), 1 Scrum Master (quien se asegura que el equipo cumpla con los lineamientos definidos) y de 3 a 9 desarrolladores. En ocasiones un solo miembro realiza el rol de Scrum Master y de desarrollador. Con más de 10 personas involucradas en el proyecto, la metodología no se aplica correctamente (Anwer et al., 2017).
4. **Complejidad del Proyecto:** Se puede aplicar Scrum a un proyecto con cualquier nivel de complejidad (Anwer et al., 2017).
5. **Documentación Necesaria:** Según Anwer et al. (2017), la documentación necesaria por fase es:
    a. **Pre-juego:** Product Backlog
    b. **Juego:**
        i. **Planificación del sprint:** Product Backlog con cualquier cambio decidido entre las dos partes.
        ii. **Desarrollo del Sprint:** No hay documentación establecida.
        iii. **Scrum Diario:** Son reuniones orales, por lo que no requieren documentación.
        iv. **Revisión del Sprint:** Es una reunión oral e informal, por lo que no requiere documentación.
        v. **Retrospectiva del Sprint:** No hay documentación establecida.
        vi. **Refinamiento del Product Backlog:** Nueva versión del Product Backlog.
    c. **Post Juego:** Manuales de usuario y materiales de capacitación.

6. **Enfoque al Paradigma Móvil:** Sangama (2020) indica que “varios autores proponen el empleo de Scrum para el desarrollo de software móvil.”, por lo que sí puede ser aplicado a este paradigma.
7. **Enfoque General:** Scrum se centra en los equipos, en su configuración y relación, tratando de que las interacciones entre los miembros sean claras (Schwaber y Sutherland, 2020).
8. **Requerimientos:** El Product Backlog es un planteamiento muy claro de los requerimientos. La metodología tiene lineamientos correctamente definidos de cómo trabajar con los requerimientos, y con el cliente para poder definirlos.
9. **Tiempo de Desarrollo Promedio:** 4 semanas por Sprint (Anwer et al., 2017). Para un proyecto estimado para 3 iteraciones, se esperaría una duración mínima de 12 semanas.

2.4.2 Programación Extrema (XP)

2.4.2.1 Descripción y origen

Como explican Anwer et al. (2017), el origen de la metodología de Programación Extrema se produce en 1996, cuando Kent Beck la desarrolla trabajando en Chrysler Comprehensive Compensation (C3). La metodología nace como un conjunto de buenas prácticas que se fueron perfeccionando poco a poco para mejorar la calidad del proyecto en el que estaban trabajando en C3. (Wells, Your host: Don Wells, 2009)

Según la fuente (Wells, Extreme Programming: A gentle introduction, 2013), la metodología XP es una variante ágil que se enfoca en el trabajo en equipo. El autor menciona que la metodología propone prácticas que mejoran los productos de software de compañías pequeñas o grandes, aumentando su probabilidad de éxito.

2.4.2.2 Fases

Las fases contempladas en XP son 6, como lo explican Maida y Pacienzia (2015). Primero, se encuentra la “Fase de Exploración”, que se centra en tres aspectos principales: la concepción de la idea de la aplicación de parte del usuario (por medio de la escritura de historias de usuario[5]), la familiarización del equipo con la herramienta de desarrollo (Spike arquitectónico[6]), y una metáfora del negocio[7].

---

[5] **Historias de Usuario:** Descripciones, a grandes rasgos, de las funcionalidades y requerimientos que el aplicativo debe resolver. (Maida y Pacienzia, 2015)

[6] **Spike arquitectónico:** Es un conjunto de actividades realizadas por el equipo de desarrollo como pruebas y familiarización con las tecnologías, reconocimiento de posibles arquitecturas y realización de prototipos (Maida y Pacienzia, 2015)

[7] **Metáfora de Negocio:** “Marco de trabajo con los objetos básicos [de la arquitectura] y sus interfaces.” (Anwer et al., 2017)

Segundo, está la “Fase de Planificación”. Esta etapa se encarga de construir una planificación flexible que define elementos como el tamaño del equipo, el horario, las horas de trabajo, entre otros. Es una fase importante pues organiza qué historias de usuario se resolverán en qué iteración, intentando mantener todos los ciclos del mismo tamaño. Esta fase tiene dos partes fundamentales: la planificación de lanzamiento (que establece las fechas de entrega de cada característica a implementar) y la planificación de iteraciones (que define el plan de trabajo de la iteración) (Anwer et al., 2017; Maida y Pacienzia, 2015).

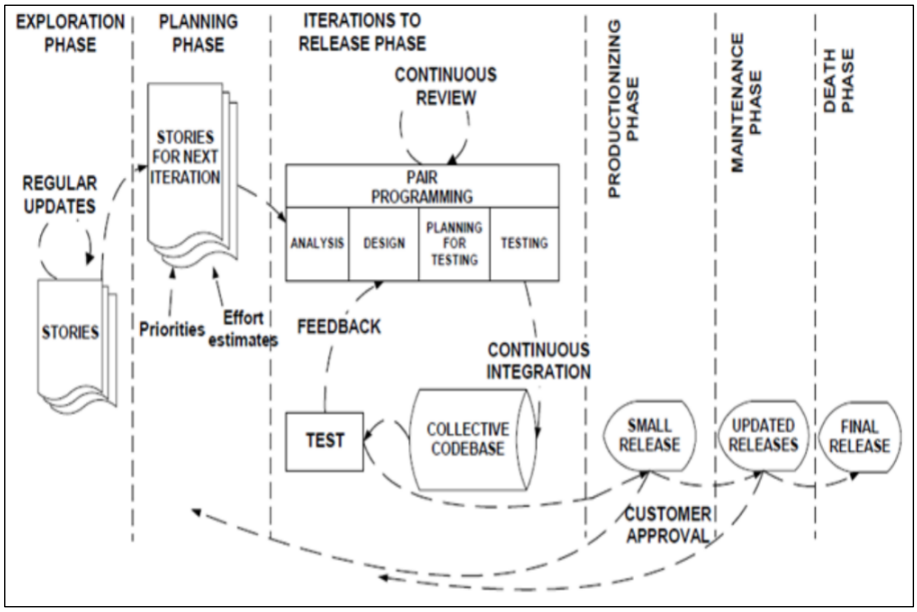


Ilustración 2 Anwer, F., Aftab, S., Shah, S. M., & Waheed, U. (2017). “Ciclo de Vida de Extreme Programming” [Figura] Recuperado de https://www.researchgate.net/profile/Shabib-Aftab-2/publication/316845761_Comparative_Analysis_of_Two_Popular_Agile_Process_Models_Extreme_Programming_and_S

Tercero, se contempla la “Fase de Iteraciones”, que representa el “diseño, codificación, pruebas e integración” (Anwer et al., 2017) de cada una de las iteraciones, repitiendo según el número de ciclos definidos previamente. Cada iteración puede tomar de una a cuatro semanas. En la Ilustración 2, se pueden apreciar todas las 6 fases de la metodología XP, donde se muestra el proceso repetitivo de la tercera fase.

En cuarto lugar, aparece la “Fase de Producción”. Cuando se han cumplido con las necesidades mínimas del proyecto, se puede poner al software en producción para que el usuario final ya pueda comenzar a utilizar la aplicación. Al mismo tiempo, se continúa con las otras iteraciones. Esta etapa también maneja entregas cíclicas, pero con un periodo más espaciado (Maida y Pacienzia, 2015).

Quinto, está la “Fase de Mantenimiento”. Esta etapa se incorpora una vez que se ha cumplido con todas las funcionalidades antes definidas. Se estudia el agregar nuevas

funcionalidades que fueron surgiendo a través del ciclo de vida del proyecto. Se debe tener cuidado con que las nuevas implementaciones no interfieran con el software que ya estaba en fase de producción.

Finalmente, la sexta etapa se refiere a la “Fase de Muerte del Proyecto”. Para llegar al final de la vida del software, existen dos posibilidades, como explican Anwer et al. (2017). La primera se refiere a cuando todas las funcionalidades (historias de usuario) han sido atendidas, hasta el punto en que el cliente ya esté satisfecho; o cuando la organización ya no puede pagar por las funcionalidades a implementar, por lo que se acuerda un cierre del proyecto. Al final del proyecto, se espera un archivo con información sobre el sistema.

2.4.2.3 Estudio de los criterios establecidos

A continuación, en base a los aspectos de análisis, se expone el estudio de la metodología XP:

1. **Aprendizaje:** Debido a todos los procesos definidos en la metodología, no es tan fácil aplicarla (Muñoz, 2020).
2. **Cantidad de Documentación Oficial sobre la Metodología:** La mayoría de la documentación oficial encontrada sobre XP corresponde a sus orígenes al principio de la década de los 2000.
3. **Cantidad de Personas en el Equipo:** La metodología está enfocada a grupos pequeños de menos de 10 personas. Sin embargo, una de las prácticas más importantes de XP, es la programación en parejas, dando como resultado un equipo necesario de 2 a 10 programadores (Maida y Pacienzia, 2015).
4. **Complejidad del Proyecto:** Se entiende que XP es una metodología más enfocada a pequeños y medianos proyectos (Anwer et al., 2017).
5. **Documentación Necesaria:** Según Maida y Pacienzia (2015), la documentación necesaria por fase es:
    a. **Fase de exploración:** Historias de usuario, Spike arquitectónico, metáfora del negocio.
    b. **Fase de planificación:** Plan de lanzamiento, plan de iteraciones.
    c. **Fase de iteraciones:** No hay documentación establecida.
    d. **Fase de producción:** Documentación del Sistema, de Operaciones, de Soporte, y de Usuario. (Ambler, 2002)
    e. **Fase de mantenimiento:** No hay documentación establecida.
    f. **Fase de muerte del proyecto:** No hay documentación establecida.
6. **Enfoque al Paradigma Móvil:** XP puede atender las exigencias del paradigma móvil (Sangama, 2020).

7. **Enfoque General:** Se enfoca en aspectos ingenieriles y de gestión de proyectos. (Anwer et al., 2017; Muñoz, 2020).
8. **Requerimientos:** XP tiene lineamientos muy bien descritos de cómo obtener los requerimientos del usuario con las Historias de Usuario.
9. **Tiempo de Desarrollo Promedio:** de 1 a 3 semanas por iteración (Anwer et al., 2017). Para un proyecto de 3 iteraciones, se tendría una duración aproximada de 3 a 9 semanas.

2.4.3 Mobile-D

2.4.3.1 Descripción y origen

Según Muñoz (2020), Mobile-D es una metodología ágil dedicada al desarrollo móvil estrictamente. Está pensado para equipos con pocos recursos (económicos y de personal). Se enfoca en la interacción con el cliente, así como en la rápida adaptación a los cambios.

Los creadores de Mobile-D, Abrahamsson et al. (2004), en el artículo original sobre el marco de referencia explica la forma en la que el nacimiento de la metodología se produce como una respuesta a las dificultades que representaba la nueva tecnología móvil en ese tiempo:

> Para superar los desafíos involucrados en el desarrollo de aplicaciones móviles, hemos desarrollado un acercamiento ágil de desarrollo llamado el Mobile-D. El acercamiento está basado en Extreme Programming (prácticas de desarrollo), metodologías Crystal (escalabilidad del método) y el Proceso Unificado Racional (cobertura del ciclo de vida).

2.4.3.2 Fases

Originalmente, los autores Abrahamsson et al. (2004) definieron 5 fases para la metodología Mobile-D, dando una primera directiva de en lo que se defería centrar el marco de referencia: configuración, núcleo, núcleo2, estabilización y envoltura. Con el tiempo, estas fases cambiaron de nombre a fin de mejorar el entendimiento del enfoque, siendo: exploración, inicialización, producción, estabilización y pruebas del sistema. Cada una de estas fases tiene ciertas actividades principales que las diferencian de las otras, como se puede apreciar en la Ilustración 3. A continuación, con base en Muñoz (2020), se presenta la información respectiva de cada fase.

La fase de “Exploración” se centra en el establecimiento de los grupos de interés, definición del alcance y objetivos, y configuración de los equipos de desarrollo. Después, la fase de “Iniciación” genera los diseños de arquitecturas y procesos con diagramas de casos de uso, pantallas de interfaz. Tercero, se encuentra la fase de “Producción”, donde se aplican los conceptos de día de planificación, día de trabajo y día de liberación, que representan el

corazón de la metodología, ya que es donde se implementan las funcionalidades planificadas y se va realizando las diferentes entregas del software.

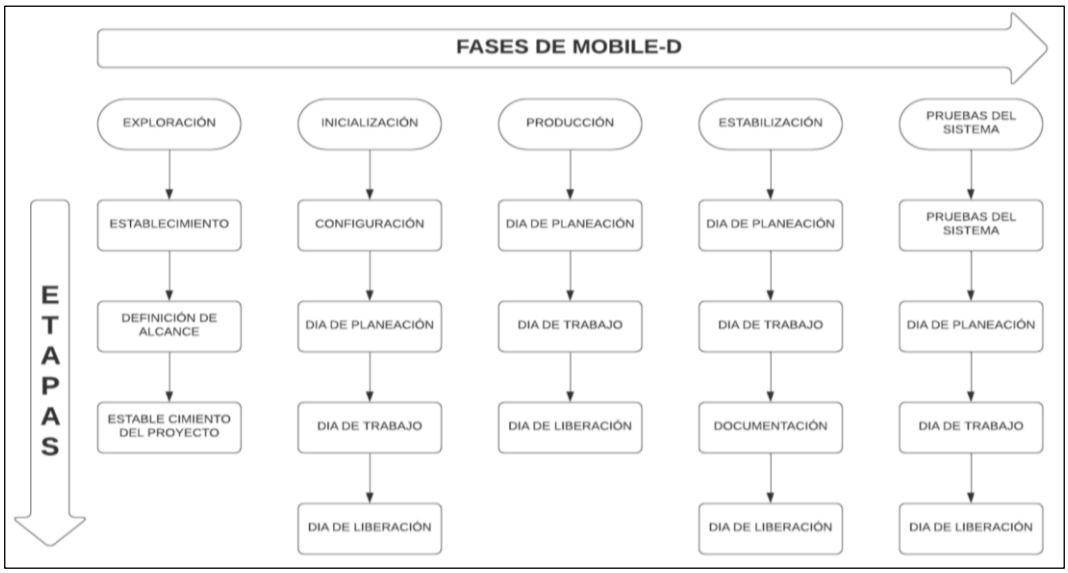


Ilustración 3 Muñoz, C. (2020) "Ciclo de Desarrollo de MOBILE-D" [Figura] Recuperado de: http://dspace.unach.edu.ec/handle/51000/7073

Luego, se encuentra la fase de "Estabilización", que se trata del conjunto de tareas para poder integrar todo el sistema y asegurarse que funcione correctamente. Finalmente, Está la fase de "Pruebas del Sistema", donde se prueba según lo requiera el cliente y se corrigen los errores encontrados.

Esta metodología tiene una organización más secuencial que el resto de las metodologías ágiles estudiadas. Sin embargo, como explican Corral et al. (2015), cada fase del marco de referencia incita a la utilización de iteraciones. Sin embargo, éstas dependen del desarrollador.

2.4.3.3 Estudio de los criterios establecidos

En base a los criterios de estudio, se presenta, seguidamente, el análisis correspondiente a la metodología Mobile-D:

1. **Aprendizaje:** Al no estar aún tan definida, no proporciona un marco de referencia importante para el proyecto (Sangama, 2020).
2. **Cantidad de Documentación Oficial sobre la Metodología:** No es muy fácil encontrar información de canales oficiales respecto a la metodología.
3. **Cantidad de Personas en el Equipo:** Los autores Abrahamsson et al. (2004) indican que Mobile-D originalmente fue pensado para menos de 10 programadores, sin un número mínimo.

4. **Complejidad del Proyecto:** Es una metodología pensada para proyectos menores a 10 semanas (proyecto mediano o pequeño) (Muñoz, 2020).
5. **Documentación Necesaria:** según Muñoz (2020), la documentación necesaria por fase es:
    a. **Exploración:** "Los requisitos iniciales, plan del proyecto, descripción de los procesos, plan de medida, plan de capacitación."
    b. **Inicialización:** "El plan actual del proyecto, la versión de la arquitectura del software y la descripción del diseño, requisitos iniciales modificados, interfaces de usuario, diagramas de casos de uso".
    c. **Producción:** "Funcionalidades puestas en funcionamiento, anotaciones del desarrollo, esquemas de la interfaz de usuario de la aplicación, historias de usuario, requisitos modificados."
    d. **Estabilización:** Documentación del producto terminado.
    e. **Pruebas del sistema:** No hay documentación establecida.
6. **Enfoque al Paradigma Móvil:** Nació para solventar los inconvenientes de las metodologías en el paradigma móvil (Abrahamsson et al., 2004).
7. **Enfoque General:** Se enfoca en el desarrollo de la aplicación con relación al cliente (Abrahamsson et al., 2004; Muñoz, 2020).
8. **Requerimientos:** Mobile-D tiene una fase específica para definir los requisitos. Sin embargo, no le da mucha importancia, dado que no define claramente cómo extraerlos del cliente.
9. **Tiempo de Desarrollo Promedio:** La metodología fue concebida para proyectos cortos, con una duración menor a 10 semanas. (Abrahamsson et al., 2004).

2.4.4 Resumen en Base a los Criterios

A partir de los criterios analizados en el apartado anterior por cada una de las metodologías, se construyó la Tabla 4, con el fin de resumir y hacer más comprensiva la información, para facilitar el análisis sobre qué metodología se empleó en el resto del proyecto.

Criterio	Scrum	XP	Mobile-D
Aprendizaje	Muy alto	Muy alto	Medio
Cantidad de Documentación Oficial sobre la Metodología	Información oficial actualizada	Información oficial desactualizada	Información no oficial, no actualizada

Criterio	Scrum	XP	Mobile-D
Cantidad de Personas en el Equipo	De 5 a 10 miembros	Desde 2 a 10 desarrolladores	Menos de 10 programadores
Complejidad del Proyecto	Cualquiera	Pequeños y medianos	Pequeños
Documentación Necesaria	Sí	Sí	Sí, pero no tan definida
Enfoque al Paradigma Móvil	Sí	Sí	Sí
Enfoque General	Trabajo en equipo	Aspectos ingenieriles y de gestión de proyectos	Aplicación y cliente
Requerimientos	Sí	Sí	Sí, pero no muy enfocado
Tiempo de Desarrollo Promedio	12 semanas	De 3 a 9 semanas	Menos de 10 semanas

Tabla 4 Silverio, V. (2021) Resumen de los criterios establecidos según la información de las tres metodologías de desarrollo de software [Tabla]

2.5 Comparación y Análisis

Con el análisis individual, basado en los criterios y ponderaciones establecidas en la sección 2.3, se pudo realizar la respectiva comparación entre las tres metodologías: Scrum, XP y Mobile-D. En este aspecto, también se establece una calificación por cada criterio, que varía entre 1 y 3, siendo “1” cuando la metodología no tiene características que se alineen con los objetivos del proyecto, y “3” cuando la metodología resultaría ser muy beneficiosa en el criterio estudiado. En la Tabla 5, se puede apreciar las diferentes calificaciones para los tres marcos de referencia, así como la respectiva explicación de por qué se otorgaron los valores.

Criterio	Ponderación (1-3)	Calificación (1-3)			Explicación
		Scrum	XP	Mobile-D	
Aprendizaje	1	1	1	2	El proyecto necesita una metodología que sea fácil de implementar, pero que mantenga una guía completa de cómo

Criterio	Ponderación (1-3)	Calificación (1-3)			Explicación
		Scrum	XP	Mobile-D	
					realizar el proceso de Ingeniería de Software.
Cantidad de Documentación Oficial sobre la Metodología	3	3	2	1	Para el éxito de un proyecto, se necesita tener documentación actualizada y relevante.
Cantidad de Personas en el Equipo	2	1	2	3	Debido a que el equipo estuvo formado por un solo desarrollador, equipos grandes y medianos no son convenientes para el proyecto.
Complejidad del Proyecto	2	3	3	3	El software tiene un tamaño mediano, soportado por cualquiera de las tres metodologías.
Documentación Necesaria	2	3	3	2	Para el desenvolvimiento del proyecto, es necesaria una documentación que no sea muy extensa, pero que sea lo suficientemente definida para otorgar una guía correcta en todas las fases de la metodología.
Enfoque al Paradigma Móvil	3	3	2	3	Gracias a que el producto del trabajo es una aplicación móvil, se espera un marco de referencia que contenga guías para dicho paradigma.
Enfoque General	3	1	2	3	Debido a los presupuestos limitados del proyecto, el enfoque se debe centrar en la aplicación

Criterio	Ponderación (1-3)	Calificación (1-3)			Explicación
		Scrum	XP	Mobile-D	
					y agilidad, más que en el equipo de desarrollo.
Requerimientos	3	3	3	1	Es necesario que las metodologías se enfoquen mucho en la ingeniería de requerimientos.
Tiempo de Desarrollo Promedio	3	2	3	3	El desarrollo del proyecto cuenta con una duración estimada de 14 semanas.
TOTAL		51	**53**	51	

Tabla 5 Silverio, V. (2021) Comparación de las tres metodologías de desarrollo de software según los criterios establecidos [Tabla]

En función de los valores de la Tabla 5, se puede apreciar que el marco de referencia que mejor se alinea con el trabajo, según los criterios estudiados, es la metodología XP. Con esta última, el proyecto tiene un desarrollo rápido, una buena gestión de requerimientos, y una documentación definida para representar un modelo en el cual la aplicación se puede guiar.

El principal inconveniente de la metodología Mobile-D, es la poca madurez que tiene en comparación con los otros dos marcos de referencia. Mobile-D no tiene documentación oficial que pueda convertirse en una pauta clara de cómo desarrollar la aplicación. Pese a tener el enfoque y la simplicidad orientados a los fines del presente proyecto, no define explícitamente qué documentos se necesitan presentar por cada fase, o cómo extraer los requerimientos del usuario.

En el caso de Scrum, la desventaja notable que se existe es respecto al trabajo centrado en equipos medianos (de 5 a 10 personas), y el enfoque general basado en la relación entre sus miembros. En el caso del presente trabajo, donde existió un único desarrollador, no se podría sacar todo el provecho a la metodología.

Esto representa un obstáculo para las tres metodologías estudiadas, y para otras tantas no contenidas en este análisis. Como indican Mogollón y Esteban (2010), es común para todas las metodologías de desarrollo el promover el trabajo en equipo para obtener un producto de software. Para los autores, es mucho más práctico poder adaptar una metodología existente que generar una nueva para ser de uso individual.

En el presente trabajo, eXtreme Programming, pese a ser la metodología escogida para ser la base del desarrollo, tuvo que ser modificada en criterios como trabajo en equipo.

Se adoptaron las prácticas en un nivel simplificado, para poder solventar la falta de guía en desarrollos individuales. Sin embargo, procedimientos como la ingeniería de requerimientos fueron muy útiles para el objetivo del proyecto.

2.6 Conclusión

Se puede concluir que las metodologías de desarrollo de software representan un marco de referencia para poder guiar el proceso de ingeniería de software a lo largo de todo el proyecto. Es fundamental escoger una metodología que se adapte a los objetivos del proyecto, para poder sacar provecho de todo el conjunto de prácticas. Por esta razón, lo primero es seleccionar correctamente entre un modelo ágil o tradicional.

En este aspecto, se analizaron criterios como la relación con el usuario, la cantidad de documentación, y la gestión de requerimientos y de cambios. Se demostró que para el presente proyecto no existe mucha diferencia entre las dos modalidades, pero que una metodología ágil puede aportar muchas ventajas claves al trabajo.

Posteriormente, se consideraron tres metodologías: Scrum, XP y Mobile-D, y se analizaron las características como la documentación oficial, el enfoque general y la cantidad de personas necesarias en el equipo. El objetivo era determinar cuál marco de referencia significaría una mayor guía para el proyecto. Sin embargo, una dificultad para cualquier metodología de desarrollo es la falta de prácticas para un desenvolvimiento individual. Por esta razón, se estableció que el mejor camino a tomar, en el caso de este proyecto, es la modificación de una metodología actual. Así, finalmente, se indica el por qué se usó a XP como marco de referencia adaptando ciertas prácticas para poder ser aplicadas a usa sola persona.

3 CAPÍTULO 2: EXPLORACIÓN Y PLANIFICACIÓN

Tal y como explica Ambler (2002) en su estudio paso a paso de las fases de XP, las etapas de Exploración y Planificación son las partes más importantes del proyecto y deben ser analizadas en los inicios de la aplicación. El presente capítulo aborda el estudio de ambas fases de manera que se definan las directrices que darán forma al desarrollo.

La primera fase se enfoca en la comunicación con el cliente para determinar los requisitos del sistema en un formato específico llamado “historias de usuario”. La segunda fase toma cada historia y define las tareas necesarias para poder satisfacer cada requerimiento.

3.1 Fase de Exploración

La Fase de Exploración es la primera etapa en el ciclo de desarrollo de XP. Su objetivo principal es sondear ciertos elementos importantes para la implementación del software y darles un correcto tratamiento antes de acoplarlos completamente al proceso de construcción de la aplicación. Cabe destacar que las fases de XP, al ser una metodología ágil, se enfocan en el dinamismo y flexibilidad, dando como resultado una guía poco estricta de cómo comenzar el desarrollo de software.

En esta etapa, se abordan tres aspectos primordiales: las Historias de Usuario (de las que se obtienen los requerimientos), el Spike Arquitectónico (un primer vistazo a la arquitectura de la aplicación, así como de la familiarización con las herramientas), y la Metáfora del Negocio (un concepto para simplificar el entendimiento de la aplicación). Se muestran, por lo tanto, cada uno de estos elementos aplicados al presente proyecto.

3.1.1 Historias de usuario

Una historia de usuario es la unidad básica en la determinación de necesidades dentro de la metodología XP. Es un concepto usado a lo largo de las metodologías ágiles debido a su cambio de paradigma y enfoque al trabajar con el cliente. No se trata de una definición de requerimientos tradicional, pues no contiene ninguna forma de leguaje técnico, sino es una descripción ligera de lo que el usuario necesita que el sistema haga por ellos, con el único fin de poder proporcionar una estimación de bajo riesgo (Wells, User Stories, 2001).

El punto de vista principal de las historias de usuario es diferente a como se concebían elementos de la documentación tradicional. Una de las principales diferencias radica en el hecho de que es el cliente quien las escribe, sin ninguna descripción de la interfaz o profundidad de cómo debería comportarse el software.

Para Roden (2002), se refiere a una narrativa corta que describe la situación en la que el consumidor se encuentra. Está compuesta de tres partes fundamentales: un quién, un qué

y un por qué. La autora explica que de las descripciones de las historias de usuario se derivan las tareas de usuario que, a su vez, contienen una definición técnica de cómo implementar el proyecto.

Sin embargo, nótese que, pese a ser el primer acercamiento al proyecto de software, no es necesario establecer todas las historias en la primera fase. El hecho de que XP esté basada en un ciclo de vida incremental permite que se puedan definir más historias de usuario en otras etapas de la metodología. Inclusive, para llegar a la última fase del marco de referencia, la etapa de muerte del proyecto es necesario haber cumplido con todas las historias de usuario que hayan surgido a lo largo del proceso de construcción.

3.1.1.1 Características de una historia de usuario

Según Agile Alliance (2019) y Bowes (2014) las historias de usuario deben cumplir con ciertas propiedades para un aseguramiento de calidad y para poder sacar todo el provecho de la metodología de software. Si no se llegara a satisfacer las cualidades, es recomendado que la historia de usuario sea reformulada, ya que puede causar problemas en el desarrollo de las siguientes fases.

En primer lugar, los autores destacan que se debe cumplir con el acrónimo INVEST que, por sus siglas, se refiere a que la historia de usuario debe ser:

- **Independent (independiente):** No depende de otras historias de usuario.
- **Negotiable (negociable):** Al no ser descrita con un nivel alto de detalle, su interpretación puede alterarse durante el proceso de construcción del software.
- **Valuable (valiosa):** Aporta valor al usuario o consumidor.
- **Estimable:** Se puede asignar una buena aproximación al tiempo real de implementación.
- **Small (pequeña):** Debe tener un tamaño óptimo para poder ser implementado en una sola iteración.
- **Testable (comprobable):** Se debe poder definir y aplicar pruebas a la funcionalidad discutida.

De esta manera, se puede garantizar que la historia sea útil para el resto del proyecto cumpliendo con los estándares de calidad de la metodología de desarrollo.

Igualmente, se tienen las características de las 3 C's. Como explica Mistry (2017), la primera C se refiere a "Card" (carta o tarjeta). Aquí se recomienda que se traduzca las historias de usuario a un medio físico, donde se agregue la descripción en una de las caras. Esto permite la familiarización del equipo con características secundarias de las historias de usuario: se mantiene la cualidad de atomicidad de una historia, se puede manipular

físicamente (asignar responsable o romperse), permite adjuntar notas o elementos en otras fases, y facilita la conceptualización de la cualidad tangible (alcanzable) de las historias.

La segunda C es para “Conversation” (conversación). Es una característica que explica que el origen de las historias de usuario debe ser a partir de una conversación entre las diferentes partes interesadas del proyecto.

La última C significa “Confirmation” (confirmación). Esta cualidad explica que debe existir un criterio de aceptación para la historia de usuario que indique si se ha cumplido con el objetivo de esta.

3.1.1.2 Elementos de una historia de usuario

Mistry (2017) explica que una historia de usuario correctamente formulada, que cumpla con las características antes descritas, debería tener los siguientes componentes:

- **Identificador:** Puede ser usado para diferenciar correctamente a las historias de usuario con una notación propia de la empresa o del equipo de trabajo, sin tener que depender del título de la historia. En el caso del proyecto, se utilizó una numeración combinada para identificar las diferentes historias de usuario. La estructura empleada es: “Iteración. Orden dentro de la iteración – Orden General de la historia”. Por ejemplo, la tarjeta que se desarrolló en cuarto lugar dentro la segunda iteración, pero que representa la decimoprimera historia en referencia a todas las otras, tiene el identificador: “2.4 – 011”.
- **Título:** Puede ser útil para reconocer la historia de usuario de una manera más fácil sin necesariamente tener que leer todo el contenido de la tarjeta. Para la construcción del título, se pensó en pocas palabras que resumirían toda la historia de usuario.
- **Prioridad:** Es preciso señalar la prioridad de cada historia de usuario con el fin de ordenarlas y así poder determinar qué requerimientos se deberían implementar primero. En el caso del proyecto, con ayuda del cliente, se asignó una prioridad de entre 1 y 3 a cada historia de usuario, donde “1” representa una historia con baja prioridad y “3” representa la más alta prioridad. Sin embargo, la decisión final del orden de las historias, pese a ser influenciado por las prioridades, depende del equipo de desarrollo (Bowes, 2014). Posteriormente, se tradujo la prioridad de los números a etiquetas: alta, media y baja, para mejorar el entendimiento.
- **Estimación:** Es la duración aproximada de la implementación de una historia. Wells (2001) explica que la estimación a considerar es concebida como si se tratase de un tiempo normal de trabajo sin distracciones y sabiendo exactamente lo que hay que hacer. El mismo autor también indica que el desarrollador encargado de implementar la historia de usuario es el único miembro permitido para determinar la estimación por

cada historia, basándose en las conversaciones con el cliente y toda la información obtenida de ellas. En el caso del proyecto, la unidad de tiempo utilizada son los días, dado que son más precisos para determinar la estimación que las semanas.

- **Formato (oración):** Anteriormente se expuso que una historia de usuario era una narrativa corta que contenía un quién, un qué y un por qué. Bowes (2014) propone que la oración que acople estas tres partes cumpla con un formato específico: “Como un <Usuario>, quiero una <Característica> para un <Beneficio>”. Escribir la proposición de esa manera puede ayudar al equipo de trabajo a entender mejor las historias de usuario, facilitando la comprobación de si se ha cumplido con el acrónimo INVEST. Igualmente, permite tener presente la perspectiva del consumidor en todo momento.
- **Criterios de Aceptación:** También llamados Criterios de Completitud, son aquellas condiciones que se deben cumplir para considerar que se ha terminado la implementación de una historia de usuario. Estos criterios dictarán las pruebas de aceptación que se realizarán en el paso de la fase de iteraciones a la fase de producción (Ambler, 2002). Al igual que con la oración que describe la historia de usuario, se propone que los Criterios de Aceptación sigan un formato característico, comúnmente llamado “Given-When-Then” (Dado-Cuando-Entonces). Según Agile Alliance (2017), la plantilla debería seguir la forma: “Dado un <Contexto>, cuando una <Acción se lleva a cabo>, entonces un <Conjunto particular de consecuencias observables se debería obtener>”.

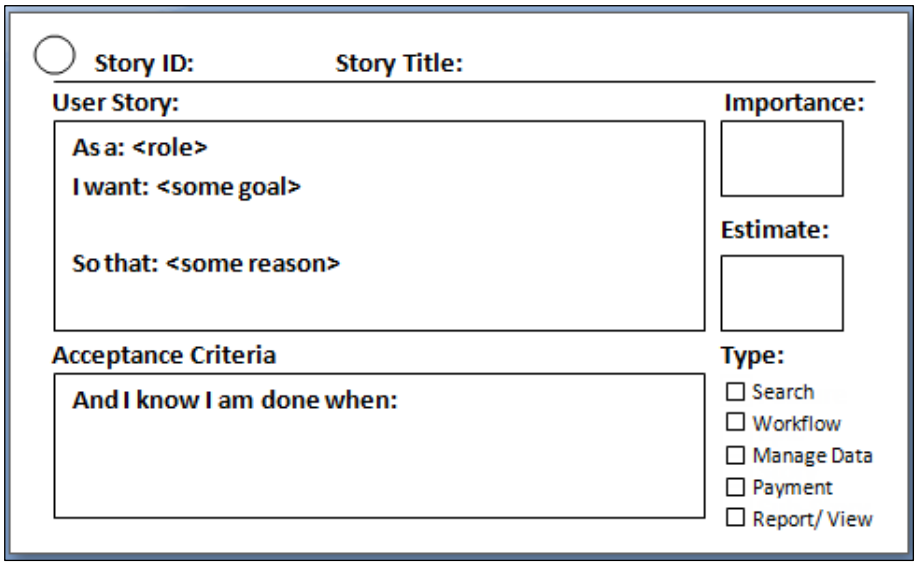


Ilustración 4 Mistry, A. (2017) Plantilla de una historia de usuario [Figura]. Recuperado de https://www.c-sharpcorner.com/article/what-is-user-story-in-agile-scrum/

La Ilustración 4 representa un ejemplo de plantilla de una historia de usuario con todos los elementos discutidos, mostrando una sección designada para cada componente: identificador, título, descripción, estimación, prioridad y criterios de aceptación. Esta plantilla se usó como base para la construcción del formato que se ha empleado en las historias de usuario del presente proyecto.

Cabe resaltar que la fase de exploración no contempla todos los elementos de las historias de usuario, dado que el objetivo de la fase es únicamente sondear los requerimientos y transformarlos en una descripción fácil de entender (historias de usuario). Elementos tales como la prioridad, estimación y criterios de aceptación se realizarán en la fase de planificación.

3.1.1.3 Aplicación al proyecto

Una vez definidas correctamente las características de las historias de usuario, es pertinente indicar las historias a las que se ha llegado juntamente con el cliente. Cabe destacar que se sigue el formato establecido en el apartado anterior, de modo que se puedan contestar las tres preguntas (quién, qué y para qué) en una sola oración.

A continuación, por lo tanto, se adjuntan las historias de usuario sin ningún orden específico, que se obtuvieron de conversaciones con el usuario y que, posteriormente, fueron revisadas y aceptadas. Pese a que la metodología de desarrollo sugiere que sea el cliente quien escriba las historias de usuario, hay que tener siempre en cuenta que este proyecto no fue una aplicación propuesta por el cliente y, por consiguiente, el equipo de desarrollo tiene que compensar ciertas tareas del consumidor.

1. **Como** organizador del festival, **quiero** medios **para** hacer llegar información a cada director o corista.
2. **Como** parte del público del festival, **quiero** un repositorio de información sobre los coros **para** consultar datos relevantes de manera centralizada.
3. **Como** interesado en los eventos, **quiero** una sección de contactos **para** comunicarme con el festival.
4. **Como** asistente del festival, **quiero** un programa de mano **para** tener la información centralizada de los eventos.
5. **Como** organizador del festival, **quiero** un control sobre suscripciones a un sistema PPV[1] **para** poder brindar transmisiones de los eventos.
6. **Como** una persona interesada en el festival, **quiero** un cronograma de eventos **para** visualizar la información.

[1] **Pago Por Visión (PPV):** “Un servicio de televisión en el cual los espectadores requieren pagar una tarifa con el fin de ver un programa específico”. (Lexico, 2019)

7. **Como** promotor de cultura, **quiero** un medio de promoción nuevo **para** solventar los inconvenientes publicitarios surgidos en la pandemia.
8. **Como** corista internacional, **quiero** tener un medio de comunicación **para** que mis familiares reciban las noticias de mis presentaciones.
9. **Como** interesado en el festival, **quiero** información logística **para** conocer respecto a los talleres.
10. **Como** creador de contenido, **quiero** un medio de escritura de publicaciones en las redes sociales **para** informar al público en general.
11. **Como** organizador del festival, **quiero** un medio digital **para** que la comunidad universitaria se entere de las actividades realizadas en el festival.
12. **Como** parte del equipo organizador, **quiero** un resumen de las actividades a realizar ese día **para** no confundirme ni olvidarme nada.
13. **Como** asistente a los eventos, **quiero** una forma de saber respecto a los coros participantes de cada concierto **para** informarme antes de comprar las entradas.
14. **Como** organizador del festival, **quiero** un historial de todos los anteriores festivales **para** mostrar el progreso del proyecto.
15. **Como** diseñador del festival, **quiero** un medio que se adapte al tema artístico del presente año **para** mantener un estilo homogéneo.
16. **Como** organizador del festival, **quiero** una galería multimedia **para** mostrar de una forma visible la experiencia de los eventos.
17. **Como** parte del comité organizador, **quiero** varios medios en los que los interesados puedan enviar preguntas **para** atender las consultas de la mayor cantidad de personas posible.
18. **Como** encargado de responder las preguntas, **quiero** un módulo que pueda generar réplicas automáticas **para** optimizar el proceso de contestación.
19. **Como** persona que no conoce la universidad o los teatros, **quiero** una guía de cómo llegar a los eventos **para** no perderme.
20. **Como** interesado en los eventos del festival, **quiero** información de cómo y dónde comprar las entradas **para** no confundirme.
21. **Como** persona interesada en los eventos, **quiero** un recordatorio por evento **para** no perderme de información importante.
22. **Como** interesado en el festival, **quiero** un lugar donde pueda consultar preguntas frecuentes **para** guiarme de mejor manera.

Con estas descripciones identificadas, se pudo construir un Spike Arquitectónico que, a su vez, serviría como base para el establecimiento de la Metáfora del Negocio. Es preciso enfatizar en el hecho de que en la Fase de Planificación se completarán las historias de usuario con los demás elementos del apartado 3.1.1.2.

3.1.2 Spike Arquitectónico

Una vez definidas las primeras historias de usuario, el equipo de desarrollo ya puede centrarse en las tecnologías y herramientas que se usarán en el proyecto. Al primer acercamiento del estudio y familiarización del equipo con las tecnologías se le conoce como Spike Arquitectónico (Ambler, 2002). Este tiene la finalidad de construir una idea simple de qué implica el sistema, llamada Metáfora del Negocio o Metáfora del Sistema.

Ambler (2002) indica que, al ser una metodología ágil, se reemplazan todos aquellos conceptos tradicionales de arquitectura. Se recomienda el uso de diagramas para mejorar el entendimiento del sistema, pero estos no tienen que seguir los estándares rígidos de UML[2], sino enfocarse en el contenido del diagrama. De la misma forma, el autor explica que la arquitectura debe enfocarse en la flexibilidad ante el cambio, en lugar de establecerse como una arquitectura absoluta para todo el sistema, como en las metodologías tradicionales.

3.1.2.1 Arquitectura Utilizada

La arquitectura que se empleó en el presente proyecto fue una combinación entre las arquitecturas “serverless” (sin servidor) y “MVC”. La arquitectura sin servidor refiere a una idea de que el equipo de desarrollo no se enfoca en la administración y manejo de un servidor (Amazon Web Services, 2016). El concepto se alinea muy bien a la metodología XP, de modo que es muy flexible a los cambios y crece conforme al proyecto. De esta forma, el equipo de desarrollo puede enfocarse más en el producto, y apoyarse en los servicios que ofrece la empresa que maneja el servidor y la infraestructura, permitiendo una arquitectura más flexible.

Desde el lado de la aplicación que se desarrolló, se aplicó la arquitectura MVC o Modelo-Vista-Controlador. Es un marco de trabajo que tiene tres principales elementos que se especializan en un ámbito específico dentro de la aplicación: el modelo (responsable del manejo de los datos), la vista (o presentación de la información) y el controlador (una interfaz[3] entre el modelo y la vista, y la encargada de toda la lógica del negocio) (Lou, 2016).

Como se puede apreciar en la Ilustración 5, la vista debería comunicarse únicamente con el controlador a través de las acciones del usuario. El controlador puede modificar la vista directamente, pero es recomendado que se manipule el modelo. Este último actualiza la vista según los datos, sin regresar hacia el controlador.

---

[2] **Lenguaje Unificado de Modelado (UML):** Herramienta usada “para forjar un lenguaje de modelado visual común y semántica y sintácticamente rico para la arquitectura, el diseño y la implementación de sistemas de software complejos, tanto en estructura como en comportamiento”. (Lucidchart, 2017)

[3] **Interfaz:** “Dispositivo capaz de transformar las señales generadas por un aparato en señales comprensibles por otro” (Lexico, 2019)

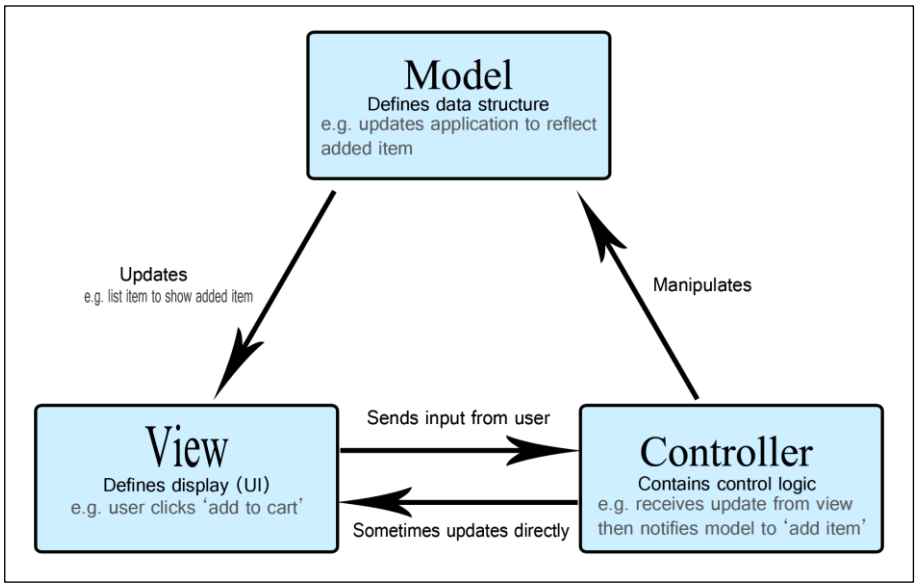


Ilustración 5 MDN Web Docs (2021) Diagrama de la arquitectura MVC [Figura]. Recuperado de: https://developer.mozilla.org/en-US/docs/Glossary/MVC

La arquitectura MVC también es aplicable para la metodología XP, por su facilidad en la escalabilidad. De la misma forma, en el proyecto, se siguió la lógica de MVC para comunicar las diferentes partes de la aplicación y que gestionen únicamente lo que les corresponda.

3.1.2.2 Tecnologías empleadas

Una vez analizados los modelos de arquitecturas en los cuales se ha basado la aplicación, es pertinente describir las tecnologías que se usaron. En primer lugar, alineándose con la arquitectura sin servidor, se utilizó Google Firebase, parte de Google Cloud. Con ella, se puede consumir servicios tales como: manejo de notificaciones, bases de datos, repositorios de información, autorizaciones, entre otros, haciendo uso del tablero de control[4] de la herramienta, llamado Firebase Console. Estos productos son administrados fuera de la aplicación, y se los implementa en el proyecto con el uso de bibliotecas[5].

Para la construcción de la aplicación, se empleó una cadena de frameworks y bibliotecas. En primer lugar, se encuentra React Native que "es un framework de código abierto para construir aplicaciones de Android y iOS usando React y las capacidades nativas

[4] **Tablero de Control (Dashboard):** "Es una herramienta de administración de información que rastrea, analiza y presenta indicadores de rendimiento clave, métricas y puntos de datos clave para monitorear la salud de un negocio, departamento o proceso específico". (Klipfolio, 2016)
[5] **Bibliotecas:** "es un conjunto de subprogramas utilizados para desarrollar software" (Sensagent, 2010)

de la plataforma de la app" (React Native, 2021). A su vez, React es un conjunto de bibliotecas de JavaScript para la construcción de interfaces de usuario que, en el caso del presente proyecto, constituye núcleo de la lógica y lenguaje de programación usado por React Native (React, 2019).

Para facilitar el desarrollo de aplicaciones basadas en la lógica de React, se empleó a Expo, un conjunto de herramientas y servicios que "ayudan al equipo de trabajo a desarrollar, construir, desplegar e iterar rápidamente en iOS, Android y apps web de una misma base de código de JavaScript/TypeScript" (Expo, 2020).

Finalmente, para llegar a satisfacer todos los requerimientos descritos y recogidos de las historias de usuario, es pertinente el uso de SendGrid, una tecnología que administra la comunicación por correo electrónico por medio de una API[6]. En el caso del proyecto, no se utilizó dicha herramienta para enviar correos en masa a los usuarios de la aplicación, sino para permitir una comunicación unidireccional automatizada desde un formulario hasta el correo del equipo del festival.

3.1.2.3 Familiarización con las Herramientas

El objetivo último del Spike Arquitectónico es la familiarización del equipo de desarrollo con las tecnologías descritas. Es decir, es fundamental reconocer si es viable el uso de alguna herramienta antes de comenzar la planificación de la aplicación, a fin de encontrar diferentes acercamientos a la solución propuesta.

Con relación a las tecnologías descritas para el presente proyecto, se han realizado pequeñas aplicaciones de prueba para determinar que el núcleo de React Native + Expo es propicio para los propósitos pertinentes, complementando con conexiones con servicios y APIs de Google Firebase, como el uso de la base de datos (Firestore), repositorio de datos (Storage), asignación de roles a usuarios en Firebase Console. Igualmente, se han analizado ejemplos implementados de SendGrid, así como de la integración entre todas estas tecnologías, como el descrito por Malek (2019).

De esta forma, se concluyó que las herramientas descritas pueden cumplir con los requerimientos obtenidos a partir de las historias de usuario. Nótese que la naturaleza adaptable de la metodología de desarrollo sugiere que se pudiera cambiar la arquitectura utilizada a lo largo de la construcción del proyecto, permitiendo incluso la expansión de la aplicación hacia tecnologías propias.

---

[6] **Interfaz de Programación de Aplicaciones (API):** "es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones". (Red Hat, 2019)

3.1.2.4 Aplicación al Proyecto

Una vez que se han descrito las tecnologías que se emplearon en la construcción de la aplicación, así como la viabilidad determinada por la familiarización con las mismas, es pertinente mostrar gráficamente el acercamiento arquitectónico que se empleó, describiendo las relaciones entre herramientas, así como el papel del usuario. La Ilustración 6 resume estas interacciones entre componentes, basándose en diagramas UML, pero no se los sigue estrictamente. Como indica Ambler (2002), en las metodologías ágiles es mejor representar correctamente las ideas, antes que regirse a estándares rigurosos o hacer que los diagramas se vean agradables.

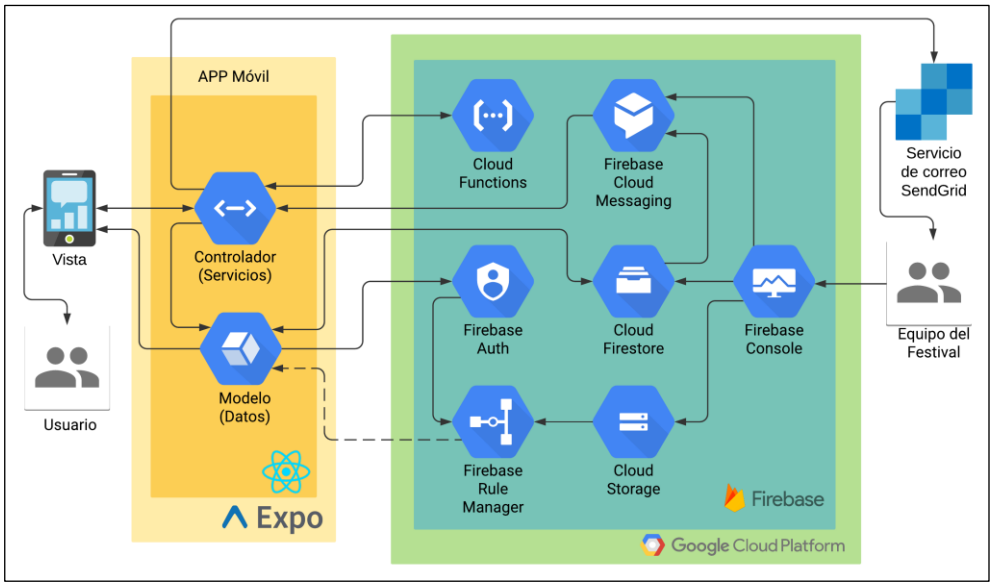


Ilustración 6 Silverio, V. (2021) Diagrama de la arquitectura que sigue la aplicación móvil. [Figura].

Como se puede apreciar en la Ilustración 6, la arquitectura se compone de tres partes fundamentales: los actores, el núcleo de la aplicación y los servicios externos. En primera parte, los actores son de dos tipos: el usuario general y el equipo del festival (generadores de contenido, director del festival, equipo técnico, comité organizador). Los usuarios generales se comunicarán con los servicios únicamente a través de la aplicación móvil: ahí podrán encontrar todo el contenido multimedia, información de los coros, cronogramas, entre otros elementos. Por otro lado, el equipo del festival, con el fin de agregar información a la base de datos, enviar notificaciones o agregar nuevo contenido multimedia, debe hacerlo por medio del Firebase Console.

En segundo lugar, se puede apreciar el núcleo de la aplicación siendo manejado a partir de una sub-arquitectura MVC, usando las tecnologías propuestas de React Native + Expo. La vista es la ventana por la que el usuario general puede acceder a la información, el

controlador se encarga de los servicios dentro y fuera de la aplicación, y el modelo se enfoca de la administración de los servicios de Firebase enfocados al manejo de datos.

Finalmente, en tercer lugar, siguiendo el concepto de arquitectura serverless, se encuentra la parte de los servicios externos. En esta sección figuran los siguientes productos que tendrán un uso específico, según los requerimientos extraídos a partir de las historias de usuario:

- **Cloud Functions:** Es el servicio encargado de realizar funciones automatizadas despendiendo de ciertas condiciones dentro de la aplicación. En este concepto entraría el sistema de respuestas automáticas a ciertas preguntas del usuario.
- **Firebase Cloud Messaging:** Es la herramienta de notificaciones de Firebase. Con ella, se pueden administrar los mensajes dentro y fuera de la aplicación, a fin de hacer llegar información importante a cada usuario general de la aplicación.
- **Cloud Firestore:** Es la base de datos que se usó en el proyecto. Dentro de ella, se puede encontrar información estructurada, organizada y centralizada respecto a los coros, eventos, talleres, cronogramas, entre otros.
- **Cloud Storage:** Es el repositorio de los elementos multimedia que usaría la aplicación (imágenes o videos). Se diferencia de la base de datos en el sentido de que no está estructurada, y guarda archivos de gran tamaño.
- **Firebase Console:** Es el tablero que facilita el acceso y manejo de todos los servicios que provee Firebase de parte del equipo del festival. Cumple con elementos importantes de personalización como control de roles por usuario, tiene secciones correctamente etiquetadas y guías para facilitar el entendimiento del usuario y cualquier acción queda registrada en logs[7]. De esta manera, no es necesaria la realización de otra aplicación para el manejo de información de parte del comité organizador.
- **Firebase Auth:** Maneja las credenciales dentro de la aplicación, como los datos de inicio de sesión. Puede ser usada para identificar a las personas que se han suscrito a transmisiones en vivo con el fin de presentar el contenido por el cual han pagado, como se ha descrito en las historias de usuario. Sin embargo, se hace hincapié en el hecho de que no se ha concebido el uso de sesiones para acceder a contenido exclusivo, sino se utiliza el paso de tokens[8] autorizados.

[7] **Registro (log):** “es un archivo de texto en el que constan cronológicamente los acontecimientos que han ido afectando a un sistema informático (programa, aplicación, servidor, etc.), así como el conjunto de cambios que estos han generado.” (Ankama, 2015)

[8] **Token:** “es una unidad de valor que una organización crea para gobernar su modelo de negocio y dar más poder a sus usuarios para interactuar con sus productos, al tiempo que facilita la distribución y reparto de beneficios entre todos sus accionistas” (Communications, 2021)

- **Firebase Rule Manager:** Administra las condiciones en las que el usuario puede acceder a la información de la aplicación. Es un servicio complementario al Firebase Auth para el acceso a suscripciones. A partir del administrador de reglas, se envía información filtrada desde el Cloud Storage. Para representar el contenido filtrado, en la Ilustración 6, se diagrama una flecha entrecortada.
- **SendGrid:** Como se indicó anteriormente, SendGrid es el servicio de envío de correos de forma automática. Se ha utilizado esta herramienta con el fin de enviar comentarios o preguntas al equipo del festival desde la aplicación móvil, y mantener un buen nivel de comunicación entre el público de los eventos y el comité organizador.

De esta forma, las tres partes (los actores, núcleo de la aplicación y servicios externos) componen toda la arquitectura usada para la implementación del sistema. Con esta descripción, se puede establecer la Metáfora del Negocio y dar paso a la fase de planificación de la construcción de la app móvil, para determinar los elementos de la arquitectura que sí estarían dentro del alcance del proyecto.

3.1.3 Metáfora del Negocio

Finalmente, como tercer elemento destacable de la Fase de Exploración de XP, se encuentra la Metáfora del Negocio o Metáfora del Sistema. Esta es una explicación simple que facilita el entendimiento de la aplicación a todo actor interesado en el proyecto, sea este técnico o no técnico. De la misma manera, la importancia de una metáfora del negocio es poder servir como resumen de la fase de exploración para ser tratada más a fondo en la planificación de lanzamiento (Ambler, 2002).

3.1.3.1 Metáfora

La Real Academia Española (2020) define la metáfora como la “traslación del sentido recto de una voz a otro figurado, en virtud de una comparación tácita”. Igualmente, Léxico (2019) indica que una metáfora es la “figura retórica de pensamiento por medio de la cual una realidad o concepto se expresan por medio de una realidad o concepto diferentes con los que lo representado guarda cierta relación de semejanza”.

A partir de ambas definiciones, se puede entender a una metáfora como una figura retórica que permite explicar un concepto mediante una comparación, o relación de semejanza, con otro elemento.

En el caso de la metáfora del sistema, por lo tanto, se refiere a una realidad conocida para el usuario, pero que se relacione correctamente con el software, de forma que lo defina con alto grado de precisión.

3.1.3.2 Enfoque al Proyecto

Una metáfora del negocio que aplica para el presente proyecto es el concepto de "Programa General". Un usuario podría adquirir un folleto que contenga el programa general en cualquier momento, el cual contendría la especificación del lugar, hora y fecha de los eventos, así como información de los coros participantes e ilustraciones de presentaciones anteriores. Se tendría una sección para poder contactar directamente con el equipo organizador del festival. De la misma forma, los generadores de contenido del equipo del festival podrían generar nuevos diseños cada tiempo para ser distribuidos a los interesados en el festival, de manera que sirva como un medio publicitario. Finalmente, serviría como recuerdo para los participantes y coristas nacionales e internacionales, respecto a las experiencias en los eventos.

3.1.4 Recapitulación

Se ha seguido el proceso de la fase de Exploración de la metodología XP, con la finalidad de poder dar una primera visión de los requerimientos, arquitectura y del concepto del proyecto. En primer lugar, se discutieron las historias de usuario con el cliente, manteniendo las características INVEST y de las 3C's. A partir de ellas, se podría concebir un primer acercamiento a la arquitectura del proyecto, así como la familiarización con las herramientas que se aplicaron en el proyecto. Finalmente, con una idea clara del flujo de los servicios en la app móvil, se puede simplificar la manera en cómo se entiende al proyecto, proveyendo la metáfora del negocio. Esta última, es el puente de conexión entre la fase de Exploración con la de Planificación.

3.2 Fase de Planificación

Con la exploración realizada en la primera fase, se pueden analizar las tareas que deben ser realizadas para una correcta implementación del sistema. Las historias de usuario escritas en la fase de exploración se completan en la fase de planificación y se las divide en iteraciones con sus respectivas fechas de entrega. Posteriormente, cada iteración es estudiada con mayor profundidad determinando las acciones seguidas en la fase de iteraciones. La presente sección examina el proceso de planificación de XP.

3.2.1 Planificación de lanzamiento

La planificación de lanzamiento es el primero de los dos pasos en la construcción de la planificación dentro de la metodología XP. Se enfoca en la determinación de las características principales que se requiere en el sistema, y sus respectivas fechas de entrega

(Anwer et al., 2017). En otras palabras, es la subetapa encargada de organizar las historias de usuario y dividirlas en iteraciones para poder realizar entregas periódicas al usuario.

En este sentido, como lo explican Maida y Pacienzia (2015), se puede entender un plan de lanzamiento como el conjunto de los planes de iteración (ver sección 3.2.2). Es preciso, por lo tanto, determinar el alcance del proyecto antes de definir las iteraciones y las fechas de entrega. Para este fin, Anwer et al. (2017) explican que primero se deben escribir las historias de usuario, luego asignar un valor de prioridad a cada una para ordenarlas y, finalmente, escoger una porción de ellas para cada iteración.

Pese a que Wells (2002) sugiere que se las estimaciones de las tareas se realicen una vez que se han definido las historias de cada iteración, para el presente proyecto es necesario realizar la valoración de los tiempos antes de construir las iteraciones. Esto es dado por la característica de tiempo limitado para el desarrollo. Por consiguiente, en lugar de establecer los requerimientos en cada una de las entregas y proveer, a partir de ello, una estimación de la extensión total se ha determinado un rango de tiempo máximo para cada una de las iteraciones y el número posible de historias que se pueden implementar en dicho rango, según una valoración previa de la duración individual.

Ilustración 7 Silverio, V. (2021) Proceso de la elaboración del Plan de Lanzamiento [Figura].

En la Ilustración 7, se presenta el proceso seguido para la elaboración del plan de lanzamiento, adaptado según las necesidades propias del proyecto. A continuación, se analiza cada uno de los cuatro pasos del proceso.

3.2.1.1 Determinación de la prioridad

Partiendo de las 22 historias de usuario expuestas en el apartado 3.1.1.3, es requerido que, junto con el cliente, se atribuya un valor de prioridad a cada tarjeta, con el fin de que

servirán como punto de partida para poder determinar cuáles historias se implementan en qué orden (Anwer et al., 2017).

Para este objetivo, el cliente asignó valores de prioridad de 1 hasta 3 a cada una de las historias de usuario. Posteriormente, estos valores numéricos se traducen a etiquetas de prioridad: “Baja”, “Media”, “Alta”. De esta forma se podrían ordenar todos los requerimientos de mayor importancia a menor importancia.

Cabe resaltar que el orden de las historias es relevante para el trabajo, dado que indica al equipo de desarrollo qué requerimientos implementar primero. Al ordenarlas según la importancia, se encamina al proyecto a dar prioridad a aquellas historias que aporten mayor valor a la empresa, y dejar al último a los requerimientos que no aporten mucho valor.

Adicionalmente, con el objetivo de proporcionar una estructura más propicia para la implementación, se basó en la organización dada por la prioridad y se reordenó algunas de las historias, agrupándolas por similitud en las tareas de implementación. Estas acciones, como lo describen Anwer et al. (2017) y Ambler (2002), son parte del balanceo de carga que se debe realizar cuando se van a construir las iteraciones.

3.2.1.2 Selección de las historias que implementar

En primer lugar, se acordó con el usuario que, según la prioridad de cada historia, se tomaría en cuenta la factibilidad de la implementación. De este modo, por lo tanto, se añadirían al proyecto todas las historias con prioridad alta, algunas historias con prioridad media (aquellas que mejor se adapten al objetivo del proyecto), y no se implementaría ninguna historia con prioridad baja, dado que no aportan con mucho valor al negocio actualmente. Las historias contempladas en este último grupo y que, por consiguiente, no se implementarán, son aquellas relacionadas con añadir un historial completo de los anteriores festivales, con mantener un tema artístico similar a la página web y con adjuntar una galería multimedia.

Asimismo, existen algunas excepciones en cuanto a qué requerimientos satisfacer, independientes de la prioridad que tengan. Los motivos de estas excepciones están principalmente vinculados al alcance de la aplicación, es decir, no pueden ser implementados en el tiempo previsto para la realización de este trabajo. En específico, las excepciones de implementación contemplan: un proveedor de noticias (newsfeed), alimentado por publicaciones en las redes sociales; un sistema que provea respuestas automáticas a las preguntas de los usuarios; y un sistema que administre los pagos de transmisiones de los eventos individuales a los que algún usuario se haya suscrito.

De esta manera, tomando el resto de las tarjetas, se han seleccionado 16 de las 22 historias de usuario para formar parte del presente proyecto, las cuales serán divididas en

tres iteraciones. A cada requerimiento de esta selección, se añadió una estimación, un título, un identificador, las tareas relacionadas y los respectivos criterios de aceptación.

3.2.1.3 Asignación de estimaciones

Basándose en las tareas relacionadas con cada historia de usuario, el equipo de desarrollo ha asignado una estimación a cada requerimiento. Cabe recordar que los valores de las estimaciones se refieren al número de días que toma en implementarlo por completo (hasta que satisfaga los criterios de aceptación), sin ninguna distracción y sabiendo exactamente qué hacer. En este proceso únicamente interviene el equipo de desarrollo, el cliente no interfiere en la estimación.

Posteriormente, la suma de todas las estimaciones de una iteración forma el concepto de "Velocidad del Proyecto", la cual es un referente para realizar cambios en la planificación de las iteraciones en caso de algún fallo (Wells, 2002).

3.2.1.4 Construcción de las iteraciones

Como indican Anwer et al. (2017), una iteración es un conjunto de historias de usuario que se desarrollan en un tiempo determinado antes de entregar un avance al usuario. Generalmente, explican los autores, el tiempo de vida de una iteración varía de una a cuatro semanas.

En el caso del proyecto, se contó con un tiempo estimado de siete semanas aproximadamente. Por lo tanto, se dividió en tres iteraciones de tres semanas, dos semanas y dos semanas, respectivamente. Como se explicó anteriormente, la primera iteración contiene aquellas historias de usuario con la mayor prioridad, la segunda iteración representa prioridades altas y medias, y la última iteración trabaja con historias de baja prioridad.

Número de Iteración	Temas de las historias de usuario	Duración (Semanas)	Prioridades de las historias	Historias contenidas
Iteración 1	Estructuras y bases.	3	Mayormente Altas.	Desde 001, hasta 007.
Iteración 2	Notificaciones, mapas y formulario de comunicación.	2	Medias y Altas.	Desde 008, hasta 011.
Iteración 3	Detalles de la información relevante y FAQs.	2	Únicamente Medias.	Desde 012, hasta 016.

Tabla 6 Silverio, V. (2021) Resumen de las características de las iteraciones del proyecto [Tabla]

Como se puede apreciar en la Tabla 6, la primera iteración trata las historias relacionadas con las estructuras básicas de la aplicación y contiene siete historias de usuario

a ser implementadas en tres semanas. La segunda iteración, implementa las historias de usuario referentes a las notificaciones, mapas y al formulario de preguntas; y contiene cuatro historias en un rango de dos semanas. Finalmente, la tercera iteración aborda el llenado de datos en la aplicación y las preguntas frecuentes, en referencia a 5 historias de usuario culminadas en dos semanas.

3.2.1.5 Cronograma de implementación y entregas

Como ya se explicó anteriormente, el objetivo último de un plan de lanzamiento es poder escoger qué historias de usuario se realizan en qué iteración y con qué fecha de entrega. En la Ilustración 8, se encuentra un cronograma que resume las 16 historias de usuario con las respectivas fechas de implementación asignadas.

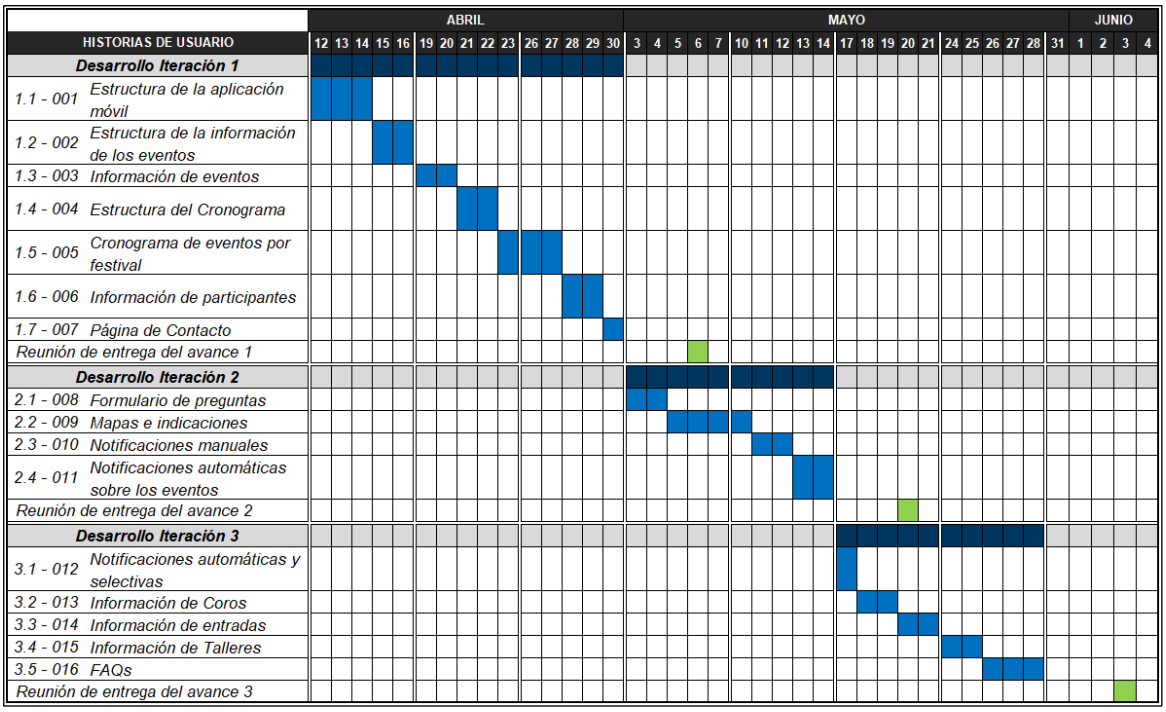


Ilustración 8 Silverio, V. (2021) Plan de Lanzamiento: Cronograma de implementación y entregas [Figura].

Adicionalmente, al finalizar cada una de las iteraciones, se ha agendado una reunión con el cliente para poder realizar la entrega de los avances respectivos, las cuales se realizan los primeros jueves consecuentes a la culminación de cada iteración de desarrollo. De la misma manera, se analiza si se agregan más historias de usuario a las siguientes iteraciones. Cabe recalcar que tras analizar las fechas de entrega de las iteraciones se ha consultado la factibilidad con el cliente, para darle la oportunidad de realizar cualquier cambio, como lo sugieren Maida y Pacienzia (2015).

3.2.2 Planificación de Iteraciones

Es la segunda de las dos sub-fases dentro de la etapa de planificación en la metodología XP. El objetivo principal de la planificación de iteraciones es poder identificar las actividades que se deben realizar para implementar las historias de usuario seleccionadas en el plan de lanzamiento. Para esto, se escriben tarjetas de tareas que son una recopilación de las acciones que realizar con el fin de poder cumplir con los criterios de aceptación de cada una de las historias de usuario. La escritura de estas tareas no tiene un formato específico y contienen un carácter técnico, por lo que no es necesario que sean escritas junto con el cliente (Anwer et al., 2017).

Wells (2002) recomienda que la determinación de las tareas específicas de cada tarea de cada iteración se realice al comienzo del ciclo de desarrollo, con la experiencia proporcionada por la anterior iteración.

Así, por lo tanto, en las siguientes secciones se profundiza el tema de cada iteración con el respectivo desglose de actividades.

3.2.2.1 Iteración 1

El tema principal de la primera iteración es la estructuración del proyecto y el establecimiento de las bases. De esta manera, en la Ilustración 9 se exponen las tareas necesarias para implementar correctamente cada una de las siete historias de usuario definidas en la primera iteración.

HISTORIAS	TAREAS			
Estructura de la aplicación móvil	Crear el proyecto de React Native en Expo.	Implementar un navegador entre páginas.	Agregar un navegador entre páginas (stack).	Armar el archivo de configuración.
	Estructurar el proyecto.	Crear los archivos de pantallas	Exportar la aplicación como APK.	
Estructura de la información de los eventos	Determinar las claves de la colección "Eventos".	Crear y configurar la colección "Eventos".	Leer todos los eventos de la colección.	Ordenar la lista de eventos por fecha.
Información de eventos	Mostrar toda la información del evento individual	Dirigir desde la lista de eventos al evento individual.		
Estructura del Cronograma	Crear un calendario dinámico	Determinar las claves de la colección "Eventos"		
Cronograma de eventos por festival	Presentar la información de la colección "Eventos".	Redirijir al evento correspondiente.		
Información de participantes	Determinar las claves de la colección "Coros".	Crear y Configurar la colección "Coros".	Enlazar las colecciones de “Coros” y “Eventos”.	Dirigir la lista de participantes a la pantalla de cada coro.
Página de Contacto	Configurar la pantalla de “Contáctanos”	Añadir enlaces externos a las redes sociales		

Ilustración 9 Silverio, V. (2021) Plan de Iteración 1: Resumen de tareas por historia de usuario [Figura].

3.2.2.2 Iteración 2

La principal temática de la segunda iteración es la implementación de notificaciones, mapas y el formulario de comunicación. La Ilustración 10 muestra las tareas necesarias para poder implementar las historias de usuario del segundo ciclo de desarrollo.

HISTORIAS	TAREAS			
Formulario de preguntas	Implementar el funcionamiento del formulario (SendGrid)	Presentar confirmación de envío del mail.		
Mapas e indicaciones	Determinar las claves de la colección "Lugares".	Crear, configurar y llenar la colección "Lugares".	Agregar indicaciones y los mapas según la colección.	Dirigir a la pantalla específica de cada lugar
Notificaciones manuales	Pedir los permisos para el envío de notificaciones	Guardar los tokens de los dispositivos suscritos	Configurar la herramienta de mensajería de Firebase	Verificar que se envíen notificaciones correctamente
Notificaciones automáticas sobre los eventos	Determinar los casos de envío de notificaciones generales	Implementar los casos automáticos de notificaciones		

Ilustración 10 Silverio, V. (2021) Plan de Iteración 2: Resumen de tareas por historia de usuario [Figura].

3.2.2.3 Iteración 3

La Ilustración 11 presenta las tareas necesarias para satisfacer los requerimientos de la tercera iteración, cuyo tema principal es el manejo de toda la información relevante de la aplicación móvil.

HISTORIAS	TAREAS				
Notificaciones automáticas y selectivas	Determinar los casos de envío de notificaciones específicas	Añadir la posibilidad de suscribirse a un evento específico	Registrar el token del dispositivo al evento.	Implementar los casos específicos de notificaciones	
Información de Coros	Llenar los datos en la colección “Coros”.	Enlistar todos los coros de la colección “Coros”	Desplegar la información completa de cada documento	Mostrar toda la información del Coro Individual	
Información de entradas	Determinar las claves de la colección "Entradas".	Crear, configurar y llenar la colección "Entradas".	Mostrar indicaciones de compra según la colección.	Mostrar cada tipo de entrada respectivo a cada evento individual.	
Información de Talleres	Determinar las claves de la colección "Talleres".	Crear, configurar y llenar la colección "Talleres".	Enlistar todos los talleres de la colección “Talleres”	Desplegar la información completa del documento	Mostrar toda la información del Taller Individual
FAQs	Determinar las claves de la colección "FAQs".	Crear, configurar y llenar la colección "FAQs".	Mostrar las preguntas y respuestas de la colección.		

Ilustración 11 Silverio, V. (2021) Plan de Iteración 3: Resumen de tareas por historia de usuario [Figura].

3.2.3 Adaptación de las Historias de usuario

Una vez que se han definido y asignado todos los elementos de las historias de usuario, según como se explicó en la sección 3.1.1.2, y según los lineamientos proporcionados por los planes de iteración y de lanzamiento, se puede colocar toda la información de las historias en una tarjeta específica que resuma dichos componentes de una manera gráfica y entendible.

Aunque no es sub-fase explícita de la etapa de desarrollo, es pertinente realizar la adaptación de las historias de usuario para guiar correctamente el proceso de desarrollo.

Seguidamente, en la Ilustración 12, se presenta el formato base usado en el resto de las historias de usuario, donde se colocan espacios dedicados para el identificador, título, prioridad, estimación, descripción, criterios de aceptación y tareas por realizar.

Identificador de la Historia	Título de la Historia	Prioridad	Estimación (en días)
Historia de Usuario:			
Como un <Usuario>,			
Quiero una <Característica>			
Para un <Beneficio>			
Criterios de Aceptación:			
Dado un <Contexto>,			
Cuando una <Acción se lleva a cabo>			
Entonces un <Conjunto particular de consecuencias observables se debería obtener>			

Tareas por realizar:

Ilustración 12 Silverio, V. (2021) Plantilla de la tarjeta de las historias de usuario que se ha usado en el proyecto (anverso y reverso) [Figura].

Basándose en la plantilla de la Ilustración 12, se exponen entonces las 16 historias de usuario que se implementaron en el proyecto, las cuales incluyen todos los elementos analizados.

Identificador de la Historia	Título de la Historia	Prioridad	Estimación (en días)
1.1 - 001	*Estructura de la aplicación móvil*	Alta	3
Historia de Usuario:			
Como promotor de cultura,			
Quiero un medio de promoción nuevo			
Para solventar los inconvenientes publicitarios surgidos en la pandemia.			
Criterios de Aceptación:			
Dado que tengo un celular que funciona correctamente y me he descargado la app del festival,			
Cuando intento inicializar la aplicación,			
Entonces se abre sin inconvenientes.			

Tareas por realizar:
Crear el proyecto de React Native en Expo.
Implementar un navegador entre páginas (menú lateral).
Agregar un navegador entre páginas (stack).
Armar el archivo de configuración que contenga información importante del festival como colores, fechas, homenaje, etc.
Estructurar el proyecto (creación de carpetas) según la necesidad de la aplicación
Crear los archivos de pantallas necesarias en el proyecto (inicio, cronograma, eventos (lista), evento (individual), coros (lista), coro (individual), talleres (lista), taller (individual), contacto (formulario), entradas, mapas, FAQs.
Exportar la aplicación como APK.

Ilustración 13 Silverio, V. (2021) Historia de Usuario 1.1 – 001: "Estructura de la aplicación móvil" (anverso y reverso) [Figura].

Identificador de la Historia	Título de la Historia	Prioridad	Estimación (en días)
1.2 - 002	*Estructura de los eventos.*	Alta	2
Historia de Usuario:			
Como	organizador del festival,		
Quiero	un medio digital		
Para	que la comunidad universitaria se entere de las actividades realizadas en el festival.		
Criterios de Aceptación:			
Dado	que he abierto la app móvil,		
Cuando	me dirijo a la sección de "Eventos",		
Entonces	se muestra los eventos próximos ordenados por fecha.		

Tareas por realizar:
Determinar las claves que conformarán a los documentos de la colección "Eventos".
Crear y configurar la colección "Eventos".
Configurar la pantalla de lista de eventos para leer todos los eventos de la colección.
Ordenar la lista de eventos por fecha.

Ilustración 14 Silverio, V. (2021) Historia de Usuario 1.2 – 002: "Estructura de la información de los eventos." (anverso y reverso) [Figura].

Identificador de la Historia	Título de la Historia	Prioridad	Estimación (en días)
1.3 - 003	*Información de eventos*	Alta	2
Historia de Usuario:			
Como	asistente del festival,		
Quiero	un programa de mano		
Para	tener la información centralizada de los eventos.		
Criterios de Aceptación:			
Dado	que puedo visualizar el cronograma general de actividades o la lista de eventos,		
Cuando	selecciono un evento,		
Entonces	se despliega toda la información de los eventos (nombre, lugar, fecha, hora, costo, participantes, etc.) de forma completa y estructurada.		

Tareas por realizar:
Configurar la pantalla de evento individual para mostrar toda la información de una manera entendible para el usuario.
Organizar el navegador en referencia a la lista de eventos y al evento individual.

Ilustración 15 Silverio, V. (2021) Historia de Usuario 1.3 – 003: "Información de eventos" (anverso y reverso) [Figura].

Identificador de la Historia	Título de la Historia	Prioridad	Estimación (en días)
1.4 - 004	*Estructura del Cronograma*	Media	2
Historia de Usuario:			
Como	parte del equipo organizador,		
Quiero	un resumen de las actividades a realizar ese día		
Para	no confundirme ni olvidarme nada.		
Criterios de Aceptación:			
Dado	que he abierto la app móvil,		
Cuando	me dirijo a la sección de "Cronograma",		
Entonces	se muestra un calendario con las fechas respectivas al festival.		

Tareas por realizar:
Configurar la pantalla para tener un calendario dinámico según las fechas establecidas en el documento de configuración.
Determinar las claves de los documentos de la colección "Eventos" para ser presentado en el resumen del cronograma.

Ilustración 16 Silverio, V. (2021) Historia de Usuario 1.4 – 004: "Estructura del Cronograma" (anverso y reverso) [Figura].

Identificador de la Historia	Título de la Historia	Prioridad	Estimación (en días)
1.5 - 005	*Cronograma de eventos por festival*	Alta	3
Historia de Usuario:			
Como una persona interesada en el festival,			
Quiero un cronograma de eventos			
Para visualizar la información.			
Criterios de Aceptación:			
Dado que puedo visualizar el calendario,			
Cuando me desplazo a lo largo de las fechas,			
Entonces puedo ver la información destacada (nombre, lugar hora) y resumida de cada evento.			

Tareas por realizar:
Configurar cada evento singular para presentar la información obtenida de la colección "Eventos". Modificar al navegador para que los eventos del cronograma redirijan al evento correspondiente.

Ilustración 17 Silverio, V. (2021) Historia de Usuario 1.5 – 005: "Cronograma de eventos por festival" (anverso y reverso) [Figura].

Identificador de la Historia	Título de la Historia	Prioridad	Estimación (en días)
1.6 - 006	*Información de participantes*	Alta	2
Historia de Usuario:			
Como asistente a los eventos,			
Quiero una forma de saber respecto a los coros participantes			
Para informarme antes de comprar las entradas.			
Criterios de Aceptación:			
Dado que he visualizado un evento que me interesó,			
Cuando expanda la información de participantes,			
Entonces puedo ver todos los coros integrantes del evento.			

Tareas por realizar:
Determinar las claves que conformarán a los documentos de la colección "Coros". Crear y Configurar la información de la colección "Coros". Enlazar la colección "Coros" con la colección "Eventos". Configurar la navegación para llevar a la pantalla específica de cada coro según la lista de participantes en la pantalla de evento individual.

Ilustración 18 Silverio, V. (2021) Historia de Usuario 1.6 – 006: "Información de participantes" (anverso y reverso) [Figura].

Identificador de la Historia	Título de la Historia	Prioridad	Estimación (en días)
1.7 - 007	*Apartado de Contacto*	Alta	1
Historia de Usuario:			
Como interesado en los eventos,			
Quiero una sección de contactos			
Para comunicarme con el festival.			
Criterios de Aceptación:			
Dado que he abierto la app móvil,			
Cuando me dirijo a la sección de "Contacto",			
Entonces encuentro todos los canales de contacto con el equipo del festival.			

Tareas por realizar:
Configurar la pantalla de "Contáctanos", incluido los campos del formulario. Añadir enlaces externos a las redes sociales del festival.

Ilustración 19 Silverio, V. (2021) Historia de Usuario 1.7 – 007: "Página de Contacto" (anverso y reverso) [Figura].

Identificador de la Historia	Título de la Historia	Prioridad	Estimación (en días)
2.1 - 008	*Formulario de preguntas*	Alta	2

Historia de Usuario:

Como parte del comité organizador,
Quiero varios medios en los que los interesados puedan enviar preguntas
Para atender las consultas de la mayor cantidad de personas posible.

Criterios de Aceptación:

Dado que he abierto la app móvil,
Cuando me dirijo a la sección de "Contacto",
Entonces puedo llenar un formulario para una comunicación directa con el equipo del festival.

Tareas por realizar:

Implementar el funcionamiento del formulario según el servicio de SendGrid hacia el correo del festival.

Presentar una animación/mensaje de confirmación, para denotar que se ha enviado el mail.

Ilustración 20 Silverio, V. (2021) Historia de Usuario 2.1 – 008: "Formulario de preguntas" (anverso y reverso) [Figura].

Identificador de la Historia	Título de la Historia	Prioridad	Estimación (en días)
2.2 - 009	*Mapas e indicaciones*	Media	4

Historia de Usuario:

Como persona que no conoce la universidad o los teatros,
Quiero una guía de cómo llegar a los eventos
Para no perderme.

Criterios de Aceptación:

Dado que he visualizado un evento que me interesó,
Cuando expanda la información de lugar,
Entonces puedo encontrar un mapa con indicaciones de cómo llegar.

Tareas por realizar:

Determinar las claves que conformarán a los documentos de la colección "Lugares".

Crear, configurar y llenar la información de la colección "Lugares".

Modificar la pantalla de Mapas para mostrar indicaciones y los lugares según la información de la colección.

Configurar la navegación para llevar a la pantalla específica de cada lugar respectivo a cada evento individual.

Ilustración 21 Silverio, V. (2021) Historia de Usuario 2.2 – 009: "Mapas e indicaciones" (anverso y reverso) [Figura].

Identificador de la Historia	Título de la Historia	Prioridad	Estimación (en días)
2.3 - 010	*Notificaciones manuales*	Alta	2

Historia de Usuario:

Como organizador del festival,
Quiero medios
Para hacer llegar información a cada director o corista.

Criterios de Aceptación:

Dado que los usuarios han aceptado los permisos de envío de notificaciones,
Cuando el equipo del festival genere una notificación desde la Firebase Console,
Entonces las notificaciones llegarán a los teléfonos suscritos sin problema.

Tareas por realizar:

Codificar los permisos para ser pedidos cuando se inicie la aplicación por primera vez.

Guardar los tokens de los dispositivos suscritos en una colección en Firestore.

Configurar la herramienta de mensajería de Firebase para poder enviar notificaciones a los teléfonos suscritos.

Verificar que no se envíen notificaciones a teléfonos no suscritos.

Ilustración 22 Silverio, V. (2021) Historia de Usuario 2.3 – 010: "Notificaciones manuales" (anverso y reverso) [Figura].

Identificador de la Historia	Título de la Historia	Prioridad	Estimación (en días)
2.4 - 011	*Notificaciones automáticas*	Media	2

Historia de Usuario:

Como corista internacional,

Quiero tener un medio de comunicación

Para que mis familiares reciban las noticias de mis presentaciones.

Criterios de Aceptación:

Dado que los usuarios han aceptado los permisos de envío de notificaciones,

Cuando suceda un acontecimiento importante dentro del festival,

Entonces se enviarán notificaciones automáticamente a todos los teléfonos suscritos.

Tareas por realizar:

Determinar los casos en los cuales es necesario el envío de notificaciones generales para todos los usuarios suscritos.

Implementar eventos disparados dependiendo de los casos automáticos tomando en cuenta las condiciones establecidas y los tokens de la base de datos.

Ilustración 23 Silverio, V. (2021) Historia de Usuario 2.4 – 011: "Notificaciones automáticas sobre los eventos" (anverso y reverso) [Figura].

Identificador de la Historia	Título de la Historia	Prioridad	Estimación (en días)
3.1 - 012	*Notificaciones automáticas y selectivas*	Media	1

Historia de Usuario:

Como persona interesada en los eventos,

Quiero un recordatorio por evento

Para no perderme de información importante.

Criterios de Aceptación:

Dado que los usuarios han aceptado los permisos de envío de notificaciones,

Cuando se acerque la fecha de acontecimiento de un evento,

Entonces se enviarán notificaciones automáticamente a los teléfonos de los interesados.

Tareas por realizar:

Determinar los casos en los cuales es posible el envío de notificaciones específicas para los usuarios interesados en cierto evento.

Añadir la posibilidad de suscribirse a un evento específico, por medio de un botón en la pantalla del evento individual.

Registrar el token del dispositivo al evento.

Implementar eventos disparados dependiendo de los casos específicos tomando en cuenta las condiciones establecidas y los tokens de la base de datos.

Ilustración 24 Silverio, V. (2021) Historia de Usuario 3.1 – 012: "Notificaciones automáticas y selectivas" (anverso y reverso) [Figura].

Identificador de la Historia	Título de la Historia	Prioridad	Estimación (en días)
3.2 - 013	*Información de Coros*	Media	2

Historia de Usuario:

Como parte del público del festival,

Quiero un repositorio de información sobre los coros

Para consultar datos relevantes de manera centralizada.

Criterios de Aceptación:

Dado que he abierto la app móvil,

Cuando me dirijo a la sección de "Coros",

Entonces se despliega una lista completa de todos los coros que han participado en el festival hasta el momento con su respectiva información (contexto, director, lugar, etc.).

Tareas por realizar:

Llenar los datos respectivos a cada coro en la colección "Coros".

Enlistar todos los coros de la colección "Coros" en la pantalla respectiva.

Configurar la navegación para que cada registro de la lista pueda desplegar la información completa del documento de la base de datos.

Configurar la pantalla de coro individual para mostrar toda la información de una manera entendible para el usuario.

Ilustración 25 Silverio, V. (2021) Historia de Usuario 3.2 – 013: "Información de Coros" (anverso y reverso) [Figura].

Identificador de la Historia	Título de la Historia	Prioridad	Estimación (en días)
3.3 - 014	*Información de entradas*	Media	2
Historia de Usuario:			
Como	interesado en los eventos del festival,		
Quiero	información de cómo y dónde comprar las entradas		
Para	no confundirme.		
Criterios de Aceptación:			
Dado	que he abierto la app móvil,		
Cuando	me dirijo a la sección de "Entradas",		
Entonces	se muestra toda la información de las entradas.		

Tareas por realizar:
Determinar las claves que conformarán a los documentos de la colección "Entradas".
Crear, configurar y llenar la información de la colección "Entradas".
Modificar la pantalla de Entradas para mostrar indicaciones y los lugares de compra según la información de la colección.
Configurar la navegación para llevar a la pantalla específica de cada tipo de entrada respectivo a cada evento individual.

Ilustración 26 Silverio, V. (2021) Historia de Usuario 3.3 – 014: "Información de entradas" (anverso y reverso) [Figura].

Identificador de la Historia	Título de la Historia	Prioridad	Estimación (en días)
3.4 - 015	*Información de Talleres*	Media	2
Historia de Usuario:			
Como	interesado en el festival,		
Quiero	información logística		
Para	conocer respecto a los talleres.		
Criterios de Aceptación:			
Dado	que he abierto la app móvil,		
Cuando	me dirijo a la sección de "Talleres",		
Entonces	se despliega toda la información de los talleres (orador, tema, lugar, fecha, hora, costo, etc.) de forma completa y estructurada.		

Tareas por realizar:
Determinar las claves que conformarán a los documentos de la colección "Talleres".
Crear, configurar y llenar la información de la colección "Talleres".
Enlistar todos los talleres de la colección "Talleres" en la pantalla respectiva.
Configurar la navegación para que cada registro de la lista pueda desplegar la información completa del documento de la base de datos.
Configurar la pantalla de taller individual para mostrar toda la información de una manera entendible para el usuario.

Ilustración 27 Silverio, V. (2021) Historia de Usuario 3.4 – 015: "Información de Talleres" (anverso y reverso) [Figura].

Identificador de la Historia	Título de la Historia	Prioridad	Estimación (en días)
3.5 - 016	*FAQs*	Media	3
Historia de Usuario:			
Como	interesado en el festival,		
Quiero	un lugar donde pueda consultar preguntas frecuentes		
Para	guiarme de mejor manera.		
Criterios de Aceptación:			
Dado	que he abierto la app móvil,		
Cuando	me dirijo a la sección de "Preguntas Frecuentes",		
Entonces	se muestra una lista de preguntas con su respectiva contestación.		

Tareas por realizar:
Determinar las claves que conformarán a los documentos de la colección "FAQs".
Crear, configurar y llenar la información de la colección "FAQs".
Modificar de la pantalla de FAQs para mostrar las preguntas y respuestas respectivas de la colección.

Ilustración 28 Silverio, V. (2021) Historia de Usuario 3.5 – 016: "FAQs" (anverso y reverso) [Figura].

Estas historias de usuario correctamente definidas serán la guía concreta de cómo implementar los requerimientos al sistema y en qué orden hacerlo. Cabe recalcar que cualquier falla en la estimación de las historias no afecta al proceso de desarrollo, debido a la naturaleza enfocada al cambio de la metodología XP. Para corregir los errores de estimación, se agregan las historias de usuario a la siguiente iteración o se aumenta un ciclo de desarrollo más.

3.2.4 Recapitulación

La fase de planificación se enfoca en determinar las tareas necesarias para satisfacer los requerimientos descritos en las historias de usuario. Para eso, se organizan las historias según una prioridad asignada, se selecciona una porción de requisitos que se puedan implementar en el tiempo designado para el proyecto según una estimación otorgada, y se construyen iteraciones. Cada iteración tiene su conjunto de historias de usuario, así como una fecha de entrega de estas. De esta manera, se puede analizar cada requerimiento con el fin de determinar las acciones fundamentales para satisfacer sus criterios de aceptación. Finalmente, para culminar el proceso, se traduce toda la información obtenida en la fase hacia tarjetas individuales para su fácil manejo entre el equipo de desarrollo.

3.3 Conclusión

La característica adaptativa y enfocada al cambio de la metodología XP permite tomar la forma que se necesita en algún proyecto específico. Las fases de Exploración y de Planificación dictan qué necesidades satisfacer con el proyecto de software y de qué manera hacerlo. Sin embargo, estas pautas no tienen un método riguroso para ser construidas, ni pretenden ser normas estrictas para el desarrollo del proyecto.

Al trabajar con XP como metodología central para el proyecto, se espera que el equipo de desarrollo y el cliente enfoquen sus principales esfuerzos en el análisis de necesidades, en lugar de seguir una lista de acciones. Se reconoce que cada proyecto es diferente y la metodología debe acoplarse a estas condiciones.

De la misma forma, el equipo de desarrollo, dejándose guiar por XP, debe poder acomodar el trabajo realizado en cada una de las seis fases según nuevas necesidades del cliente o cambios en las características en las que se desenvuelve el proyecto.

Por lo tanto, se concluye que las necesidades del proyecto son la guía principal de cómo construir las fases de desarrollo, por sobre las recomendaciones que propone una metodología. Por esta razón, es provechoso apoyarse de un marco de referencia que se enfoque en el cambio.

4 CAPÍTULO 3: ITERACIONES PARA PUBLICACIÓN

Esta sección aborda la tercera fase de la metodología Extreme Programming: Iteraciones para Publicación. Esta etapa representa el núcleo del proyecto, repartido en cuatro pilares principales: programación, integración, pruebas y modelamiento. Se analizará cada historia de usuario en base a estas cuatro categorías, presentando la totalidad del proceso de construcción de la aplicación móvil.

4.1 Sobre las Iteraciones

Antes que nada, es importante clarificar qué es una iteración y cuál es su propósito en el contexto de Extreme Programming. Una iteración se entiende como la unión de varias historias de usuario agrupadas y organizadas según su importancia y valor que aporten al negocio. Wells (2001) sugiere que una iteración tenga una duración de una a tres semanas, de manera que, al finalizar cada iteración, el cliente tenga un sistema de software con suficiente nivel de completitud para llevarlo a producción (cuarta fase de la metodología XP).

Cada una de las repeticiones de la Fase de Iteraciones para Publicación representan el esfuerzo principal del proyecto de software, con el propósito de que cada actividad que se realice aporte el mayor valor posible al negocio en un instante de tiempo determinado. Maida y Pacienzia (2015) explican que la priorización en base al valor aportado se basa en la premisa que, si se requiriera reducir el alcance del sistema y acelerar la muerte del proyecto, no se afecten las funcionalidades principales.

En este contexto, Ambler (2002) y Anwer et al. (2017) indican que, para alcanzar el propósito de cada iteración, se realizan cuatro actividades principales repartidas a lo largo del ciclo de vida: modelamiento, pruebas, programación, e integración. A continuación, se presenta un estudio más a fondo de cada una, basado en las reglas que Wells (2006) establece para cada actividad.

Cabe recalcar que hay ciertas características que se aplican transversalmente a las cuatro actividades, como el concepto de Propiedad Colectiva, que implica que todo el equipo tiene la misma importancia en el proyecto: se comparte la responsabilidad en el diseño, en la codificación, realización de pruebas, mantenimiento, entre otras actividades.

4.1.1 Modelamiento

Ambler (2002) indica que el modelamiento es un paso muy importante en el desarrollo de las iteraciones. Es la etapa donde se clarifica cualquier duda respecto al diseño de las interfaces o a cómo debería implementarse una funcionalidad. Es, también, un espacio de discusión con el grupo de trabajo para entender más a fondo la tarea a la que se está afrontando.

El autor también indica que no es necesario tener una documentación formal respecto a los diagramas que se realicen. Uno puede construir diagramas UML estandarizados para explicar algún módulo, o se puede tener un diagrama en un pizarrón para dar una ligera idea del diseño.

En el caso del proyecto, se han empleado dibujos en pizarrón para el diseño de la disposición de las pantallas, así como diagramas influenciados por UML. Cabe resaltar que, en las metodologías ágiles, como XP, se alienta a que el contenido sea más importante que el estándar de diagramación, por lo que no es imperativo que se cumpla rigurosamente con las normas de UML.

Con el fin de facilitar el entendimiento del flujo que deberían tener las pantallas dentro de la aplicación, se ha generado, basándose en UML, el mapa de navegación de las mismas. Seguidamente, la Ilustración 29 muestra tres tipos de pantallas: primero, resaltado en morado, se presenta el núcleo de la navegación que organizándolas en un menú lateral; en amarillo se encuentran las pantallas que redirigirán a otras más; y en verde están aquellas pantallas que no navegan hacia otras.

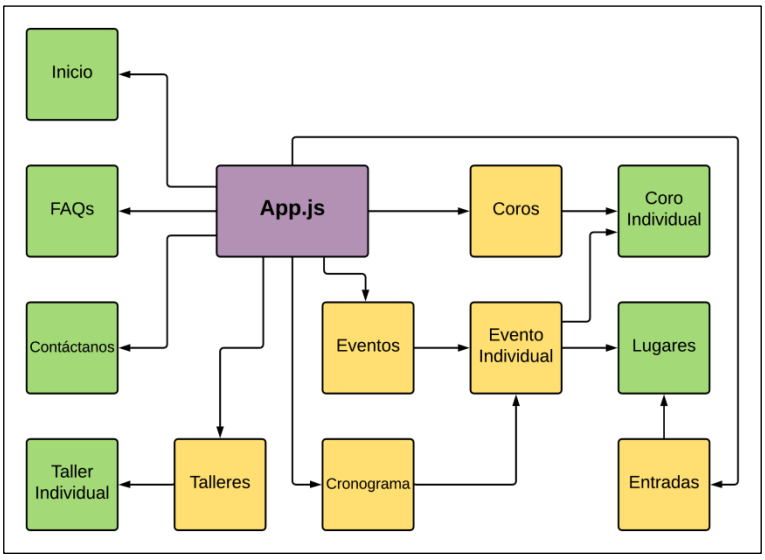


Ilustración 29 Silverio, V. (2021). Diseño de la navegación entre páginas [Figura].

De la misma manera, se han generado modelos para guiarse en cómo construir la interfaz de usuario. Estos modelos se realizaron en pizarras de tiza líquida y surgieron a partir de la reflexión individual realizada antes de programar y diseñar, según el ciclo de vida que se presenta en la sección 4.3.

A continuación, se muestran las pantallas modeladas para la aplicación, que corresponden a requisitos tales como la información de contacto, entradas, eventos, preguntas frecuentes, lugares, notificaciones, y de talleres.

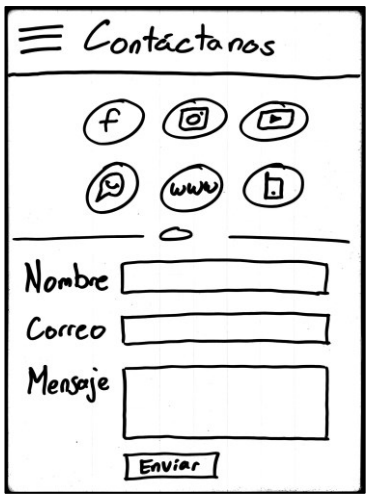


Ilustración 30 Silverio, V. (2021) Modelo realizado en pizarrón referente a la pantalla de Contacto [Figura].

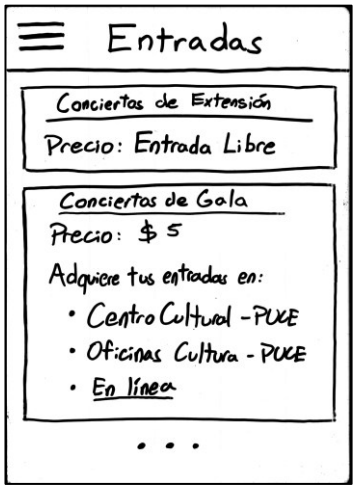


Ilustración 31 Silverio, V. (2021) Modelo realizado en pizarrón referente a la pantalla de Entradas [Figura].

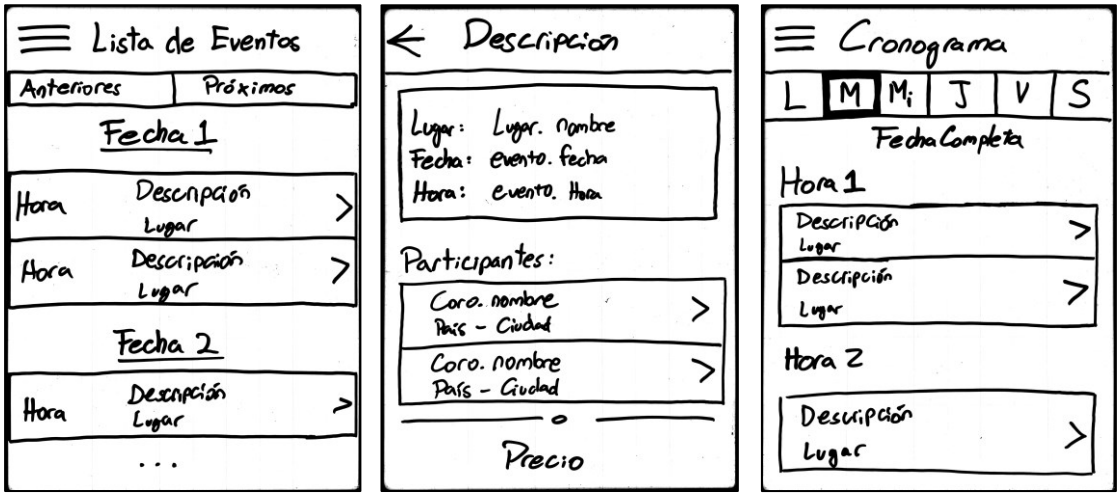


Ilustración 32 Silverio, V. (2021) Modelo realizado en pizarrón referente a las pantallas de Eventos y Cronograma [Figura].

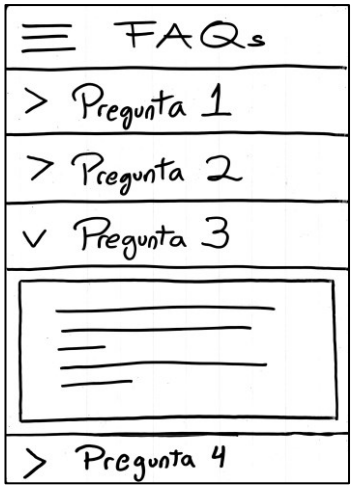


Ilustración 33 Silverio, V. (2021) Modelo realizado en pizarrón referente a la pantalla de Preguntas Frecuentes [Figura].

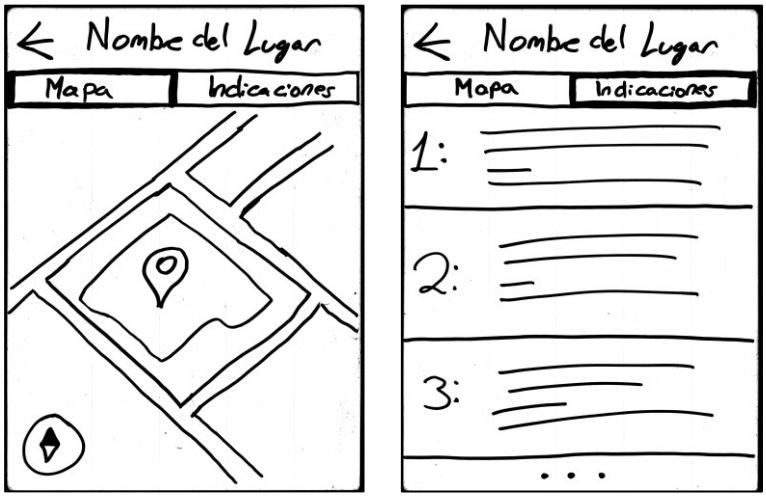


Ilustración 34 Silverio, V. (2021) Modelo realizado en pizarrón referente a las pantallas de Lugares e Indicaciones [Figura].

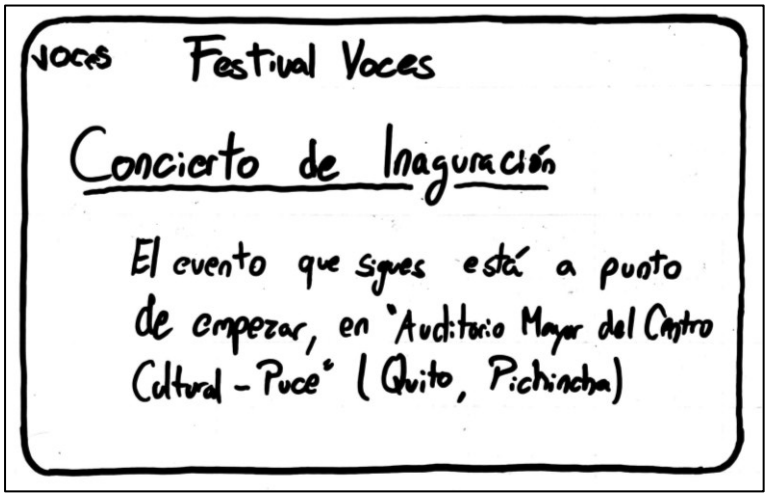


Ilustración 35 Silverio, V. (2021) Modelo realizado en pizarrón referente al diseño de las notificaciones [Figura].

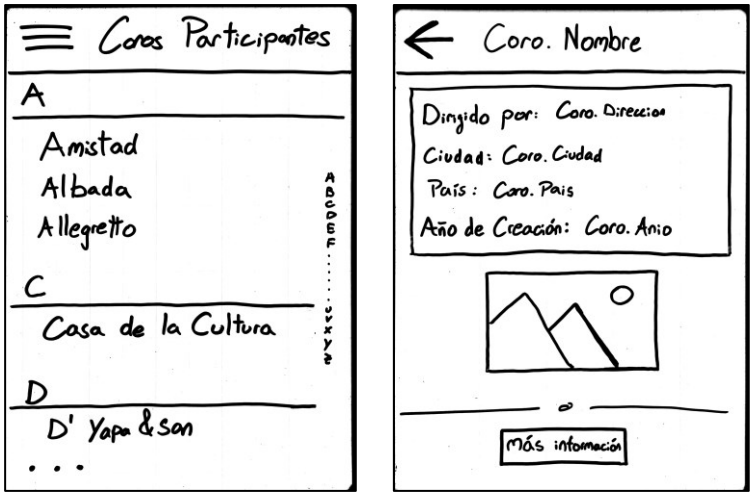


Ilustración 36 Silverio, V. (2021) Modelo realizado en pizarrón referente a las pantallas de Coros Participantes [Figura].

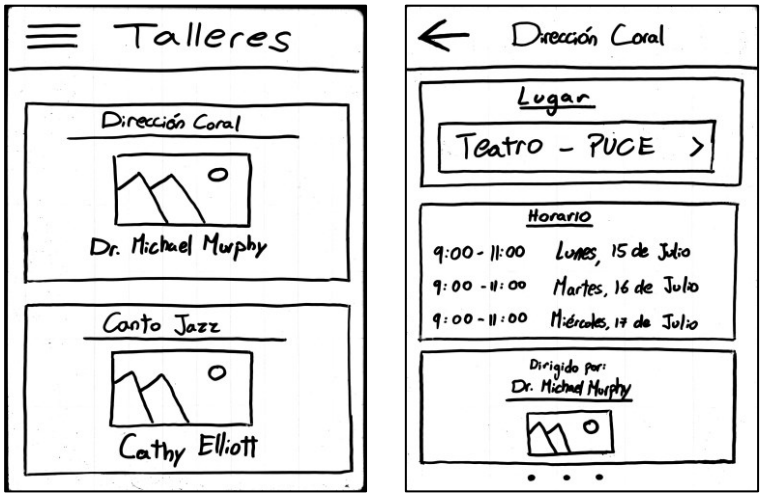


Ilustración 37 Silverio, V. (2021) Modelo realizado en pizarrón referente a las pantallas de Talleres [Figura].

4.1.2 Pruebas

La actividad de pruebas se realiza varias veces a lo largo del desarrollo de una historia de usuario, como explica Wells (2006). Antes de la codificación, es necesario crear las pruebas unitarias [1] relacionadas a cada tarea o pequeña funcionalidad dentro de la implementación general de la historia de usuario. Una vez que se ha generado el mínimo código para satisfacer las condiciones establecidas, se corren las pruebas. Hay que completar todas las pruebas unitarias antes de pasar a la siguiente funcionalidad o tarea.

[1] **Pruebas Unitarias:** Son una manera de poder probar cada unidad de código. Es decir, un grupo de líneas de código que puede ser lógicamente aislado de un sistema: funciones, propiedades, subrutinas. (SmartBear, 2015)

Cuando se han programado correctamente todas las tareas de la historia, el equipo de QA[2] aplica la prueba de aceptación, la cual ya estaba escrita y adjunta a la tarjeta, a fin de determinar si se ha culminado la historia de usuario. Se publica el resultado de la prueba para que, tanto el cliente como el equipo de desarrollo, puedan saber el estado del requerimiento.

Como se indicó, el autor sugiere que antes de comenzar a programar, es recomendado que se creen las pruebas unitarias. Esto ahorra tiempo al programador dado que promueve que se codifique el mínimo código que satisfaga la prueba, evitando que se agreguen funcionalidades que no son necesarias. Esta regla se relaciona con las tareas de las historias de usuario. En el proyecto se ha generado una prueba unitaria para cada actividad especificada en la tarjeta, asegurándose de satisfacer la prueba antes de continuar con la siguiente tarea.

Cabe recalcar que todas las pruebas, unitarias y de aceptación, deben ser automatizadas. Para esto, el autor explica que el equipo de desarrollo debe descargarse o crear un framework de pruebas según el lenguaje de programación que se esté usando.

Para el presente proyecto, se intentó usar la herramienta Jest, un framework para pruebas del lenguaje JavaScript y recomendado por React Native (2021). El instrumento de pruebas permite crear suites de pruebas[3], así como configurar las pruebas unitarias dentro de cada suite. De la misma manera, es posible realizar mocking[4] de partes del sistema para moldear correctamente las dependencias de cada prueba unitaria.

Sin embargo, para el desarrollo de esta aplicación, no era óptimo utilizar Jest en las pruebas respectivas. Nótese que la metodología XP recomienda realizar pruebas automáticas como consecuencia de un trabajo en grupos, de modo que, al cambiar alguna función o método, se pudiera probar inmediatamente el impacto que han tenido los cambios sin preocuparse de escribir las pruebas a cada instante. Pero, al tratarse de un desarrollo de software individual, no resulta ser productivo el uso de pruebas automáticas, optándose por pruebas manuales en base a capturas de pantalla.

De la misma manera, existen razones paralelas que sustentan la decisión de no hacer uso de Jest en el proyecto. Hay que tener en cuenta que Expo y React Native no son lenguajes de programación como tal, sino frameworks construidos sobre JavaScript, el cual

---

[2] **Quality Assurance (QA):** En español, Aseguramiento de la Calidad, “es definido como un procedimiento para asegurar la calidad de productos de software o servicios proporcionados a los clientes de una organización”. (Guru99, 2021)

[3] **Suites de Pruebas:** “Es una colección de casos de prueba que se agrupan con el propósito de ejecutar pruebas”. (IBM, 2020)

[4] **Mocking (imitación):** “es un proceso usado en las pruebas unitarias cuando la unidad que está siendo probada tiene dependencias externas. El propósito del mocking es aislar y concentrarse en el código que se está probando y no en el comportamiento o estado de las dependencias externas. En el mocking, las dependencias son reemplazadas por objetos controlados cercanamente que simulan el comportamiento de objetos reales.” (Telerik, 2013)

es el nivel donde trabaja Jest. Pese a que esta última se recomienda en proyectos construidos sobre Expo y React Native, existen limitantes en cuanto a la complejidad de una aplicación para poder ser probada. Esto se da por diferencia entre las velocidades de avance de las tecnologías en comparación con la construcción de frameworks de pruebas para dichas tecnologías. Al trabajar con frameworks de vanguardia, es común encontrarse con funcionalidades disponibles en Expo y React Native, pero que no es posible escribir pruebas automáticas con Jest sin encontrarse con errores propios de esta y gastando tiempo excesivo en su creación. En otras palabras, la tecnología es mucho más madura que sus herramientas de pruebas. De este modo, se puede decir que es posible lograr el mismo nivel de aseguramiento de calidad con pruebas manuales y capturas de pantalla, gastando menos tiempo, recurso que es limitado en el desarrollo de una disertación de grado.

También, hay que tener en cuenta el uso de la arquitectura serverless sobre la presente aplicación. Esto conlleva a que gran parte de la lógica de la aplicación sea en base a dependencias externas, complicando mucho la tarea de mocking. Un ejemplo es la base de datos configurada en Google Firestore. Las herramientas de imitación de Firestore únicamente evitan que Jest busque dependencias externas, pero no es posible realizar pruebas "End-to-End"[5], algo que si se puede lograr en pruebas manuales. Al trabajar con una arquitectura sin servidor, es imperativo verificar el funcionamiento correcto de todas las partes del sistema y la integración con las dependencias externas, haciendo que no sea muy productivo gastar tiempo en pruebas automatizadas que no soportan todas las funcionalidades con las que se está trabajando.

Caben destacar, por esta razón, ciertos tipos de pruebas imprescindibles en los proyectos elaborados bajo el marco de referencia de XP. Estas pruebas son necesarias para que una iteración pueda pasar a fase de producción:

- **Pruebas de aceptación:** Como se ha mencionado, una prueba de aceptación analiza el cumplimiento de una historia de usuario. Cada tarjeta debe tener una prueba de aceptación que pueda indicar que se ha culminado la implementación del requerimiento. En el caso de la aplicación móvil, se ha verificado que se cumpla la prueba de aceptación antes de pasar a la siguiente historia.
- **Pruebas de Sistema:** También llamadas pruebas de Caja Negra, las pruebas de sistema son aquellas que "validan el funcionamiento completo e integrado de un producto de software", como se indica en Guru99 (System Testing, 2021). Se tratan de pruebas desde el punto de vista del usuario, donde se evalúan funcionalidades

[5] **End-to-end (E2E):** Traducido al español como extremo a extremo, se define como un proceso "que incluye todo lo que es necesario para todas las partes de una red computacional para ser conectada y que funcione conjuntamente". (Cambridge Dictionary, 2020)

End-to-End. En el caso del proyecto, junto con las pruebas de aceptación, al probar la implementación de alguna característica del software, se realizaba desde el punto de vista del cliente hasta el último componente necesario en el sistema, asegurándose que las funcionalidades hayan cohesionado correctamente.

- **Pruebas de Carga:** En Guru99 (Load Testing, 2021) se explica que una prueba de carga determina el manejo del sistema frente a conexiones de varios usuarios. Se realizan estas pruebas para resolver posibles cuellos de botella y asegurar un funcionamiento fluido. Para el caso de la presente aplicación, se apoyó en las ventajas que trae una arquitectura sin servidor, estudiada en el apartado 3.1.2.1. Al trabajar con los servicios de una base de datos en la nube (Firestore), Google se asegura de que las consultas se realicen sin ser afectadas por el número de usuarios conectados.
- **Pruebas de Instalación:** Pruebas realizadas para "verificar que el software se ha instalado correctamente con todas las características heredadas" (Professional QA, 2020). En el caso del proyecto, el servicio de Google Play aseguraría que la aplicación se instale correctamente en los dispositivos móviles.

4.1.3 Programación

La codificación en XP sigue ciertas reglas específicas, explicadas por Wells (2006). En primer lugar, se menciona la unión con la fase de exploración y planificación determinando que el cliente siempre debe estar disponible para preguntas respecto al software. Dado que el consumidor es quien debió escribir las historias de usuario, puede que las tarjetas no contengan toda la información relevante y sea necesario que el equipo de desarrollo realice preguntas. Sin embargo, como se explica en la sección 3.1.1.3, el equipo de desarrollo debe compensar ciertas tareas del cliente debido a que el proyecto no fue una necesidad imperativa requerida por este último. De esta manera, no fue siempre posible que el consumidor estuviera disponible todo el tiempo.

Asimismo, el código debe regirse a ciertos estándares de programación. Wells (2006) no menciona que sean los estándares de cierto lenguaje de programación o arquitectura específica, sino que incita a que el equipo construya sus propios estándares y los respete.

Tal es también la filosofía de Gaba y Ramachandran (2017), quienes presentan normativas generales a ser seguidas para proyectos relacionados con React Native. Siguiendo estos lineamientos, el desarrollo del presente proyecto se ha basado en las normas de formato integradas en la herramienta de TypeScript Language Features (Características del Lenguaje TypeScript), que forma parte de Visual Studio Code, el editor de código empleado en la construcción de la aplicación. Estas características de formato funcionan también con JavaScript, el lenguaje de programación relacionado con React Native. El

formateador por defecto incluye los estándares básicos del lenguaje y sugiere correcciones en cuanto a la escritura de las variables, colocación de las llaves entre otros.

Para complementar la tarea el formateador de código se ha empleado la herramienta ESLint, sugerida por Gaba y Ramachandran (2017). Se usó para dar un estilo homogéneo a ciertas partes del código. Se revisa, por ejemplo, que todos los archivos tengan una sangría basada en espacios con un valor de 2, que todas las cadenas de texto usen doble comilla en lugar de comilla simple, que todas las líneas terminen siempre con punto y coma, o que todas las funciones tengan un nombre y se defina claramente el tipo de variable de sus parámetros.

Cabe resaltar que no se han empleado estándares estrictos de código dado a que el equipo de desarrollo es conformado por una sola persona, y no es obligatorio mantener los mismos estilos para el entendimiento del código. Sin embargo, se aplica formato al código para organizarlo mejor y que sea comprensible para el programador, o cualquier persona interesada en revisar el código adjunto al trabajo de titulación. De igual manera, al tratarse de un trabajo con una metodología de desarrollo ágil, no se es estricto con ninguna normativa, sino resultan ser sugerencias para ayudar a los grupos de trabajos en la construcción de software.

A pesar de esto, se han seguido ciertas buenas prácticas de programación sobre el framework de React Native. Estas prácticas incluyen el idioma del código y de los comentarios, lineamientos en el nombrado de variables y de archivos, y la posición de la declaración de las variables.

De igual forma que esta disertación de grado, el código fue escrito en español. Las variables, comentarios, nombres de archivos y funciones fueron todas escritas en español. Sin embargo, existen algunas excepciones donde se tuvo que escribir en inglés por el bien del entendimiento del código.

Por ejemplo, palabras como “get” (obtener) o “set” (establecer) son prefijos usados en el nombramiento de funciones o métodos donde su uso implica la acción de obtener información de algún repositorio de datos o de establecer cierto valor, respectivamente. Estas palabras, en inglés, son entendidas independientemente del lenguaje de programación, o del país donde se esté desarrollando.

Asimismo, se nombraron variables en inglés en casos en los cuales estas están íntimamente relacionadas con alguna biblioteca o Hook[6] de React Native. Por ejemplo, se usan las palabras “navigation” (navegación) o “route” (ruta), ya que están relacionadas con los Hooks de useNavigation() y useRoute(), respectivamente. A pesar de que se pueden

[6] **Hooks:** “son funciones integradas que permiten a los desarrolladores de React usar métodos de estado y ciclo de vida dentro de los componentes funcionales, también trabajan junto con el código existente, por lo que se pueden adoptar fácilmente en el código” (Akintayo, 2020)

nombrar esas variables en español, se mantuvo el nombre en inglés para evitar confusiones en quienes quieran revisar el código.

Tal y como lo recomienda Jabbar (2021), el nombramiento de variables y de archivos en React Native debe ser diferente. En primer lugar, a los archivos se los nombra con el uso de PascalCase, es decir, se unen todas las palabras en una sola y se las diferencia poniendo en mayúscula a la primera letra de cada palabra. Por ejemplo, “CoroIndividual” o “ListaPorHora”. Por el contrario, a las variables y funciones se las nombra por el uso de camelCase, similar al PascalCase, pero la primera letra de toda la palabra es minúscula. Por ejemplo, “fechaConHora” o “consultarPorIdGeneral”.

Por otro lado, pese a que en React Native, o en el paradigma web y móvil en general, no es de mucha importancia, el equipo de desarrollo ha establecido que cualquier declaración de las variables debe ser indicada al principio del fragmento de código, sea este un archivo, un método, una función o un bucle. Esto permite que el programador pueda guiarse mejor sobre qué variables están contenidas en el fragmento de código, aun cuando ESLint ayude resaltando funciones y variables no utilizadas.

Una característica muy importante dentro de la actividad de programación es el agrupamiento de los desarrolladores en parejas de trabajo. Wells (2006) explica que se trata de una regla de XP que permite mejorar el aseguramiento de la calidad del software cuando dos programadores se concentran en la construcción de una aplicación en una sola computadora, intercambiándose el teclado y el ratón. El autor advierte que no se trata de una relación maestro-alumno, sino una relación entre iguales (principio de Propiedad Colectiva).

No obstante, en referencia al proyecto, tal y como se concluyó al final del Capítulo 1: Determinación de la Metodología de Desarrollo de Software, ninguna metodología analizada se enfoca en el desarrollo de software individual, por lo que es necesario acoplar las actividades de los marcos de referencia a razón de que el equipo de desarrollo consta de una sola persona. Por esta razón, aquellas reglas de la actividad de programación que impliquen más de una persona deberán ser adaptadas.

4.1.4 Integración

Finalmente, la cuarta actividad se refiere a la integración de los cambios y nuevas funcionalidades al resto del código. Wells (2006) describe las características referentes a la integración en un proyecto de software desarrollado bajo XP.

En primer lugar, basándose en el concepto de Propiedad Colectiva, todas las parejas de programación tienen la misma responsabilidad sobre la integración de sus códigos. El autor explica que los equipos de trabajo se deben acostumbrar a la integración continua, es decir, subir cambios de pocas líneas de código varias veces, en lugar de acumular los avances para integrarlos en escasas ocasiones a lo largo de la vida del proyecto. La

integración continua exige que todas las parejas de programación tengan la última versión del proyecto, evitando trabajar en código desactualizado, algo que complica las integraciones de código. Así, como indica el autor, una pareja de trabajo no debe quedarse con los cambios más de un día.

Sin embargo, es preciso seguir siendo cautelosos en cuanto a los cambios en el repositorio. Se sugiere que las integraciones continuas las haga una pareja por vez. Primero, cada grupo de trabajo se encarga de contar con la última versión y de incorporar sus cambios a esta. Luego, se toman turnos para utilizar una computadora dedicada a la integración de código. Con esto, indica Wells (2009), se logra tener un trabajo paralelo entre todas las parejas de desarrollo, pero un solo hilo de integración, evitando la confusión de versiones.

La herramienta enfocada al manejo de repositorios de código utilizada para alojar las versiones del proyecto es GitHub. Se han aplicado las reglas de XP sobre la integración de código con ayuda del repositorio web. Para ello, se crearon dos ramas[7]: la principal ("master"), que contendría el avance real del proyecto, y la secundaria ("VSilverio"), que sirve para añadir todos los cambios temporales sin riesgo a estropear el proyecto. De esta manera, si la consistencia del proyecto llegara a comprometerse con algún cambio en la rama secundaria, el avance del proyecto se mantiene intacto.

Del mismo modo, para cumplir con la integración continua, se han realizado "commits"[8] diarios del avance de las tareas por historia hacia la rama secundaria. Una vez que se culminaba una historia de usuario, se integraba el avance de la rama secundaria a la principal, por medio de revisiones de "pull requests"[9]. Esta revisión verifica si existen conflictos en la integración de los cambios. Cabe destacar que el proceso de commits diarios y de integración a la rama principal es igual para todas las historias de usuario implementadas en el proyecto.

Sin embargo, es importante indicar que el concepto de integración por turnos no tiene validez en el presente proyecto: al contar con una sola persona en el desarrollo, se tuvo siempre un único hilo de integración de los cambios.

La Ilustración 38 representa un ejemplo de integración continua, mostrando la cantidad de "commits" realizados por día durante la primera semana de desarrollo. Nótese que la antigüedad de los cambios no era mayor a un día.

---

[7] **Branch (rama):** "Es una forma de mantener separadas las líneas de desarrollo" (The Pilcrow, 2015)
[8] **Commit (compromiso):** "Fragmentos lógicos de cambios que se guardan en secuencia, formando un historial que luego puede revisar" (The Pilcrow, 2015)
[9] **Pull Request (Solicitud de extracción):** "Permiten informar a otros usuarios sobre los cambios que se ha enviado a una rama en un repositorio en GitHub. Una vez que se abre una Pull Request, se puede discutir y revisar los cambios potenciales con los colaboradores y agregar confirmaciones de seguimiento antes de que sus cambios se fusionen en la rama base." (GitHub, 2012)

Ilustración 38 GitHub (2021) Commits realizados por cada día de la primera semana del proyecto. Gráfico generado por GitHub Insights [Figura]. Recuperado de: https://github.com/VictronJosepe/Festival/graphs/commit-activity

4.2 Relación entre las pruebas y la programación

Como se estudió en la sección anterior, a partir de las cuatro actividades principales de cada ciclo (modelamiento, pruebas, programación e integración), se construyó el software satisfaciendo las 16 historias de usuario determinadas.

El ciclo que se empleó fue tomar una por una las tareas de cada historia de cada iteración, modelar si fuera necesario, determinar las pruebas, programar en base a las condiciones establecidas, probar el código escrito según cada prueba unitaria, integrar los cambios a diario y repetir el proceso para cada tarea. Sin embargo, en caso de el fallo en la ejecución de una prueba, se determinan nuevas pruebas o se editan las existentes para corregir el código y volver a probar hasta que se cumpla el requisito necesario. Una vez que se completaran todas las tareas de una historia, satisfaciendo las respectivas pruebas, se pasó a la siguiente tarjeta hasta terminar la iteración.

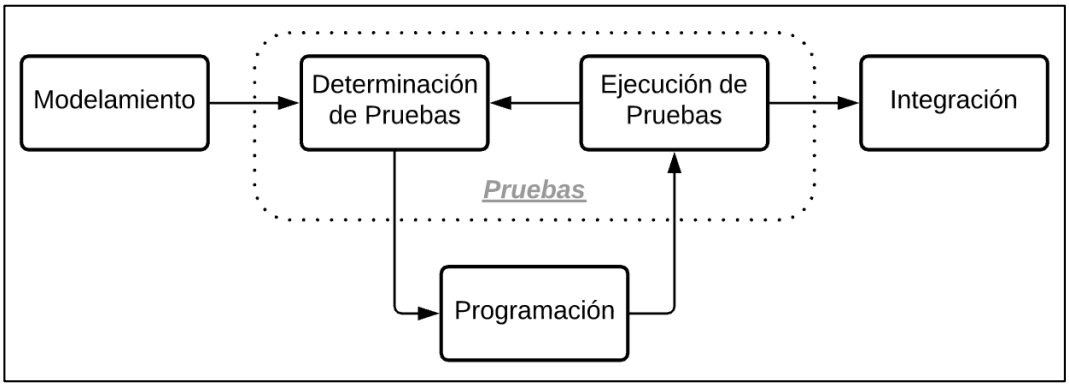


Ilustración 39 Silverio, V. (2021) Ciclo a seguir de cada tarea [Figura].

Como se puede apreciar en la Ilustración 39, en cada tarea se debe pasar por la actividad de pruebas al menos dos veces, primero para definirla y luego para ejecutarla. En este contexto, es pertinente destacar la íntima relación entre las pruebas y la programación, siendo éste el enlace que ha permitido un correcto crecimiento del proyecto. En otras palabras, la conexión entre las dos actividades es el corazón de la iteración.

Por este motivo, es pertinente estudiar las tareas más importantes del proyecto, analizando el vínculo entre las pruebas y la programación en cada una de ellas. Para esto, de un total de 55 tareas, tomadas a partir de las 16 historias de usuario, se han agrupado en seis categorías principales: navegación, presentación de datos, contacto, lugares, notificaciones remotas y notificaciones locales. Cada clase reúne a tareas que sean semejantes en cuanto a programación y pruebas, ya que, se han seguido pasos similares para implementarlas.

4.2.1 Navegación

El grupo de Navegación corresponde a todas aquellas tareas a lo largo del proyecto que se relacionaban con el paso de una pantalla a otra, o la renderización de los menús. El nombre viene dado por el Hook propio de React Native useNavigation().

La navegación permite dar una correcta estructura a la aplicación móvil en la cual un usuario debería poder moverse libremente. Se han usado tres tipos de navegadores en el proyecto: de cajón, de pila y de pestañas. Cada uno de estos tiene una implementación distinta, pero se usan los mismos conceptos y se parten de las mismas pruebas definidas.

Cabe destacar que la navegación en React Native gestiona los encabezados de las pantallas. Es decir, aquellos íconos que representan menú, opciones, o ir a la pantalla anterior son gestionados por un navegador. Tener esto en cuenta permite saber dónde codificar un requisito para su correcta implementación.

4.2.1.1 Navegador de Cajón (Drawer Navigator)

El navegador de cajón es el principal componente de la aplicación móvil. Todo lo que se renderizará estará dentro del Drawer Navigator, incluidos los demás navegadores. Se le llama “de cajón” por las animaciones que están relacionadas con él. Viene con un menú lateral integrado que al abrirse y cerrarse parece un cajón en el borde de la pantalla.

Antes de implementar el navegador se establecieron las pruebas. Estas cumplían con ser individuales, concisas y atómicas. Ejemplos de pruebas en esta categoría eran: ¿Se puede ver y abrir el menú lateral con todos los elementos?, ¿Al presionar, se dirige a la pantalla correcta?, ¿Al dirigirme a cualquier pantalla, existe un ícono para que vuelva a abrir el menú?

A partir de estas pruebas, se generó el mínimo código que las satisfaga. Se envolvió a la aplicación entera en un componente llamado NavigationContainer. Este permitiría moverse libremente entre páginas y renderizar menús, pestañas y demás elementos. Dentro se pondría un componente Drawer.Navigator, que tendrá de componentes hijos todas las pantallas del menú. Para esto se debe usar un Drawer.Screen:

```
<NavigationContainer>
  <Drawer.Navigator>
    <Drawer.Screen />
  </Drawer.Navigator>
</NavigationContainer>
```

Al agregar la propiedad de componente al Drawer.Screen se puede enlazar al elemento del menú con una pantalla desarrollada. Esto permite que, al ingresar a la nueva página, existan los componentes necesarios de la pantalla: encabezado (con título, íconos y estilo) y el cuerpo.

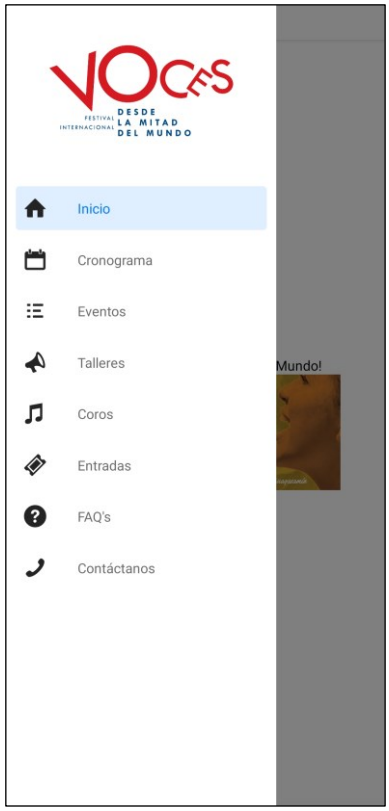


Ilustración 40 Silverio, V. (2021) Satisfacción de las pruebas del navegador de cajón [Figura].

4.2.1.2 Navegador de Pila (Stack Navigator)

El navegador de pila, llamado así por la estructura de datos[10], es el segundo elemento más importante dentro de la presente aplicación móvil. Esto permite generar un conjunto de páginas apiladas una encima de otra, de modo que, al querer volver a la página anterior, solo se debe retirar la pantalla actual de la pila. Tiene propiedades similares a un navegador de cajón, pero no tiene un menú incluido.

Las pruebas en esta sección corresponden, también, a cómo debe comportarse la navegación: ¿Al dirigirme de la pantalla A, hacia la pantalla B, el ícono de "volver" en B me dirige de nuevo hacia A?, ¿La información enviada de la pantalla A hacia la pantalla B se mantiene consistente?

En cuanto a la implementación del navegador de pila, al estar dentro del componente de cajón, ya no se debe envolver en un NavigationContainer. Por lo demás, se comporta de la misma forma que el navegador anterior: un componente Stack.Navigator cuyos hijos son los componentes Stack.Screen, que representan cada una de las pantallas hacia donde se va a navegar.

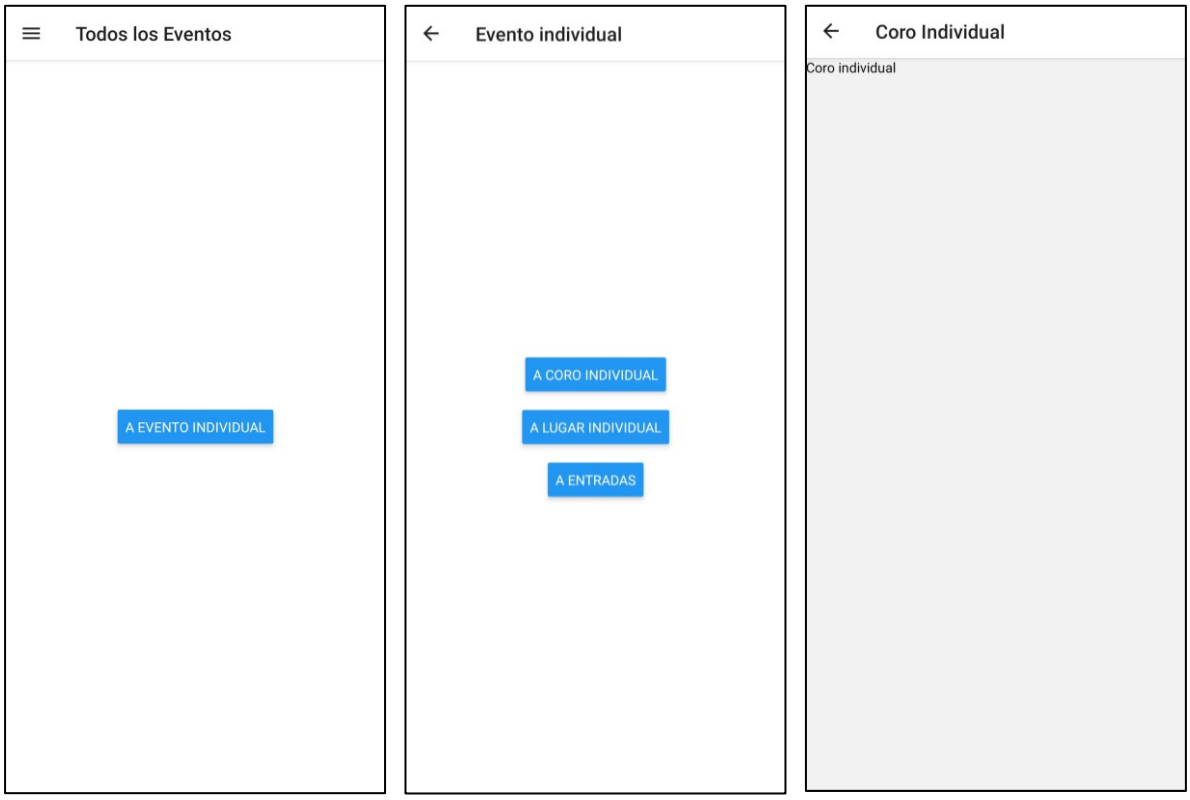


Ilustración 41 Silverio, V. (2021) Satisfacción de las pruebas del navegador de pila [Figura].

[10] **Pila (estructura de datos):** "Una colección de elementos en la que solo se puede eliminar el elemento agregado más recientemente. El último elemento agregado está en la parte superior." (Black, 2019)

4.2.1.3 Navegador de Pestañas (Tab Navigator)

El navegador de pestañas permite presentar varias pantallas en una sola. No se necesita cargar los datos cada vez que se pasa de una pestaña a otra, la información ya está en memoria. Su comportamiento, propiedades, implementación y pruebas son similares a los otros dos tipos de navegación. Tampoco requiere de una NavigationContainer, al ya ser envuelto por el navegador de cajón. Se ha usado este tipo de navegador en el cronograma, en la pantalla de lugares, en la lista de eventos.

En cuanto a las pruebas de este navegador, se ha establecido: ¿Al moverme de la pestaña A hacia la pestaña B se cargan de nuevo los datos?, ¿Toda la información se mantiene independiente entre pestañas?

De la misma manera que en los anteriores navegadores, se utiliza un componente Tab.Navigator, cuyos componentes hijos deben ser los Tab.Screen, para mostrar el contenido de cada pestaña.

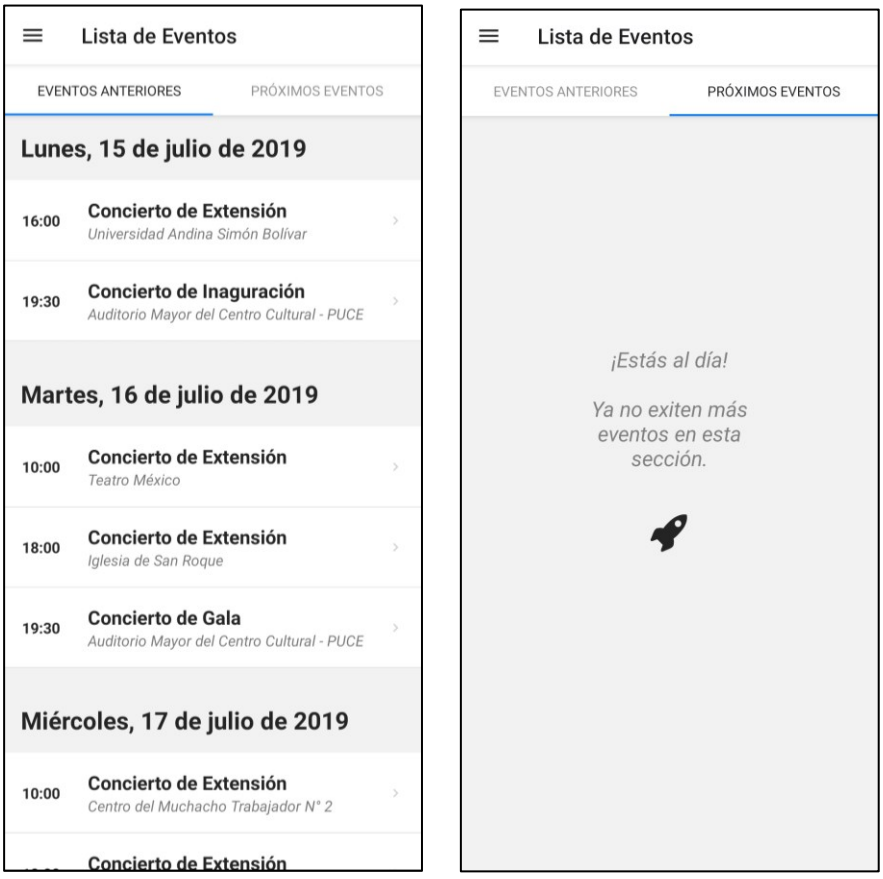


Ilustración 42 Silverio, V. (2021) Satisfacción de las pruebas del navegador de pestañas [Figura].

4.2.2 Presentación de Datos

Las tareas relacionadas con la categoría de presentación de datos son todas aquellas que implementarán un diseño mostrando información de la base de datos. Existen dos sub-agrupaciones para aquellos datos en forma de lista, y aquellos individuales.

Pese a que el diseño de las pantallas pueda cambiar, así como la información presentada en pantalla, los pasos de implementación y las pruebas son similares entre sí. Cabe resaltar que el diseño e información presentada responde a lo planteado en la actividad de modelamiento en la tarea, de modo que las actividades de pruebas y programación sean independientes a esta.

De esta manera, las pruebas planteadas en esta sección corresponden a preguntas tales como: ¿Los datos obtenidos de la base de datos reflejan fielmente a lo ingresado en Google Firestore?, ¿La información es traída desde la base de datos antes de renderizar el contenido?, ¿La información de las colecciones tiene la estructura necesaria para cumplir los requisitos?

En cuanto a la implementación, siempre se seguían los mismos cuatro pasos, sea para datos en lista, o individuales:

1. Declarar una variable donde se guardaría la información traída desde la base de datos.
2. En el Hook de useEffect(), ejecutado antes de la primera renderización, se hacía un llamado a una función asíncrona[11] para acceder a la base de datos, y asignar la información obtenida a la variable definida.
3. En la renderización se consulta si ya existe una definición para la lista. En el caso de que no lo haya, el cuerpo de la pantalla se deja vacío. Si la variable de la lista ya está definida, se procede al siguiente paso.
4. Finalmente, se presenta la información según cada tipo de subcategoría.

4.2.2.1 Listas

La subcategoría de Listas representa todas aquellas tareas relacionadas con pantallas de agrupación de datos, como lista de eventos, de talleres, de coros, preguntas frecuentes, entre otros.

Se pueden generar pruebas individualmente para esta subsección: ¿Existe la misma cantidad de elementos presentados, que elementos en la base de datos?, ¿Toda la

[11] **Computación Asíncrona:** “se refiere a datos que no están sincronizados cuando se envían o reciben, lo que significa que la sincronización no ocurre a intervalos regulares o predeterminados. La comunicación asincrónica implica datos que se pueden transmitir de forma intermitente en lugar de en un flujo constante” (Beal, 2021)

información obtenida de la base de datos está disponible?, ¿Se puede dirigir correctamente de una lista a un elemento individual?

Para la programación, se diferencia la lista de las pantallas individuales porque es necesario iterar los datos antes de presentar los componentes. Para esto, se acude al método .map() del arreglo.

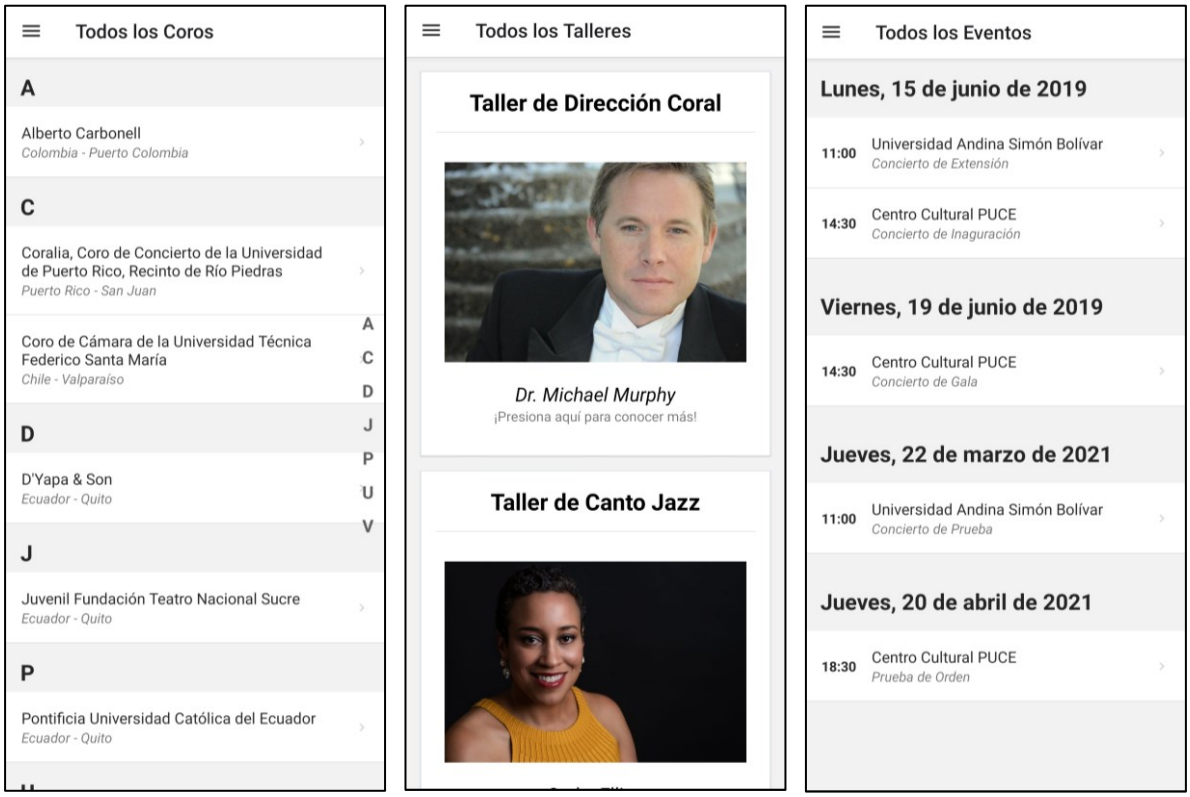


Ilustración 43 Silverio, V. (2021) Satisfacción de las pruebas del grupo de Listas [Figura].

4.2.2.2 Pantalla Individual

La sub-agrupación de pantalla individual recoge todas aquellas tareas relacionadas con información individual que debe ser presentada al usuario. Algunas pantallas de esta subcategoría son: Coro Individual, Evento Individual, Taller Individual.

Las pruebas de esta subsección corresponden a: ¿Información compleja es traída correctamente?, ¿El dato seleccionado desde la lista corresponde al de la pantalla individual?

Por medio del uso de navegación, se pasa de una lista de datos a un dato individual. Se obtiene el identificador del dato, para buscar en Firebase. Se organiza la información según el modelo necesario y se presenta la información en base a los componentes visuales establecidos para la pantalla.

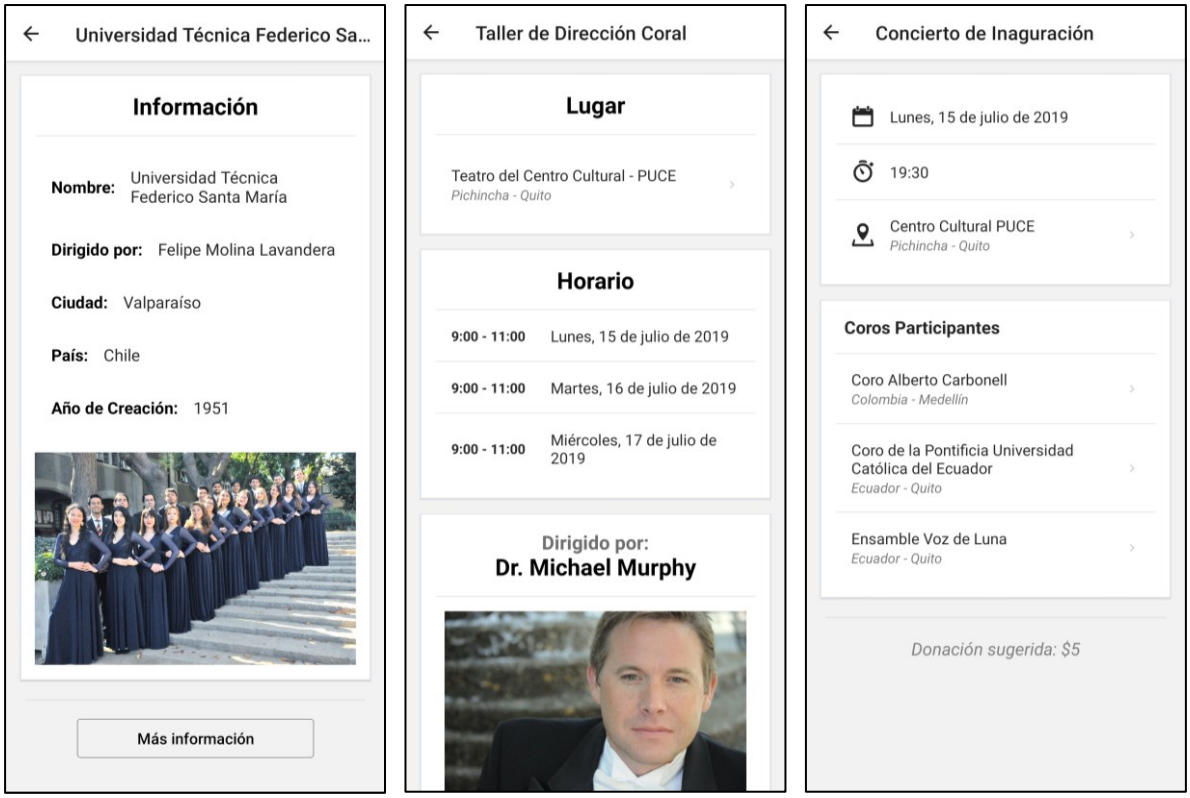


Ilustración 44 Silverio, V. (2021) Satisfacción de las pruebas de pantalla individual [Figura].

4.2.3 Contacto

La sección de contacto corresponde a las tareas relacionadas con la pantalla de contacto. Pese a responder a pruebas y a implementaciones similares que una pantalla individual, existen diferencias específicas que las destacan en su propia categoría.

En primer lugar, las pruebas deben responder a preguntas como: ¿El botón de cada red social redirige correctamente al navegador o aplicación respectiva?, ¿Es posible mandar un formulario sin datos en los campos de texto?, ¿Se notifica correctamente al usuario si se ha enviado su mensaje o ha existido un error?

Para implementar la pantalla de contactos no se trae información de la base de datos, en su lugar, solo se presentan datos cargados en el archivo de configuración del proyecto. Se usó el componente SocialIcon de React Native Elements para renderizar botones circulares con el ícono de cada red social.

Para el formulario, se verificó cada campo individualmente: debían estar llenos antes de enviar, el campo de celular acepta únicamente máximo 10 números, el campo de email se compara con una expresión regular[12], el campo de mensajes debe adaptarse al tamaño

[12] **Expresión Regular:** "Una expresión regular es una cadena de caracteres que es utilizada para describir o encontrar patrones dentro de otros strings, en base al uso de delimitadores y ciertas reglas de sintaxis." (Software Guru, 2008)

dinámicamente, entre otros. Una vez que se hayan superado todas las verificaciones de los campos, Se puede enviar el correo por medio del servicio de SendGrid, llenando correctamente todos los datos del método. Finalmente, se presenta un mensaje al usuario en el caso de que el envío haya fallado o se haya realizado correctamente.

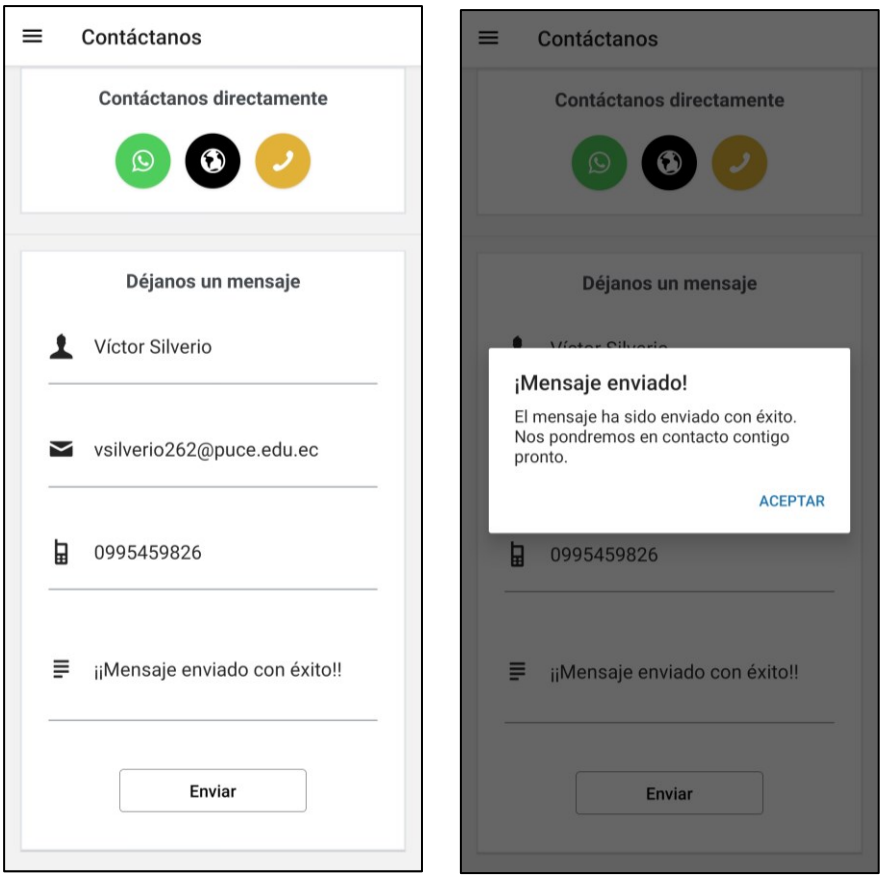


Ilustración 45 Silverio, V. (2021) Satisfacción de las pruebas del grupo de Contacto [Figura].

4.2.4 Lugares

La clase de Lugares agrupa a todas las tareas referentes al manejo de mapas e indicaciones dentro de la app móvil. Estas pantallas requieren un manejo diferente a las pantallas individuales.

Para las pruebas sobre los lugares, se plantearon preguntas como: ¿Se renderiza correctamente un mapa con el cual el usuario puede interactuar?, ¿La zona renderizada del mapa corresponde al escenario solicitado?, ¿El marcador de lugar en el mapa apunta correctamente al lugar almacenado en la base de datos?, ¿Las indicaciones se muestran en una pestaña distinta, en locaciones dentro de la PUCE?

Para implementar el código que satisfaga dichas pruebas se empleó la biblioteca “react-native-maps”, la cual toma las coordenadas que se almacenaron en la base de datos

para renderizar el mapa en la zona indicada. Para obtener las coordenadas, se siguen los mismos pasos de la sección 4.2.2 (Presentación de Datos).

Esto funciona correctamente en un ambiente de desarrollo, pero hay que tomar algunas medidas para que el código funcione en la etapa de producción. Principalmente, lo que se debe hacer es generar una conexión nueva a Google Firebase. En la configuración del servidor, se agrega una nueva aplicación de Android, y se ajustan todos los parámetros a la app móvil del proyecto que se está trabajando. Esto permite poder enlazar la llave de API (API key) de Google Maps con el software. Finalmente, la llave que se obtiene se la registra en el campo “Android.config.googleMaps.apiKey” dentro del archivo “app.json” del proyecto.

Una vez que se hayan realizado todos los pasos, se puede hacer uso de la navegación para pasar de la pantalla de eventos a la pantalla de lugar. En el caso de que la localidad cuente con indicaciones almacenadas en la base de datos, se puede agregar una pestaña con el Navegador de Pestañas y enlistar los eventos de la misma manera que una pantalla de listas (sección 4.2.2.1).

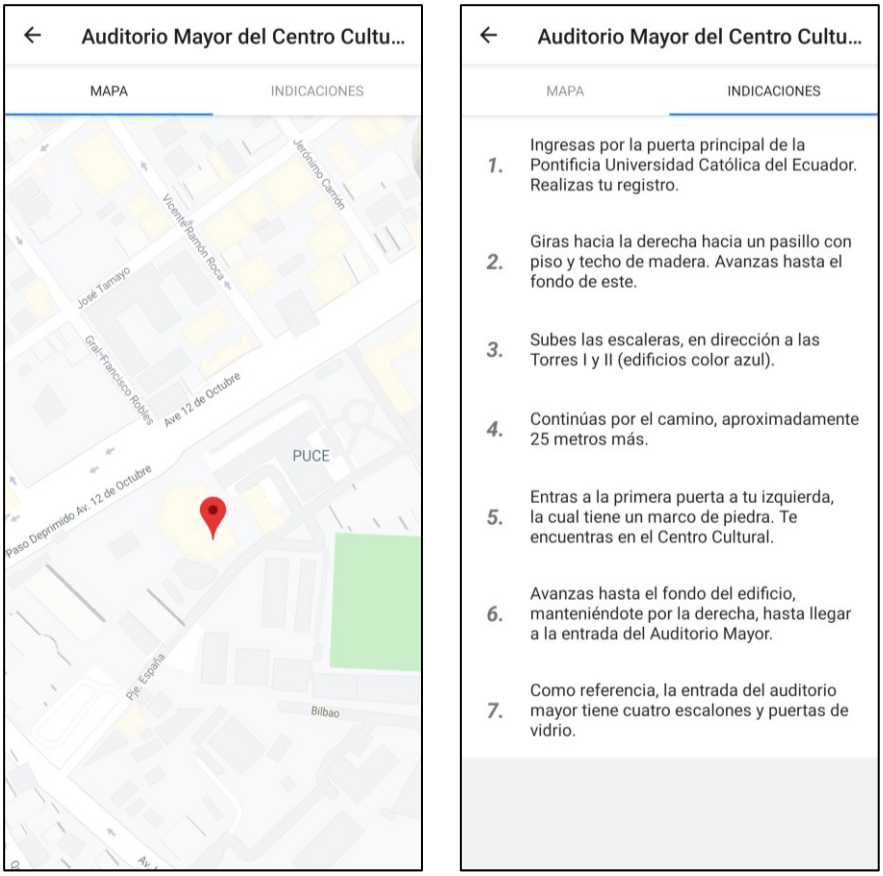


Ilustración 46 Silverio, V. (2021) Satisfacción de las pruebas del grupo de Lugares [Figura].

4.2.5 Notificaciones Remotas

Esta sección recoge las tareas relacionadas con notificaciones enviadas manualmente por el equipo del festival desde el servidor de Firebase. A estas notificaciones también se las conoce como Notificaciones Push[13]. Esta categoría se puede dividir en dos partes: notificaciones instantáneas y notificaciones programadas.

4.2.5.1 Notificaciones Push Instantáneas

Las notificaciones instantáneas son aquellas que llegarán al usuario en el momento en el cual se las ha configurado. Se plantean, primero, pruebas tales como: ¿El software puede recibir notificaciones push?, ¿La notificación configurada llega a los dispositivos?

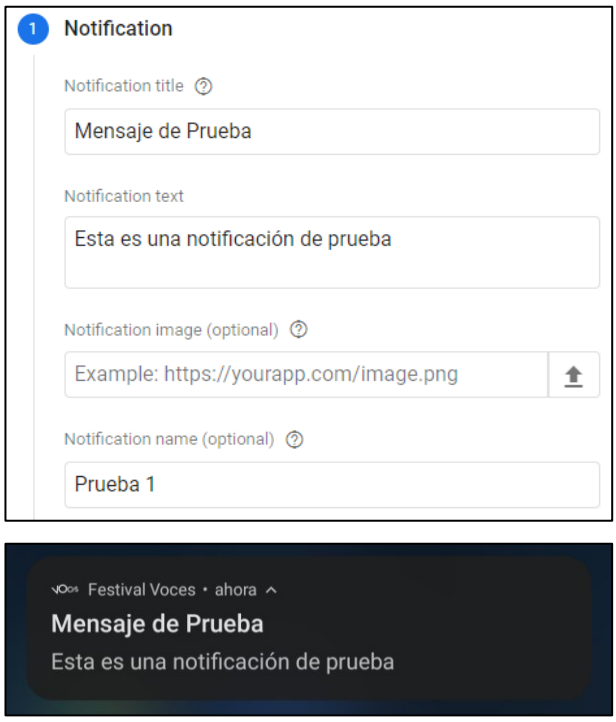


Ilustración 47 Silverio, V. (2021) Satisfacción de las pruebas de las notificaciones push instantáneas [Figura].

Para implementar este tipo de notificaciones, primero se debe obtener permiso para enviar notificaciones del usuario. Si sí se cuenta con el permiso, se configura el canal de notificación de Android para que se pueda mostrar la notificación. Finalmente, se configura el servidor de Firebase siguiendo los siguientes pasos:

[13] **Notificaciones Push:** "Las notificaciones push son mensajes que se pueden enviar directamente al dispositivo móvil de un usuario. Pueden aparecer en una pantalla de bloqueo o en la sección superior de un dispositivo móvil. Un editor de aplicaciones solo puede enviar una notificación de inserción si el usuario tiene su aplicación instalada." (Adjust, 2017)

1. Iniciar sesión en una cuenta con permisos para el envío de notificaciones.
2. Dirigirse a https://console.firebase.google.com/u/0/project/festival-voces/overview, a la sección de “Cloud Messaging”.
3. Hacer clic en Nueva Notificación.
4. En la Zona 1: Llenar los datos de título, texto, imagen y/o nombre.
5. En la Zona 2: Seleccionar como App a “com.festivalvoces.app”.
6. En la Zona 3: Establecer que se enviará la notificación ahora.
7. En la Zona 5: Escribir “default” en el campo “Canal de Notificación de Android”, y prender el sonido de la notificación.
8. Hacer clic en “Revisar” y de nuevo en “Publicar”.

4.2.5.2 Notificaciones Push Programadas

Las notificaciones programadas son aquellas que se guardan para ser enviadas en otro momento en una fecha y hora distinta. Los pasos de implementación y pruebas son iguales. Para poder programar una notificación, únicamente se debe cambiar la configuración del servidor: en la Zona 3, se especifica el horario que se desea que lleguen las notificaciones.

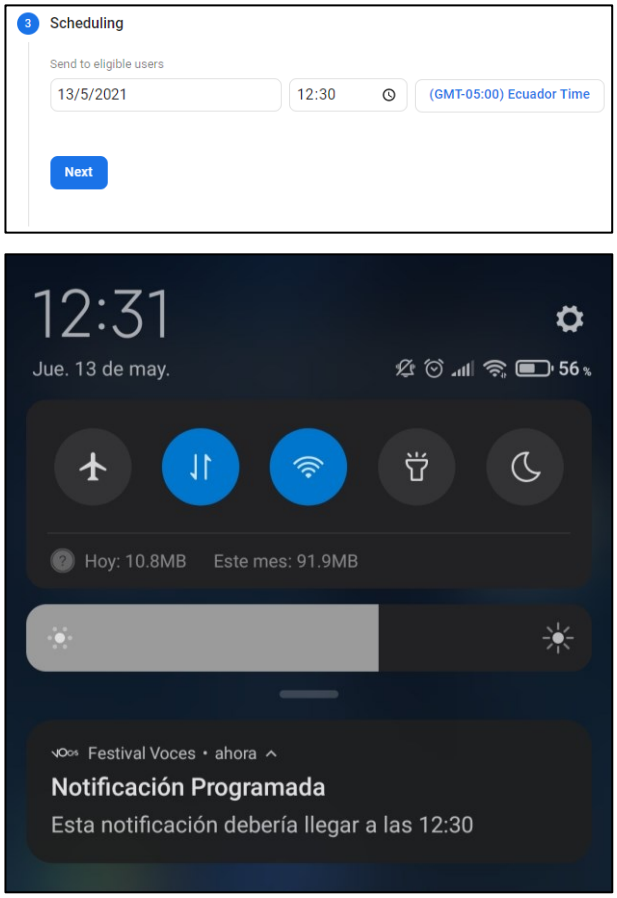


Ilustración 48 Silverio, V. (2021) Satisfacción de las pruebas de las notificaciones push programadas [Figura].

4.2.6 Notificaciones Locales

Las notificaciones locales se diferencian de las notificaciones push en el sentido de que no son gestionadas por un servidor externo y, por tanto, no requieren conexión a internet. Esta sección resume la interacción entre pruebas y programación de notificaciones creadas en el propio dispositivo móvil. Se puede dividir esta relación en dos partes principales: configuración del botón y configuración de la notificación.

4.2.6.1 Botón de activación de la notificación

La creación del botón es la primera parte para que el usuario pueda crear notificaciones para eventos que desee. En relación con las pruebas, se plantea que: ¿Si el botón está desactivado, presionarlo lo activa?, ¿Si el botón está activado, presionarlo lo desactiva?, ¿Si al dejar activado el botón, cuando se reinicia la aplicación, sigue activado?

Para implementar el código que satisfaga estas pruebas, se trabajó con almacenamiento interno del dispositivo: al prender el botón se guarda el estado en el almacenamiento; al apagar el botón se borra dicho estado. De esta manera, no se perdería la configuración al reiniciar la aplicación. También, se utilizó un renderizado condicional, de forma que, si el estado está guardado en el dispositivo, el botón luce de color verde, caso contrario luce gris.

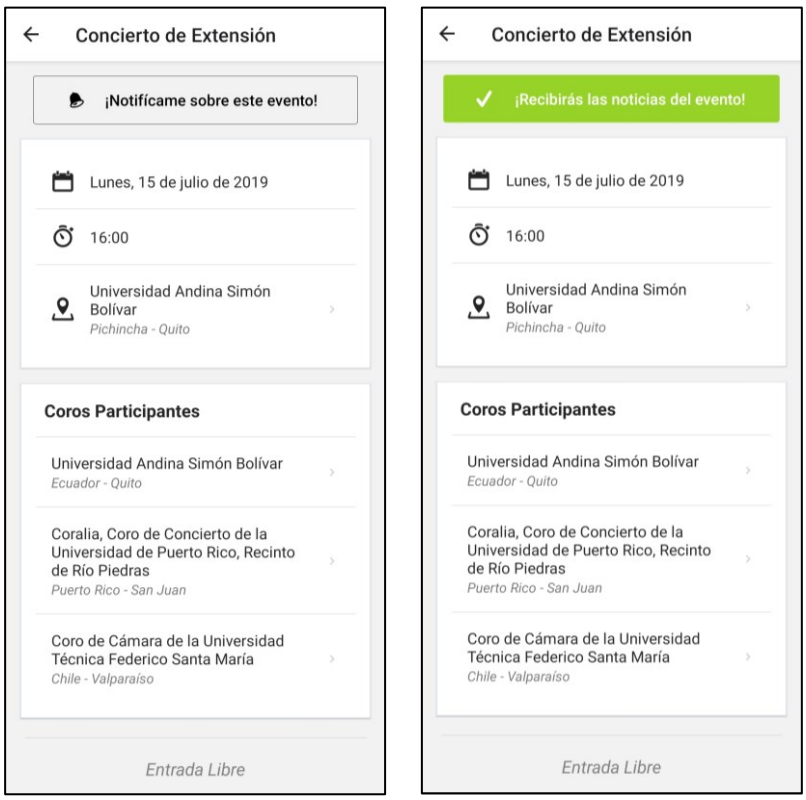


Ilustración 49 Silverio, V. (2021) Satisfacción de las pruebas del botón de las notificaciones locales [Figura].

4.2.6.2 Configuración de la Notificación

Una vez que se tenga configurado el botón, el siguiente paso es crear una notificación local. En este sentido, se plantearon las siguientes pruebas: ¿La notificación local lucirá igual que la notificación push?, ¿La hora y el día de la notificación corresponde exactamente a dos horas antes del evento?, ¿Si el usuario activa el botón y luego lo desactiva, aun recibe la notificación?

Para implementar estos requisitos, se ha usado la biblioteca “expo-notifications”, que permite agendar una notificación desde el código. Para esto, se tiene que configurar el canal de la misma manera que en la sección 4.2.5, y se dispara el evento una vez que se ha presionado el botón. Al guardar el estado en el almacenamiento interno del dispositivo móvil, también se puede guardar el identificador de la notificación creada, de manera que cuando el usuario ya no quiera recibir la notificación, se pueda destruir la notificación y no dejar residuos de información sin utilizar.

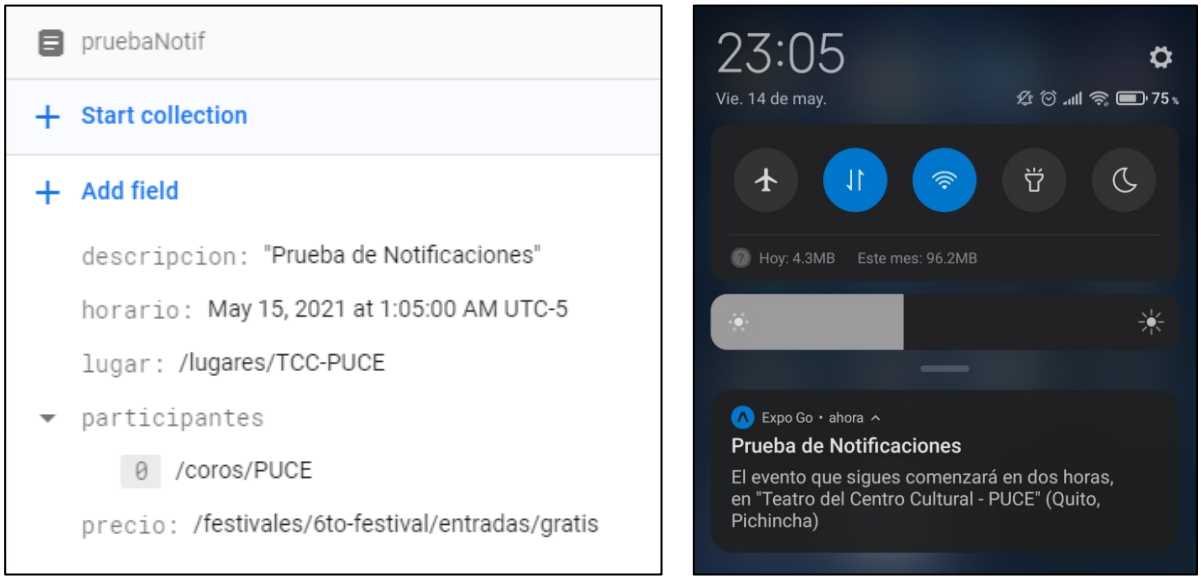


Ilustración 50 Silverio, V. (2021) Satisfacción de las pruebas de las notificaciones locales [Figura].

4.3 Ciclo de Vida de las Iteraciones

En el mismo sentido, Ambler (2002) presenta el ciclo de vida que debe tener una iteración para apegarse al propósito general de la metodología XP, expuesta en la Ilustración 51, la cual contiene aquellas actividades entendidas como las actividades transversales de una iteración. El autor da principal atención a las “Stand Up Meetings” (Reuniones de Pie), debido a que representan el punto de partida de la iteración, así como la convergencia de las actividades como modelamiento, balanceo de carga de trabajo, replanificación, entre otras. Se las llama reuniones de pie, porque deben ser simples y rápidas.

En cada una de estas reuniones, se toma la planificación de iteración, que se realiza al principio de cada repetición, y el trabajo del día anterior, en el caso de que existan pruebas de aceptación fallidas. Se discute el trabajo por realizar, distribuyendo la siguiente tarea o prueba fallida. Sin embargo, si el trabajo sobrepasara la estimación de la repetición, pasan a ser consideradas tareas sin terminar y se corrigen las planificaciones de lanzamiento y de iteración.

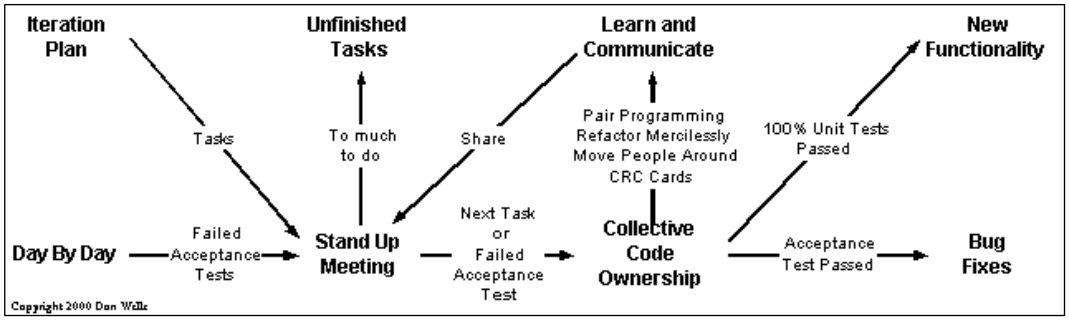


Ilustración 51 Ambler, S. (2002) Ciclo de Vida de una Iteración de XP [Figura]. Recuperado de: http://agilemodeling.com/essays/agileModelingXPLifecycle.htm

Posteriormente, con referencia al trabajo de codificación, se realizan ciclos de programación en pareja cuyos avances, dependiendo del caso, deberán ser notificados al equipo de trabajo en la siguiente Stand Up Meeting. Si el trabajo era enmendar una prueba de aceptación, se integran las correcciones una vez que se haya logrado un resultado satisfactorio. En el caso de que era una nueva tarea por implementar, se comprueba si se han solucionado todas las pruebas unitarias antes de pasar a la siguiente funcionalidad.

Sin embargo, al tratarse de un proyecto cuyo equipo de desarrollo consta de una sola persona, se deben adaptar partes del ciclo de vida de forma que se logre el mismo objetivo, pero con un aspecto individual. En cuanto a las reuniones de pie, enfocadas al diseño de interfaces y módulos, y a la discusión de problemas con las pruebas de aceptación, se reemplazaron con una retrospección al trabajo individual del día anterior. Estas reflexiones seguían siendo dinámicas y simples, pero la discusión ya no era con otros miembros del equipo.

Asimismo, respecto a la programación en parejas, las actividades seguían siendo las mismas, pero ahora las realizaba una sola persona, siguiendo la planificación establecida. En ocasiones, sin embargo, con el fin de obtener una perspectiva diferente a la del desarrollador en el diseño o programación, se preguntaba la opinión de personas externas al proyecto de software, como potenciales usuarios de la aplicación que no tuvieran ninguna idea de cómo funciona el software, o algún representante del lado del cliente.

Este trabajo individual resultó positivo para el proyecto, pues se alentaba que se incluya las tareas de diseño, pruebas e integración en cada una de las historias de usuario, con el fin de cumplir con todos los quehaceres respetando el orden planificado.

Por otro lado, el no contar con una pareja para programar puede resultar en mayor cantidad de errores en el software. Por esta razón, se tuvo que ser muy estricto en cuanto a las pruebas de aceptación y unitarias para solventar la situación.

4.4 Conclusión

De este modo, se puede ver que una iteración está constituida por varias partes fuertemente relacionadas entre sí. Por un lado, existen las actividades principales, a manera de pilares que sostienen el desarrollo de la iteración: modelamiento, pruebas, programación, integración. De la misma forma, también existen actividades transversales que están presentes en todo el ciclo de la iteración: propiedad colectiva, reuniones de pie, etc. Estas últimas pueden ser entendidas como el ciclo de vida de una iteración.

Para llevar a cabo las cuatro actividades principales, se emplearon diferentes instrumentos y técnicas específicas para cada una de ellas, mostrando que la falta de madurez de dichas herramientas puede afectar significativamente el desarrollo del proyecto.

Sin embargo, el contar con una metodología de desarrollo orientada al cambio permite sobrellevar los obstáculos en cada actividad principal. Aun si la planificación no se cumple al programar los requisitos, si no se puede seguir el ciclo de vida exactamente como está descrita por falta de personas en el equipo de desarrollo, o si es más eficiente realizar pruebas manuales en lugar de pruebas automatizadas, el enfoque al cambio permite que el marco de referencia responda a las necesidades específicas del proyecto.

5 CAPÍTULO 4: PRODUCCIÓN, MANTENIMIENTO Y MUERTE DEL PROYECTO

En el presente capítulo se abordan las últimas tres fases de la metodología de desarrollo de Extreme Programming: producción, mantenimiento y muerte. La cuarta etapa se centra en el proceso que sigue una iteración para poder llegar a generar valor para el negocio. La quinta fase se enfoca en cómo la iteración puede pasar a un estado de mantenimiento para añadir nuevas funcionalidades. Finalmente, el sexto paso del marco de referencia de XP trata sobre cómo un proyecto llega a su muerte por diferentes condiciones.

5.1 Fase de Producción

El cuarto paso dentro de la metodología de desarrollo XP es la etapa de producción. El objetivo principal de esta fase es poder generar valor al negocio con el software desarrollado hasta el momento. Según Ambler (2002), la fase de producción es paralela a la fase de iteraciones para publicación. Dicho de otro modo, no se debe esperar hasta culminar todos los grupos de historias de usuario para pasar el software a producción. Típicamente, se entiende que cada iteración debe contener las historias de usuario necesarias para aportar la mayor cantidad de valor al negocio en determinado tiempo, por lo que es importante poder entregar esa versión del software cada vez.

La etapa de producción consta de varias pequeñas entregas que se unen al software principal una vez que se cumplan las condiciones necesarias. Cuando el software esté revisado y desplegado, puede comenzar a aportar valor al negocio según desee el cliente, momento en el cual se pasa a un estado de mantenimiento (quinta fase de XP). De la misma manera, la fase de producción es importante porque, con cada lanzamiento generado, el proyecto es lo suficientemente estable para que se pueda generar documentación a partir de este.

5.1.1 Lanzamientos Pequeños

Al definirse la fase de producción por los lanzamientos realizados, es preciso entender primero qué se entiende por lanzamiento y cómo se lo debería realizar según la metodología de desarrollo empleada. Se recomienda que se realicen pequeños lanzamientos de software cada semana o dos, lo que haría que pueda coincidir con cada iteración planificada (Wells, Make frequent small releases, 2006).

Antes que nada, se podría entender un lanzamiento de software como la versión final de una aplicación, que podría ser público o privado y que representa una nueva generación de cambios de la aplicación con la que se está trabajando (TechTarget Contributor, 2018). Asimismo, Anwer et al. (2017) definen un lanzamiento como “una pequeña parte del software

planificado que implementa algunas necesidades del negocio". De esta manera, se podría entender a un lanzamiento pequeño como cada entrega hecha al cliente, pública o privada, compuesta de pequeños cambios y realizada frecuentemente.

Esta última propiedad de los lanzamientos pequeños empata con la característica de integración constante de una iteración. Se integra el software frecuentemente para que todo el equipo de desarrollo tenga la última versión y sea fácil la construcción cooperativa. Asimismo, se realizan entregas frecuentes para que el cliente esté al tanto de las versiones actualizadas y pueda sugerir cambios lo antes posible.

Anwer et al. (2017) y Ambler (2002) coinciden en que es importante llevar a cabo pruebas antes de poder pasar una iteración a producción. Algunas pruebas que son importantes de realizar en el proyecto son de aceptación, de sistema, carga e instalación, estudiadas en el apartado 4.1.2.

Para el caso del presente proyecto, se realizaron los lanzamientos pequeños al finalizar cada iteración. En las reuniones con el cliente, al final del ciclo, se presentaba el avance respectivo con la aplicación correctamente desplegada, probada e instalada.

5.1.2 Documentación

Como explica Ambler (2002), la etapa de producción es precisa para poder generar la documentación, puesto que, si el proyecto es lo suficientemente estable para generar un lanzamiento, es estable para poder documentar el trabajo realizado. Los cuatro documentos generados en esta sección son manuales enfocados a dos poblaciones especificas: La documentación del sistema y de las operaciones están orientados al equipo técnico de festival; por otro lado, la documentación de soporte y de usuario está dirigida para cualquier persona que utilice la aplicación móvil.

A continuación, se presentan los cuatro tipos de documentación necesaria en la fase de producción, abarcando elementos sugeridos por Ambler (2002).

5.1.2.1 Documentación del Sistema

Introducción

El siguiente escrito representa la documentación del sistema dirigida a el equipo técnico del Festival Voces desde la Mitad del Mundo. Aquí se resume la arquitectura técnica, e información general sobre los requerimientos y sobre el diseño de la aplicación. Fue elaborado por Víctor Silverio en junio del 2021.

Arquitectura Técnica

La arquitectura del sistema está conformada por dos componentes: por el lado de la aplicación se trabaja con MVC, mientras que, para trabajar con servicios externos, se emplea una arquitectura “serverless”.

En primer lugar, en cuanto a MVC, la vista es la ventana por la que el usuario general puede acceder a la información, el controlador se encarga de los servicios dentro de la aplicación, y el modelo se enfoca de la administración de los servicios de Firebase orientados al manejo de datos.

Respecto a la arquitectura serverless, se encuentra la parte de los servicios externos, donde se emplea los siguientes productos para un funcionamiento integral de la aplicación: Cloud Functions, Firebase Cloud Messaging, Cloud Firestore, Cloud Storage, Firebase Console, Firebase Auth, Firebase Rule Manager, SendGrid.

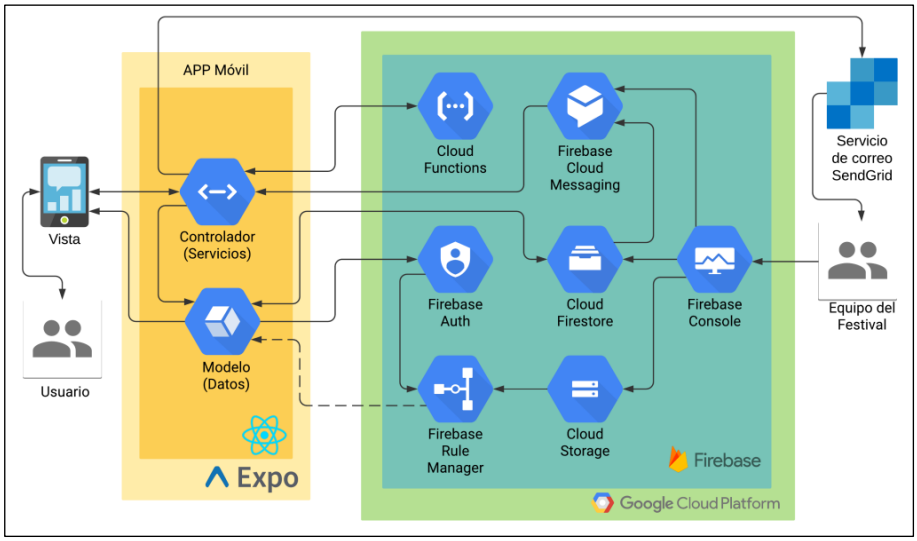


Ilustración 52 Silverio, V. (2021) Arquitectura del sistema [Figura].

Cabe destacar que existen tres partes fundamentales: los actores, el núcleo de la aplicación y los servicios externos. En primera parte, los actores son de dos tipos: el usuario general y el equipo del festival (generadores de contenido, director del festival, equipo técnico,

comité organizador). Los usuarios generales se comunicarán con los servicios únicamente a través de Vista: ahí podrán encontrar todo el contenido multimedia, información de los coros, cronogramas, entre otros elementos. Por otro lado, el equipo del festival, con el fin de agregar información a la base de datos, enviar notificaciones o agregar nuevo contenido multimedia, debe hacerlo por medio del Firebase Console.

Requerimientos de Alto Nivel

#	Requerimiento	Descripción
1	Estructura de la aplicación móvil	Conjunto de pantalla y navegadores entre ellas.
2	Estructura de la información de los eventos	Pantalla de la lista de todos los eventos de la base de datos.
3	Información de eventos	Pantalla que muestra toda la información relevante de un evento específico.
4	Estructura del Cronograma	Lista de los eventos agrupados por fecha y presentados en diferentes pestañas.
5	Cronograma de eventos por festival	Presentación de la información destacada de cada evento resumida en el cronograma.
6	Información de participantes	Visualización de todos los coros participantes dentro de un coro específico.
7	Pantalla de Contacto	Apartado específico para resumir todos los medios para poder comunicarse con el equipo del festival.
8	Formulario de preguntas	Formulario que envía una notificación al correo del festival sobre una pregunta del usuario.
9	Mapas e indicaciones	Página para visualizar un mapa con indicaciones de los lugares de cada evento.
10	Notificaciones manuales	Implementación del receptor del dispositivo móvil para notificaciones enviadas por el servidor.
11	Notificaciones automáticas sobre los eventos	Configuración del servidor para enviar notificaciones programadas a cierta fecha y hora.
12	Notificaciones automáticas y selectivas	Botón para que el usuario pueda agendar una notificación de un evento específico.
13	Información de Coros	Pantalla de la lista de todos los coros participantes de la base de datos, presentando la información específica de cada coro.
14	Información de entradas	Pantalla de la lista de todos los tipos de entradas registrados en la base de datos.
15	Información de Talleres	Pantalla de la lista de todos los talleres de la base de datos, presentando la información específica de cada taller.
16	FAQs	Pantalla de la lista de todas las preguntas frecuentes ingresadas en la base de datos.

Tabla 7 Silverio, V. (2021) Lista de requerimientos de alto nivel [Tabla].

Decisiones Críticas Sobre el Diseño

- Todo encabezado que no refiera a una lista debe representar el nombre o descripción del elemento que se visualiza. Por ejemplo, en lugar de poner “Lugar Individual”, se debe indicar el nombre: “Teatro Sucre”.
- Se deben utilizar íconos en todos los lugares donde sea posible. Si el contexto es ambiguo, es mejor poner una etiqueta en lugar de un ícono. A veces es válido poner ambos: etiqueta e ícono. Por ejemplo, al referirse a una hora, se puede agregar el ícono de un reloj; si hay varias horas en la pantalla, es mejor usar una etiqueta.
- Preferir escala de grises en toda la aplicación. El uso de colores es permitido para las imágenes. Por ejemplo, los encabezados, íconos, etiquetas no deben ser visualizados con ningún color a parte de la escala de grises.
- Evitar el uso de componentes “Card” a menos que, al presionarlos, dirijan a otra pantalla. Puede resultar confuso para el usuario el uso de tarjetas que no dirijan a ninguna pantalla.

Modelo de Diseño

Con el fin de establecer una guía referente a cómo las páginas deben comunicarse entre sí, se ha establecido el modelo de navegación, presentado a continuación. Aquellos elementos en verde son pantallas independientes que no dirigen a ninguna otra página, los elementos amarillos son pantallas que apuntarán a otras pantallas, y el elemento morado es la base de la navegación que apuntará a todas las pantallas.

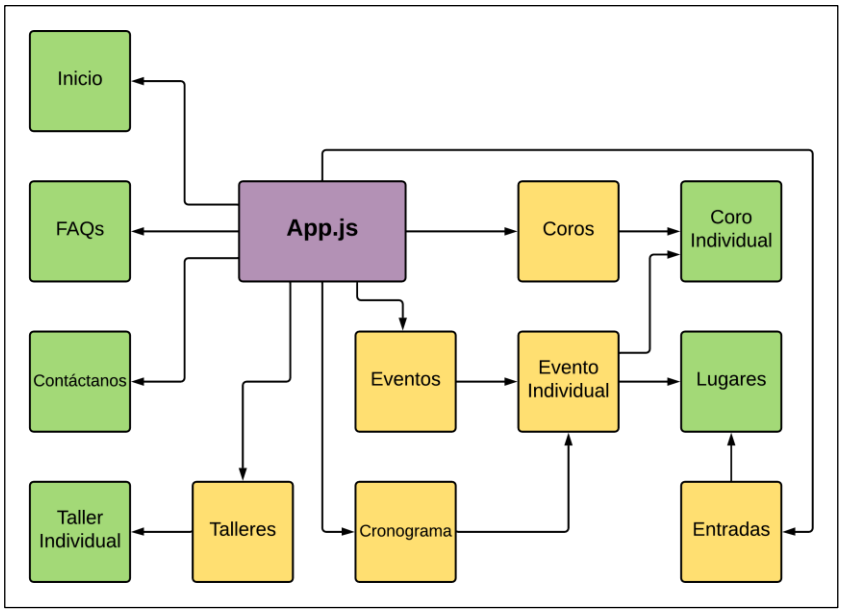


Ilustración 53 Silverio, V. (2021) Diseño de navegación entre páginas [Figura].

5.1.2.2 Documentación de Operaciones

Introducción

El siguiente escrito representa la documentación sobre las operaciones de la aplicación, dirigida a el equipo técnico del Festival Voces desde la Mitad del Mundo. Aquí se resumen las dependencias, las interacciones con sistemas externos, procedimientos de respaldo y puntos de contacto. Fue elaborado por Víctor Silverio en junio del 2021.

Dependencias

Dependencia	Versión	Descripción
@babel/core	^7.9.0	Compilador para JavaScript. Dependencia de desarrollo.
@react-native-async-storage/async-storage	^1.15.4	Se utilizó para manejar el almacenamiento local dentro del dispositivo móvil.
@react-native-community/netinfo	^6.0.0	Dependencia que permite consultar el estado de red del dispositivo móvil cuando la aplicación está en uso.
@react-navigation/drawer	^5.12.5	Se empleó para manejar el menú lateral con su respectiva navegación.
@react-navigation/material-top-tabs	^5.3.15	Se usó para poder agregar pestañas superiores a las pantallas.
@react-navigation/native	^5.9.4	Adaptación de React Navigation para React Native
@react-navigation/stack	^5.14.4	Se utilizó para implementar el navegador entre páginas del mismo grupo.
eslint	^7.26.0	Formateador del código para ayudarse a cumplir con los estándares establecidos. Dependencia de desarrollo.
eslint-plugin-react	^7.23.2	Plugin de ESLint para React. Dependencia de desarrollo.
expo	~40.0.0	SDK de Expo
expo-notifications	~0.8.2	Se usó para manejar el sistema de notificaciones remotas y locales en la aplicación.
expo-permissions	^12.0.1	Se empleó para facilitar el manejo de permisos en la aplicación.
expo-status-bar	~1.0.3	Dependencia que permite el manejo de la barra de estado en los dispositivos móviles.

Dependencia	Versión	Descripción
firebase	^8.3.3	Dependencia para poder conectar con la base de datos, mensajería (notificaciones) y el sistema de almacenamiento remoto en Firebase.
lodash	^4.17.21	Facilita el trabajo con arreglos y objetos de JavaScript.
native-base	^2.15.2	Biblioteca de componentes.
react	16.13.1	Núcleo de desarrollo
react-native	remoto (40.0.1)	Núcleo de desarrollo para aplicaciones móviles.
react-native-auto-height-image	^3.2.4	Dependencia usada para renderizar imágenes cuya altura pueda ser calculada automáticamente
react-native-elements	^3.3.2	Biblioteca de componentes.
react-native-flatlist-alphabet	^1.1.2	Dependencia usada para implementar una lista con un abecedario en el costado de la página para facilitar la navegación.
react-native-gesture-handler	^1.10.3	Manejador de gestos realizados sobre el dispositivo móvil.
react-native-maps	^0.28.0	Se utilizó para poder presentar mapas en pantalla.
react-native-phone-call	^1.0.9	Se empleó para facilitar la implementación de llamadas telefónicas
react-native-reanimated	^2.1.0	Dependencia que complementa el manejo de gestos y animaciones en la aplicación móvil.
react-native-safe-area-context	3.1.9	Asegura que el renderizado de los componentes se realice en una zona visible en la pantalla.
react-native-screens	^3.1.0	Complementa el renderizado de componentes y animaciones en la aplicación móvil.
react-native-sendgrid	^1.0.2	Dependencia para poder enviar correos dentro de la aplicación.
react-native-tab-view	^2.16.0	Complementa el uso de pestañas en las pantallas de la aplicación móvil.
react-native-web	~0.13.12	Componentes de React Native para Web. Preinstalado con el proyecto.

Tabla 8 Silverio, V. (2021) Lista de dependencias de la aplicación móvil [Tabla].

Interacción con la Base de Datos (Firestore)

1. Realizar las importaciones necesarias.

```
import firebase from "firebase";
import "@firebase/firestore";
```

2. Definir la configuración de Firebase según los datos proporcionados por el servidor.

```
var firebaseConfig = { apiKey: "API_KEY",
  authDomain: "festival-voces.firebaseapp.com",
  projectId: "festival-voces",
  storageBucket: "festival-voces.appspot.com",
  messagingSenderId: "SENDER_ID",
  appId: "APP_ID",
  measurementId: "MEASUREMENT_ID" };
```

3. Inicializar Firebase con la configuración establecida.

```
firebase.initializeApp(firebaseConfig);
```

4. Obtener la colección con la se va a trabajar.

```
const corosCollection = firebase.firestore().collection("/coros");
```

5. Acceder a la colección.

```
const informacion = await firebase.corosCollection.get();
```

Interacción con Archivos

1. Realizar las importaciones necesarias.

```
import firebase from "firebase";
import "@firebase/storage";
```

2. Definir la configuración de Firebase según los datos proporcionados por el servidor.

```
var firebaseConfig = { apiKey: "API_KEY",
  authDomain: "festival-voces.firebaseapp.com",
  projectId: "festival-voces",
  storageBucket: "festival-voces.appspot.com",
  messagingSenderId: "SENDER_ID",
  appId: "APP_ID",
  measurementId: "MEASUREMENT_ID" };
```

3. Inicializar Firebase con la configuración establecida.

```
firebase.initializeApp(firebaseConfig);
```

4. Obtener una referencia al repositorio con el que se va a trabajar.

```
const repositorio = firebase.storage().ref();
```

5. Obtener el URL de descarga del archivo del repositorio.

```
const ref = firebase.repositorio.child("Ubicación del Archivo");
await ref.getDownloadURL().then((url) => { setImagenURL(url); });
```

Interacción con el Sistema de Correos Electrónicos (SendGrid)

1. Realizar las importaciones necesarias.

```
import { sendGridEmail } from "react-native-sendgrid";
```

2. Definir la configuración de SendGrid según los datos proporcionados por el servidor.

```
const sendRequest = sendGridEmail(
  "API_KEY",
  "CORREO_DESTINO",
  "CORREO_EMISOR",
  "ASUNTO"
  "CUERPO"
);
```

3. Establecer acciones para la respuesta del servidor.

```
sendRequest.then(() => {
  // Acciones en caso de éxito
}).catch(() => {
  // Acciones en caso de fallo
});
```

Enviar una Notificación con FCM

1. Iniciar sesión en una cuenta con permisos para el envío de notificaciones.
2. Dirigirse a https://console.firebase.google.com/u/0/project/festival-voces/overview, a la sección de “Cloud Messaging”.
3. Hacer clic en Nueva Notificación.
4. En la Zona 1: Llenar los datos de título, texto, imagen y/o nombre.
5. En la Zona 2: Seleccionar como App a “com.festivalvoces.app”.
6. En la Zona 3: Establecer cuándo se quiere enviar la notificación.
7. En la Zona 5: Escribir “default” en el campo “Canal de Notificación de Android”, y prender el sonido de la notificación.
8. Hacer clic en “Revisar” y de nuevo en “Publicar”.

Procedimientos de Respaldo

1. Respaldo del Proyecto

1.1. Respaldar el repositorio.

1.1.1. Dirigirse al repositorio en GitHub.

1.1.2. Copiar el URL de descarga del proyecto.

1.1.3. Abrir un terminal del sistema operativo.

1.1.4. Escribir el comando: “git clone <URL de descarga del proyecto>”.

1.1.5. Ingresar las credenciales para clonar el repositorio.

1.2. Respaldar el avance local del proyecto.

1.2.1. Desde GitHub Desktop, hacer clic en Archivo>Nuevo Repositorio.

1.2.2. Llenar los datos del repositorio como nombre y descripción.

1.2.3. Seleccionar la ruta del proyecto.

1.2.4. Crear el repositorio.

2. Respaldo de la Base de Datos

2.1. Generar Respaldo.

2.1.1. Dirigirse a https://console.cloud.google.com/billing/projects y asegurarse que el proyecto tenga una cuenta de facturación.

2.1.2. Dirigirse a https://console.cloud.google.com/firestore/import-export.

2.1.3. Hacer clic en Exportar.

2.1.4. Seleccionar la opción de exportar la base de datos completa.

2.1.5. Escoger un destino dentro del proyecto donde se almacenará el respaldo.

2.1.6. Aceptar la configuración. Se ha creado el respaldo.

2.2. Restaurar Respaldo.

2.2.1. Dirigirse a https://console.cloud.google.com/billing/projects y asegurarse que el proyecto tenga una cuenta de facturación.

2.2.2. Dirigirse a https://console.cloud.google.com/firestore/import-export.

2.2.3. Hacer clic en Importar.

2.2.4. Seleccionar la ubicación del respaldo anterior.

2.2.5. Aceptar la configuración. Se ha recuperado el respaldo.

Puntos de Contacto

Miembro del Equipo	Método de Contacto	Descripción
Víctor José Silverio Torres	Correo Electrónico	vjose_2007@hotmail.com
	Correo Electrónico Secundario	victronjosepe@gmail.com
	Correo Electrónico Institucional	vsilverio262@puce.edu.ec
	Teléfono celular	+593 99 545 9826

Tabla 9 Silverio, V. (2021) Puntos de contacto del equipo de desarrollo [Tabla]

5.1.2.3 Documentación de Soporte

Introducción

El siguiente escrito representa la documentación respecto al soporte al proyecto de software, dirigida cualquier usuario de la aplicación del Festival Voces desde la Mitad del Mundo. Aquí se presenta una guía ligera sobre cómo resolver inconvenientes que puedan surgir al usar la aplicación, junto con puntos de contacto al equipo de soporte. Fue elaborado por Víctor Silverio en junio del 2021.

Guía de Resolución de Problemas

Problema	Solución
No me llegan las notificaciones de los eventos.	Dirígete a la configuración de tu dispositivo móvil y asegúrate de que las notificaciones estén encendidas.
No puedo visualizar ningún evento. Las pantallas están vacías.	Asegúrate de que tu dispositivo móvil está conectado a internet.
Al consultar la información de un evento, la aplicación se queda cargando interminablemente.	Cierra la aplicación, asegúrate de tener buena conexión de internet y vuélvela a iniciar.
Lleno el formulario de consultas en la pantalla de contacto, pero no se envía mi información.	Puede que el servidor esté colapsado. Espera unos minutos y vuelve a intentarlo.

Tabla 10 Silverio V. (2021) Guía de resolución de problemas. [Tabla]

Puntos de Contacto

Miembro del Equipo	Método de Contacto	Descripción
Víctor José Silverio Torres	Correo Electrónico	vjose_2007@hotmail.com
	Correo Electrónico Secundario	victronjosepe@gmail.com
	Correo Electrónico Institucional	vsilverio262@puce.edu.ec
	Teléfono celular	+593 99 545 9826

Tabla 11 Silverio, V. (2021) Puntos de contacto del equipo de soporte [Tabla]

5.1.2.4 Documentación del Usuario

Introducción

El siguiente escrito representa manual de usuario, dirigida cualquier persona interesada en la aplicación del Festival Voces desde la Mitad del Mundo. Aquí se presenta una guía ligera sobre cómo instalar y cómo utilizar la aplicación móvil. Fue elaborado por Víctor Silverio en junio del 2021.

Manual de Uso Básico

1. ¿Cómo ver la información de un evento?

1.1. Abrir el menú lateral.

1.2. Dirigirse a Cronograma o a Eventos.

1.3. Escoger un evento de entre la lista mostrada.

1.4. Dentro se podrá encontrar la información completa de fecha, hora, lugar, participantes y precio. También se puede agendar una notificación del evento presionando el botón de "¡Notifícame sobre este evento!".

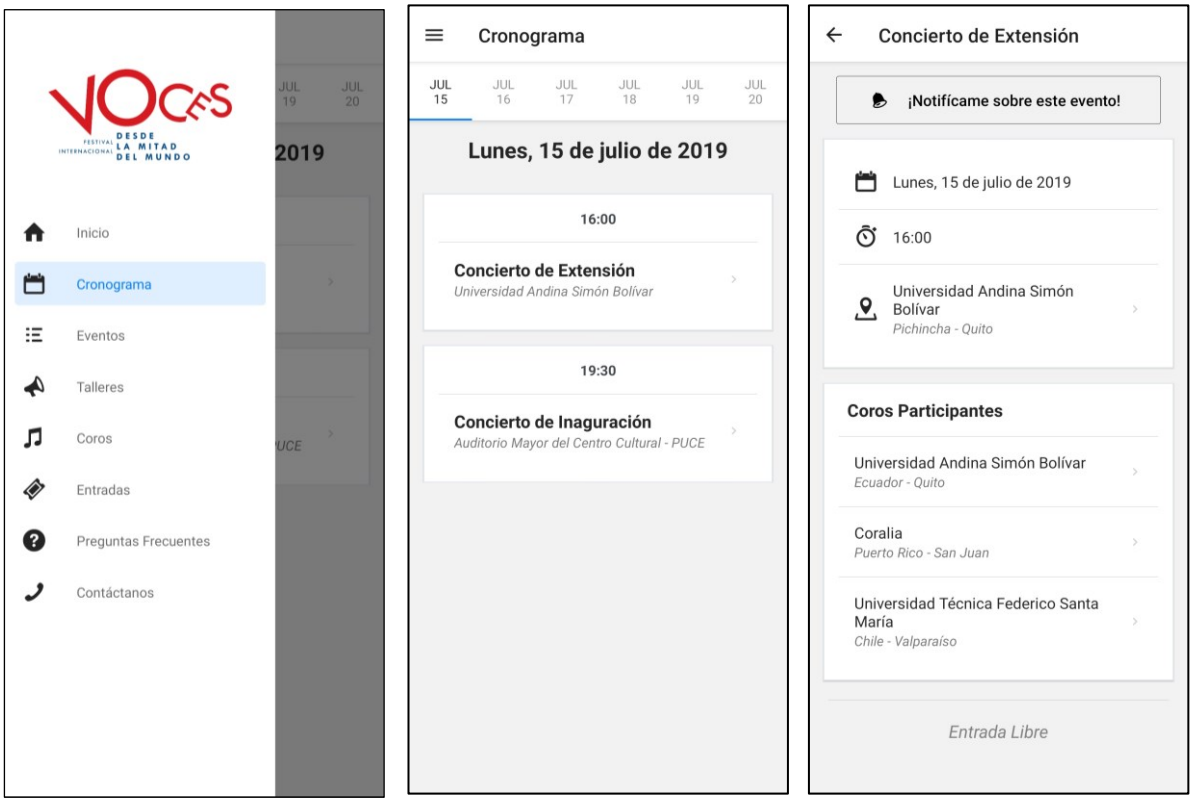


Ilustración 54 Silverio, V. (2021) Pasos para visualizar la información de los eventos. [Figura]

2. ¿Cómo ver la información de un taller?

2.1. Abrir el menú lateral.

2.2. Dirigirse a Talleres.

2.3. De entre la lista de talleres, presionar en cualquier imagen.

2.4. Se mostrará la información del lugar, horario (fechas y horas), ponente, y un botón que dirige al navegador web para más información del taller.

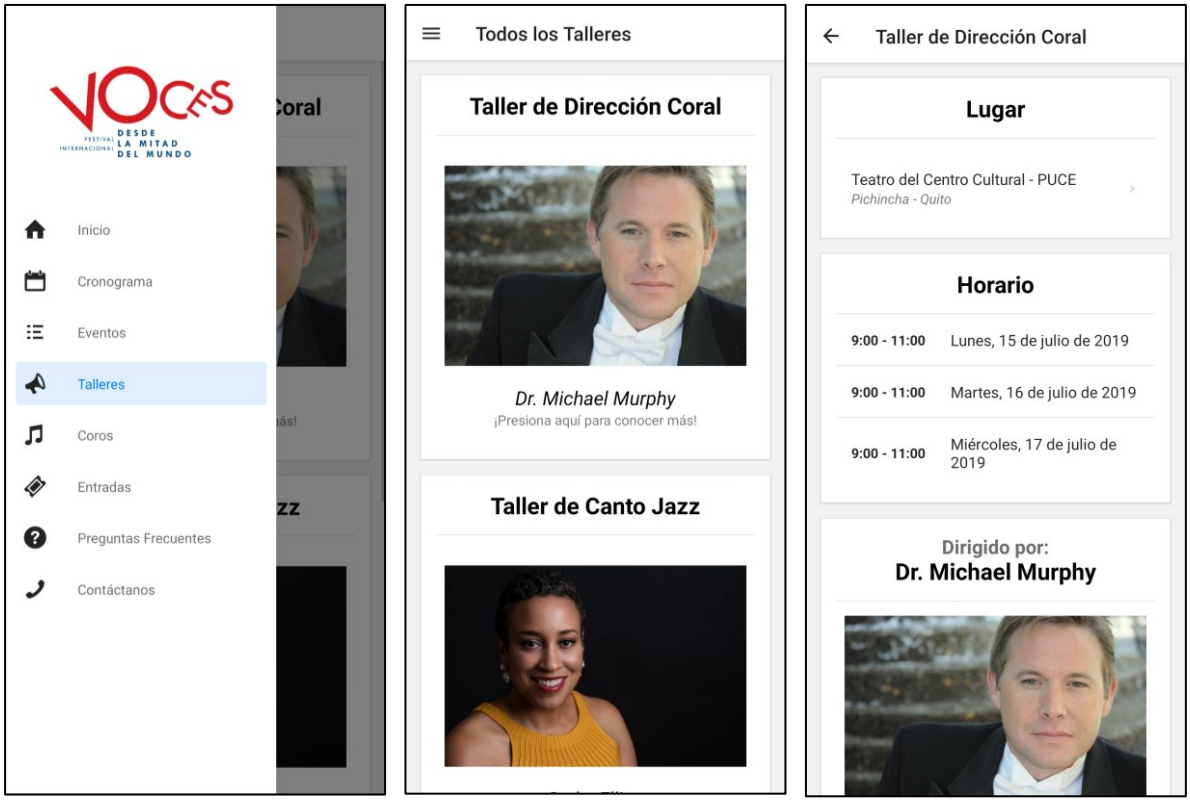


Ilustración 55 Silverio, V. (2021) Pasos para visualizar la información de los talleres. [Figura]

3. ¿Cómo ver la información de un coro participante?

3.1. Abrir el menú lateral.

3.2. Dirigirse a Coros.

3.3. De entre la lista de todos los coros participantes, seleccionar del cual se quiera visualizar su información.

3.4. Se presentará la información del nombre completo del coro, quién lo dirige, la ciudad y país de origen, el año de creación y una foto representativa.

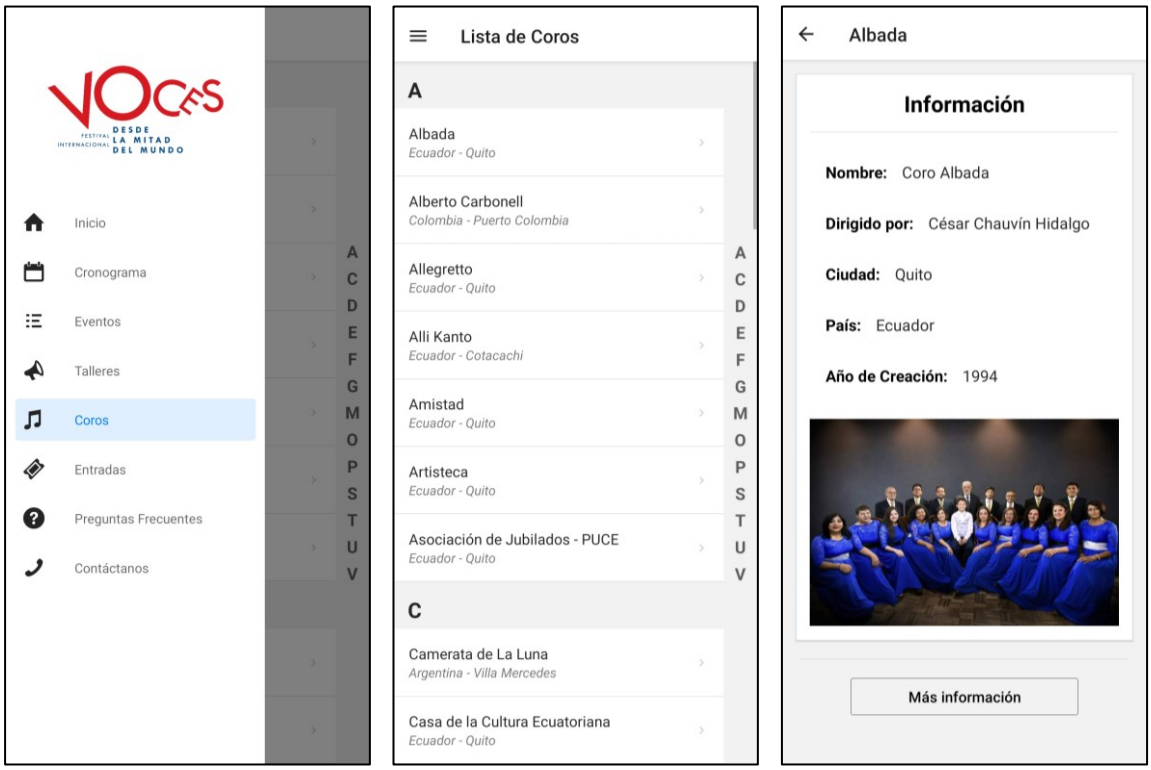


Ilustración 56 Silverio, V. (2021) Pasos para visualizar la información de los coros participantes. [Figura]

4. ¿Cómo ver la información de las entradas?

4.1. Abrir el menú lateral.

4.2. Dirigirse a entradas.

4.3. Se encontrará una lista con todos los tipos de entradas del festival, con la respectiva información de dónde adquirir la entrada.

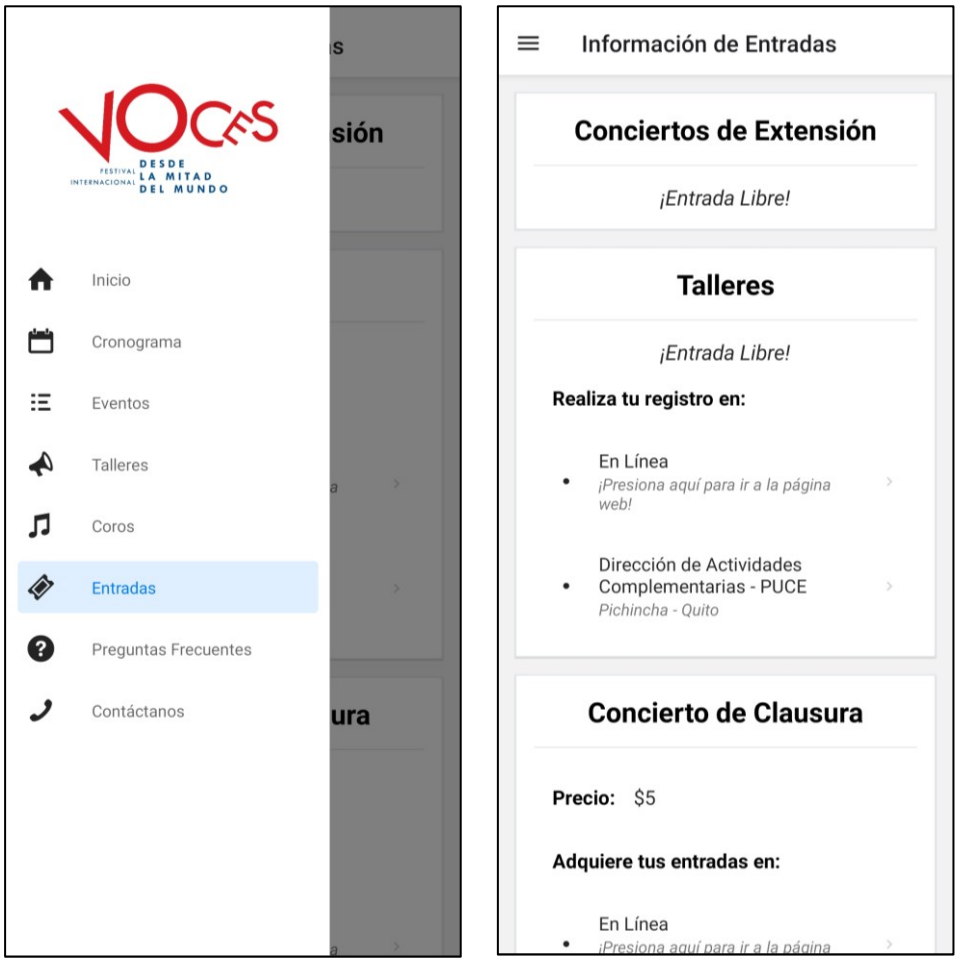


Ilustración 57 Silverio, V. (2021) Pasos para visualizar la información de las entradas. [Figura]

5. ¿Cómo ver información de preguntas frecuentes?

5.1. Abrir el menú lateral.

5.2. Dirigirse a preguntas frecuentes.

5.3. Se encontrará una lista de preguntas frecuentes con la respectiva respuesta que se muestra una vez que se presione en cualquier duda.

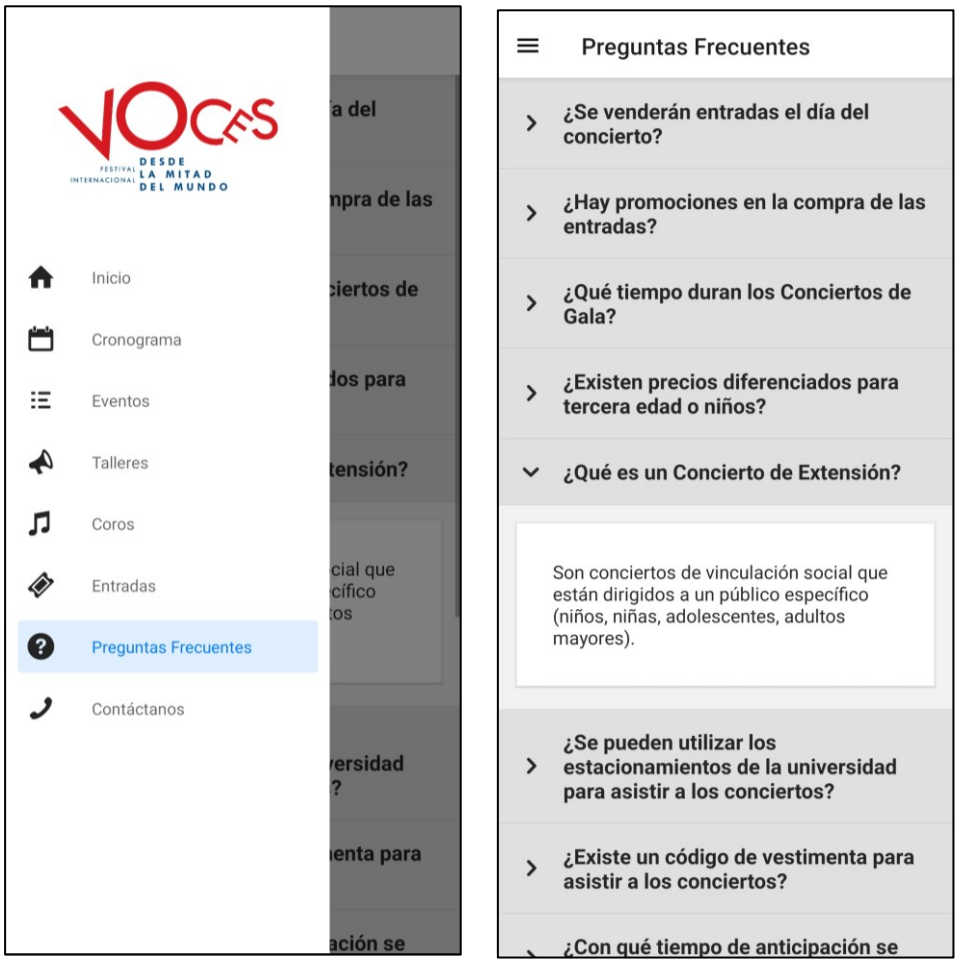


Ilustración 58 Silverio, V. (2021) Pasos para visualizar la información de preguntas frecuentes. [Figura]

6. **¿Cómo contactar con el equipo del festival?**
   6.1. Abrir el menú lateral.
   6.2. Dirigirse a la sección de contacto.
   6.3. Dentro se pueden encontrar todas las redes sociales del festival. También hay un formulario que el usuario puede llenar con el fin de enviar un mensaje directamente al correo electrónico del festival.

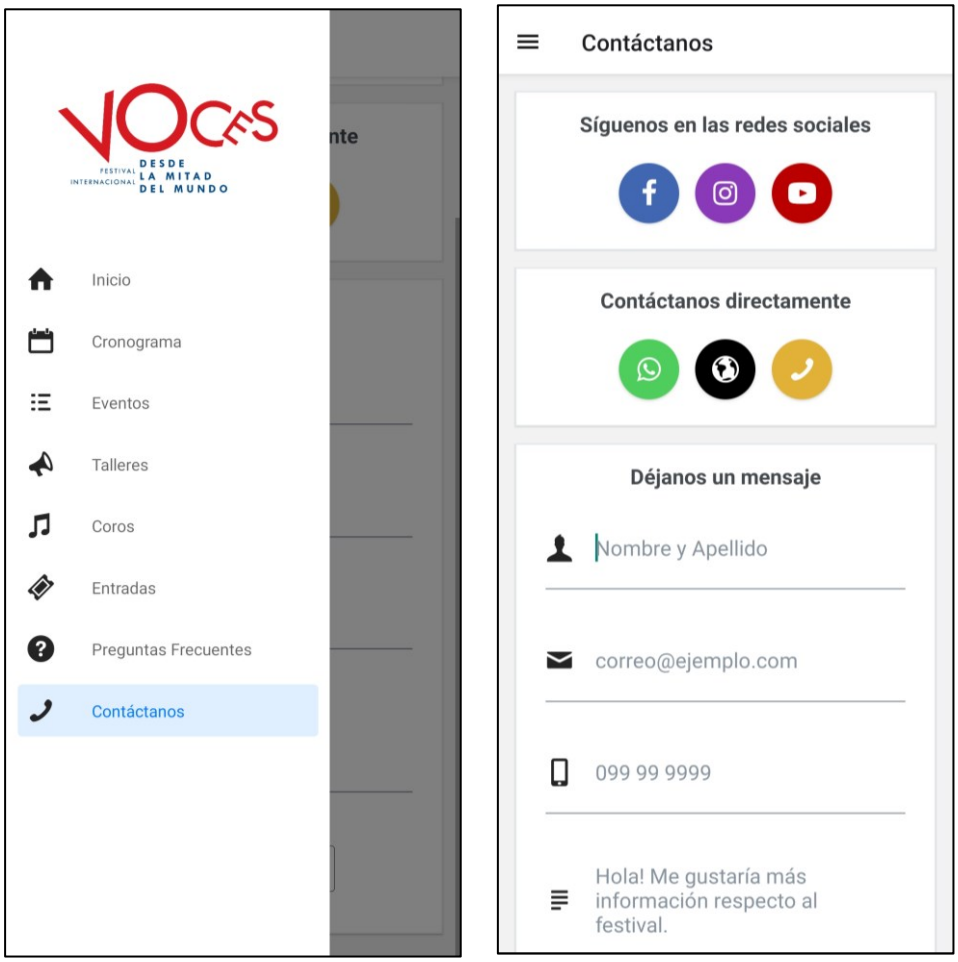


Ilustración 59 Silverio, V. (2021) Pasos para visualizar la información de contacto. [Figura]

Procedimientos de Instalación

1. Dirigirse a la tienda de aplicaciones Google Play Store.
2. Buscar por “Festival Voces”. O también se puede ingresar directamente a https://play.google.com/store/apps/details?id=com.festivalvoces.app&hl=es.
3. Verificar que el dispositivo móvil sea compatible con la app.
4. Presionar el botón de instalar.
5. La aplicación se descargará e instalará automáticamente.

5.1.3 Recapitulación

La fase de Producción es un paso que se realiza después de cada iteración. Al haberse elaborado código lo suficientemente estable como para haber pasado todas las pruebas de aceptación, sistema, carga e instalación, de manera que pueda considerarse concluida la iteración, el código es estable para generar un lanzamiento pequeño. Este último se refiere a entregas realizadas al cliente de manera frecuente que incluyen cada pequeño cambio realizado. Con cada lanzamiento, se producir valor al negocio para continuar agregando funcionalidades al sistema.

De igual forma, el producto de software al ser estable, se puede generar documentación guía a partir de este. Se recomienda que la documentación sea orientada al equipo técnico de la empresa (documentación del sistema y de operaciones) y para cualquier usuario que use la app móvil (documentación de soporte y de usuario).

5.2 Fase de Mantenimiento

La fase de Mantenimiento del Proyecto es la quinta etapa en la metodología de desarrollo XP. Su objetivo principal es analizar la posibilidad de incluir más funcionalidades al proyecto. De similar forma que la fase de producción, como indica Ambler (2002), la fase de mantenimiento parte desde la primera entrega que se ha hecho para el cliente: desde la segunda entrega hasta la enésima entrega, cualquier funcionalidad que se le agregue entra en la fase de mantenimiento. Esto se debe a la consideración de que la primera entrega debió contener las funcionalidades mínimas para un correcto funcionamiento de la aplicación para que pueda generar valor para el negocio.

El autor también explica que la etapa de mantenimiento abarca de nuevo el proceso de planificación, iteraciones y producción. Por tanto, se puede entender esta fase como un estado constante del proyecto. El proyecto dejará de estar en este estado cuando pase a la fase de muerte del proyecto, según se cumplan ciertas condiciones como la falta de presupuesto o falta de nuevas funcionalidades (Maida y Pacienzia, 2015).

El proyecto puede estar en producción y en mantenimiento al mismo tiempo, al ser estados independientes de la aplicación. Cierto conjunto de funcionalidades puede estar listo para aportar valor del negocio (estado de producción), mientras que se aumentan funcionalidades de una iteración anterior (estado de mantenimiento). Eventualmente, las versiones del software convergen a medida que los cambios implementados en el mantenimiento pasan a producción y llegan a la versión final en la muerte del proyecto.

Anwer et al. (2017) exponen que la diferencia entre la fase de mantenimiento y la solución de errores dentro de la fase de iteraciones se basa en que las funcionalidades anteriores que fueron definidas durante la planificación se deben conservar. Es decir, durante

la etapa de mantenimiento, no se reemplazan funcionalidades, solo aumentan. Los errores se discuten y se corrigen en las iteraciones de la tercera fase. Es por esta razón que el mantenimiento abarca las fases de planificación, iteraciones y producción.

5.2.1 Trabajo Futuro

Mientras el software no entre en la fase de Muerte del Proyecto, las nuevas funcionalidades que el cliente necesite en la aplicación entrarán en el Mantenimiento del Proyecto. Por esta razón, el trabajo futuro de la aplicación también está contemplada dentro de la Fase de Mantenimiento.

Funcionalidad	Historia de Usuario
Medio para subir publicaciones a un newsfeed	**Como** creador de contenido, **quiero** un medio de escritura de publicaciones en las redes sociales **para** informar al público en general.
Replicas automáticas	**Como** encargado de responder las preguntas, **quiero** un módulo que pueda generar réplicas automáticas **para** optimizar el proceso de contestación.
Suscripciones a transmisiones	**Como** organizador del festival, **quiero** un control sobre suscripciones a un sistema PPV **para** poder brindar transmisiones de los eventos.
Historial	**Como** organizador del festival, **quiero** un historial de todos los anteriores festivales **para** mostrar el progreso del proyecto.
Tema artístico	**Como** diseñador del festival, **quiero** un medio que se adapte al tema artístico del presente año **para** mantener un estilo homogéneo.
Galería Multimedia	**Como** organizador del festival, **quiero** una galería multimedia **para** mostrar de una forma visible la experiencia de los eventos.

Tabla 12 Silverio, V. (2021) Lista de nuevas funcionalidades para implementar en el futuro [Tabla].

La Tabla 12 muestra un resumen de las funcionalidades categorizadas como trabajo futuro al final de la sección 3.2.1.2. De las 22 historias de usuario contempladas originalmente para el proyecto, se repartieron 16 en tres iteraciones, resultando en seis historias pendientes. Juntamente con el cliente, se debía analizar el valor que aporte al negocio comparado con el presupuesto que se tenga.

Cabe resaltar que el presupuesto del proyecto no necesariamente se refiere a términos financieros, sino también puede tratarse sobre el tiempo que se posea para realizar entregas del software. Para el caso de la presente aplicación móvil, por el limitado tiempo para el desarrollo de la disertación de grado, se dejaron las seis iteraciones para ser analizadas en el trabajo futuro o fase de mantenimiento.

Sin embargo, se determinó que las historias de usuario no aportarían el suficiente valor al negocio como para aplazar un poco más la entrega del software. De este modo, el proyecto pasó a la última fase de la metodología de desarrollo de software: Muerte del Proyecto.

5.2.2 Recapitulación

La etapa de mantenimiento es un estado en el cual el proyecto se encuentra desde la primera entrega del software. Esta fase es la encargada de agregar nuevas funcionalidades siempre que aún exista presupuesto de parte del cliente. En el mantenimiento se contemplan tanto funcionalidades contempladas en la planificación, como aquellas que se clasificaron como trabajo futuro. El presente proyecto dejó algunas funcionalidades sin implementar por limitantes en cuanto al tiempo para el desarrollo de la disertación de grado, pasando directamente a la fase de muerte del proyecto cuando se culminó la implementación de todas las historias de usuario planificadas.

5.3 Fase de Muerte del Proyecto

La última fase de la metodología de desarrollo de software corresponde a la muerte del proyecto. Es la fase que cierra todo el proceso de construcción de la aplicación. Al igual que con la fase de mantenimiento, se puede entender a la etapa como un estado del proyecto. Es decir, el proyecto puede estar en un estado inicial (desde la fase de exploración, hasta la primera entrega en producción), en un estado de mantenimiento (desde el segundo lanzamiento hasta la enésima iteración), y finalmente, en un estado de inactividad (muerte del proyecto).

Anwer et al. (2017) y Maida y Pacienzia (2015) coinciden en que existen dos principales casos en los cuales un proyecto pueda pasar a la fase de muerte. En primer lugar, se trata de una culminación de la implementación de todas las historias de usuario posibles, inclusive aquellas referentes al trabajo futuro. En este caso, el cliente está satisfecho con el software y no requiere añadir más funcionalidades. En segundo lugar, el proyecto puede morir cuando ya no sea rentable para el negocio. En otras palabras, ya no aporta suficiente valor como para mantener la aplicación, o no es económicamente viable agregar más funcionalidades.

Adicionalmente, Maida y Pacienzia (2015) resaltan una tercera posibilidad en la que las funcionalidades de la aplicación en cuestión puedan ser absorbidas por un proyecto de software más grande. Por ejemplo, si existiera una aplicación con similares funcionalidades y que presente toda la actividad coral del Ecuador, incluyendo aquellos eventos del Festival Voces desde la Mitad del Mundo, el presente proyecto pasaría a ser parte de esta nueva aplicación, moviéndose a la fase de muerte.

5.3.1 Aplicación al proyecto

Como se indicó, el proyecto puede llegar a su muerte por la falta de funcionalidades que implementar, por no ser económicamente viable y/o por ser absorbido por otro proyecto más grande. Una vez explicadas las otras cinco etapas de la metodología de desarrollo de software, es pertinente mostrar cómo el presente proyecto ha llegado a su muerte.

La principal razón para la muerte de la presente aplicación es el aspecto económico. Tal y como lo define Sevilla Arias (2015), “la economía es una ciencia social que estudia la forma de administrar los recursos disponibles para satisfacer las necesidades humanas”. En relación con la presente disertación de grado, mientras que aún había funcionalidades para implementar, y no existe un proyecto más grande que pueda absorber a la aplicación, se determina que el recurso más limitado es el tiempo de entrega del software.

Cabe recalcar, entonces, que la única razón para llevar al proyecto hasta esta sexta fase de muerte, resulta ser la culminación del periodo para el desarrollo de la disertación de grado. Por otro lado, si se tratase de una necesidad del usuario, la aplicación podría continuar en el estado de mantenimiento hasta cumplir todas las funcionalidades.

5.3.2 Recapitulación

A la fase de muerte del proyecto se llega únicamente cuando ya no se puede continuar con este. Podría suceder porque la aplicación ha sido absorbida por un proyecto más grande, por haberse implementado todas las historias de usuario o por falta de recursos. Para la aplicación a la que se refiere el presente trabajo de titulación, la muerte del proyecto fue apresurada por escasez en el recurso del tiempo para presentar más funcionalidades que aquellas planificadas.

5.4 Conclusión

Así, al culminar todo el proceso de ingeniería de software, bajo el marco de referencia de XP, se puede concluir que las fases del proyecto pueden entenderse como estados. En un primer momento, existe el estado inicial, que incluye la fase de exploración, planificación, primera iteración y entrega; el estado de mantenimiento, donde se estudian nuevas

funcionalidades para añadir a la entrega anterior, mientras se continúan realizando los siguientes lanzamientos; y el estado de muerte del proyecto, donde se culminan todos los procesos por posibles diferentes circunstancias. Entendiendo el proceso como estados se puede ver que el proyecto puede estar en más de una etapa a la vez, haciéndolo mucho más adaptable al cambio.

De la misma manera, es preciso destacar que un correcto manejo de recursos, especialmente del tiempo de entregas, es muy importante en un proyecto de software. Este es el principal causante de que el proyecto haya llegado a la fase de muerte antes de que se implementaran todas las funcionalidades.

6 CONCLUSIONES Y RECOMENDACIONES

6.1 Conclusiones

- Es fundamental escoger la metodología de desarrollo de software como primera etapa del trabajo, puesto que dicta la totalidad del proceso de construcción de la aplicación y las características del proyecto, como la forma de extracción de requerimientos, la relación con el cliente y la documentación generada.
- Una metodología de desarrollo de software con facilidad de implementar cambios permitió al proyecto que no se genere un impacto grande por las cambiantes necesidades estudiadas con el cliente, las cuales moldearon todo el proceso de construcción de la aplicación.
- Al hacer uso de un marco de referencia orientado al cambio como XP, se logró sobrellevar las dificultades referentes a la implementación de las actividades principales (modelamiento, pruebas, programación e integración) dentro de las iteraciones, debido a la falta de madurez de las herramientas. Esto permitió que se puedan adaptar las tareas según sea más eficiente para el proyecto.
- El manejo óptimo de recursos en la construcción de la aplicación móvil permitió llevar al proyecto hasta la fase de muerte dentro del tiempo establecido y cumpliendo con las historias de usuario discutidas con el cliente, sin salirse del alcance definido.

6.2 Recomendaciones

- Se recomienda, para quienes quieren construir sistemas computacionales de manera particular, como los trabajadores autónomos (freelance) que, dado que no existen metodologías enfocadas al trabajo individual, se adapten las actividades grupales para un desarrollo individual, como se ha realizado en el presente proyecto, de manera que se alineen correctamente a los objetivos del proyecto.
- Se recomienda, para los equipos de desarrollo que trabajan con XP, que enfoquen sus esfuerzos principales en la Fase de Exploración la cual, si está correctamente construida, facilita el proceso de planificación, desarrollo y pruebas.
- Se recomienda, para quienes quieran manejar nuevas herramientas de software, que se analice si es provechoso cambiar los procesos actuales, tal como un estudio de complejidad-beneficio, para que la construcción del proyecto sea eficaz y eficiente.
- Se recomienda, para equipos trabajando bajo la referencia de XP, que generen documentación específica y orientada tanto al equipo técnico del cliente como para el usuario general, después de la fase de mantenimiento dado a que el software es lo suficientemente maduro para escribir una guía sobre este.

7 GLOSARIO

A

API: Es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones., 36, 74, 88, 89, 107, 110

Arquitectura de Software: Es un grupo de abstracciones y patrones que brindan un esquema de referencia útil para el desarrollo de software dentro de un sistema informático., 7, 8, 9, 110

Artefactos: Productos tangibles del proyecto que son producidos, modificados y usados., 7, 8, 9

Asíncrona: Se refiere a datos que no están sincronizados cuando se envían o reciben, lo que significa que la sincronización no ocurre a intervalos regulares o predeterminados. La comunicación asincrónica implica datos que se pueden transmitir de forma intermitente en lugar de en un flujo constante., 70

B

Biblioteca: Es un conjunto de subprogramas utilizados para desarrollar software., 35

C

Commit: Fragmentos lógicos de cambios que se guardan en secuencia, formando un historial que luego puede revisar., 64

D

Dashboard: Es una herramienta de administración de información que rastrea, analiza y presenta indicadores de rendimiento clave, métricas y puntos de datos clave para monitorear la salud de un negocio, departamento o proceso específico., 35

E

End-to-end: Traducido al español como extremo a extremo, se define como un proceso que incluye todo lo que es necesario para todas las partes de una red computacional para ser conectada y que funcione conjuntamente., 60

Expresión Regular: Una expresión regular es una cadena de caracteres que es utilizada para describir o encontrar patrones dentro de otros strings, en base al uso de delimitadores y ciertas reglas de sintaxis., 72

H

Historias de Usuario: Descripciones, a grandes rasgos, de las funcionalidades y requerimientos que el aplicativo debe resolver., 18, 21, 28

Hook: son funciones integradas que permiten a los desarrolladores de React usar métodos de estado y ciclo de vida dentro de los componentes funcionales, también trabajan junto con el código existente, por lo que se pueden adoptar fácilmente en el código., 62, 66, 70

I

Interfaz: Dispositivo capaz de transformar las señales generadas por un aparato en señales comprensibles por otro., 34, 36, 109

L

Liberación del Software: Es la distribución de la versión final de una aplicación de software., 15

Log: Es un archivo de texto en el que constan cronológicamente los acontecimientos que han ido afectando a un sistema informático (programa, aplicación, servidor, etc.), así como el conjunto de cambios que estos han generado., 38, 107

M

Metáfora de Negocio: Marco de trabajo con los objetos básicos de la arquitectura y sus interfaces., 18

Mocking: Es un proceso usado en las pruebas unitarias cuando la unidad que está siendo probada tiene dependencias externas. El propósito del mocking es aislar y concentrarse en el código que se está probando y no en el comportamiento o estado de las dependencias externas. En el mocking, las dependencias son reemplazadas por objetos controlados cercanamente que simulan el comportamiento de objetos reales., 59

N

Notificaciones Push: Las notificaciones push son mensajes que se pueden enviar directamente al dispositivo móvil de un usuario. Pueden aparecer en una pantalla de bloqueo o en la sección superior de un dispositivo móvil. Un editor de aplicaciones solo puede enviar una notificación de inserción si el usuario tiene su aplicación instalada., 75, 76

P

Pago Por Visión: Un servicio de televisión en el cual los espectadores requieren pagar una tarifa con el fin de ver un programa específico., 32

Pila: Una colección de elementos en la que solo se puede eliminar el elemento agregado más recientemente. El último elemento agregado está en la parte superior, 68

Product Backlog: Lista de objetivos/requisitos priorizada del producto., 15, 16, 17, 18

Pruebas Unitarias: Son una manera de poder probar cada unidad de código. Es decir, un grupo de líneas de código que puede ser lógicamente aislado de un sistema; funciones, propiedades, subrutinas., 58

Pull Request: Permiten informar a otros usuarios sobre los cambios que se ha enviado a una rama en un repositorio en GitHub. Una vez que se abre una Pull Request, se puede discutir y revisar los cambios potenciales con los colaboradores y agregar confirmaciones de seguimiento antes de que sus cambios se fusionen en la rama base., 64

Q

QA: En español, Aseguramiento de la Calidad, es definido como un procedimiento para asegurar la calidad de productos de software o servicios proporcionados a los clientes de una organización., 59, 108, 110

R

Rama: Es una forma de mantener separadas las líneas de desarrollo., 64

S

Spike arquitectónico: Es un conjunto de actividades realizadas por el equipo de desarrollo como pruebas y familiarización con las tecnologías, reconocimiento de posibles arquitecturas y realización de prototipos., 18, 20

Suites de Pruebas: Es una colección de casos de prueba que se agrupan con el propósito de ejecutar pruebas., 59

T

Token: Es una unidad de valor que una organización crea para gobernar su modelo de negocio y dar más poder a sus usuarios para interactuar con sus productos, al tiempo que facilita la distribución y reparto de beneficios entre todos sus accionistas, 38

U

UML: (Lenguaje Unificado de Modelado) Herramienta usada para forjar un lenguaje de modelado visual común y semántica y sintácticamente rico para la arquitectura, el diseño y la implementación de sistemas de

8 FUENTES BIBLIOGRÁFICAS

Abrahamsson, P., Hanhineva, A., Hulkko, H., Ihme, T., Jäälinoja, J., Korkala, M., . . . Salo, O. (2004). *Mobile-D: An Agile Approach for Mobile Application Development.* Vancouver: Technical Research Centre of Finland. Recuperado el 10 de marzo de 2021, de https://www.researchgate.net/publication/221322054_Mobile-D_An_Agile_Approach_for_Mobile_Application_Development

Adjust. (4 de mayo de 2017). *Push Notification | Meaning*. Obtenido de Adjust: https://www.adjust.com/glossary/push-notification/

Agile Alliance. (13 de junio de 2017). *Given – When – Then*. Obtenido de Agile Alliance: https://www.agilealliance.org/glossary/gwt/

Agile Alliance. (22 de julio de 2019). *User Stories*. Obtenido de Agile Alliance: https://www.agilealliance.org/glossary/user-stories/

Akintayo, S. (10 de abril de 2020). *Getting Started With The React Hooks API*. Obtenido de Smashing Magazine: https://www.smashingmagazine.com/2020/04/react-hooks-api-guide/

Amazon Web Services. (7 de junio de 2016). *Creación de aplicaciones con arquitecturas sin servidor*. Obtenido de Amazon Web Services: https://aws.amazon.com/es/lambda/serverless-architectures-learn-more/

Ambler, S. (31 de enero de 2002). *AM Throughout the XP Lifecycle*. Recuperado el 27 de marzo de 2021, de Agile Modeling: http://agilemodeling.com/essays/agileModelingXPLifecycle.htm

Ankama. (27 de abril de 2015). *¿Qué es un log?* Obtenido de Support Ankama: Support

Anwer, F., Aftab, S., Shah, S. M., & Waheed, U. (2017). *Comparative analysis of two popular agile process models: extreme programming and scrum.* Lahore: International Journal of Computer Science and Telecommunications.

Appypie. (11 de marzo de 2020). *How much does a Google & Apple Developer Account Cost?* Obtenido de Appypie: https://www.appypie.com/faqs/how-much-does-a-googleapple-developer-account-cost

Beal, V. (24 de mayo de 2021). *Asynchronous.* Obtenido de WeboPedia: https://www.webopedia.com/definitions/asynchronous/

Black, P. (1 de octubre de 2019). *Stack.* Obtenido de NIST: https://xlinux.nist.gov/dads/HTML/stack.html

Bowes, J. (28 de agosto de 2014). *Agile Concepts: User Stories*. Obtenido de manifesto: https://manifesto.co.uk/agile-concepts-user-stories/

Cambridge Dictionary. (17 de diciembre de 2020). *end-to-end*. Obtenido de Cambridge Dictionary: https://dictionary.cambridge.org/es/diccionario/ingles/end-to-end

Campillo Alhama, C., & Herrero Ruiz, L. (2015). *Experiencia de marca en los eventos para generar imagen y reputación corporativa.* Maracaibo: Universidad del Zulia.

Carrasco Usano, S. (2015). *Análisis de la aplicación de la tecnología móvil en las empresas.* Valencia: Universidad Politécnica de Valencia.

Communications. (11 de marzo de 2021). *Qué es un 'token' y para qué sirve*. Obtenido de BBVA: https://www.bbva.com/es/que-es-un-token-y-para-que-sirve/

Corral, L., Sillitti, A., & Succi, G. (2015). *Software assurance practices for mobile applications.* Bolzano: Free University of Bozen-Bolzano.

Expo. (29 de enero de 2020). *Introduction to Expo*. Obtenido de Expo Docs: https://docs.expo.io

Gaba, R., & Ramachandran, A. (2017). *React Made Native Easy.*

García Rodriguez, M. J. (Julio de 2015). *Estudio comparativo entre las metodologías ágiles y las metodologías tradicionales para la gestión de proyectos software.* Recuperado el 4 de Marzo de 2021, de Universidad de Oviedo: https://digibuo.uniovi.es/dspace/bitstream/handle/10651/32457/TFMMlJGarciaRodriguezRUO.pdf?sequence=6&isAllowed=y

GitHub. (1 de junio de 2012). *About pull requests*. Obtenido de GitHub Docs: https://docs.github.com/en/github/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-pull-requests

Google Developers. (3 de octubre de 2017). *Cloud Firestore*. Obtenido de Firebase: https://firebase.google.com/docs/firestore

Guru99. (15 de mayo de 2021). *Load Testing Tutorial: What is? How to? (with Examples)*. Obtenido de Guru99: https://www.guru99.com/load-testing-tutorial.html

Guru99. (23 de marzo de 2021). *What is Quality Assurance(QA)? Process, Methods, Examples*. Obtenido de Guru99: https://www.guru99.com/all-about-quality-assurance.html

Guru99. (9 de mayo de 2021). *What is System Testing? Types & Definition with Example*. Obtenido de Guru99: https://www.guru99.com/system-testing.html

Gutiérrez, J. J. (12 de mayo de 2014). *¿Qué es un framework web?* Obtenido de Isi.us: http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf

Herrera, S., & Fennema, M. (2011). *Tecnologías Móviles Aplicadas a la Educación Superior.* Santiago del Estero: Universidad Nacional de Santiago del Estero.

IBM. (11 de junio de 2020). *Test cases and test suites*. Obtenido de IBM: https://www.ibm.com/docs/nl/elm/7.0.3?topic=scripts-test-cases-test-suites

Jabbar, G. (10 de marzo de 2021). *React Native Coding Standards and Best Practices*. Obtenido de Medium: https://gilshaan.medium.com/react-native-coding-standards-and-best-practices-5b4b5c9f4076

Klipfolio. (13 de julio de 2016). *What is a data dashboard?* Obtenido de Klipfolio: https://www.klipfolio.com/resources/articles/what-is-data-dashboard

Lexico. (19 de julio de 2019). *Interfaz*. Obtenido de Lexico: https://www.lexico.com/es/definicion/interfaz

Lexico. (23 de junio de 2019). *Metáfora*. Obtenido de Lexico: https://www.lexico.com/es/definicion/metafora

Lexico. (22 de diciembre de 2019). *Pay-per-view*. Obtenido de Lexico: https://www.lexico.com/en/definition/pay-per-view

Lou, T. (2016). *A comparison of Android Native App Architecture MVC, MVP and MVVM.* Eindhoven: Eindhoven University of Technology. Recuperado el 2 de abril de 2021

Lucidchart. (10 de mayo de 2017). *Qué es el lenguaje unificado de modelado (UML)*. Obtenido de Lucidchart: https://www.lucidchart.com/pages/es/que-es-el-lenguaje-unificado-de-modelado-uml

Luna, D. R. (2016). *Usabilidad en Sistemas de Información: comparación del diseño centrado en el usuario vs. técnicas tradicionales.* Buenos Aires: Facultad Regional Buenos Aires.

Maida, E. G., & Pacienzia, J. (diciembre de 2015). *Metodologías de desarrollo de software.* Recuperado el 3 de marzo de 2021, de Universidad Católica Argentina: https://repositorio.uca.edu.ar/bitstream/123456789/522/1/metodologias-desarrollo-software.pdf

Malek, P. (28 de agosto de 2019). *Sending Emails with React Native*. Obtenido de Mailtrap: https://blog.mailtrap.io/react-native-send-email/

Ministerio de Trabajo. (6 de abril de 2021). *Salarios Mínimos Sectoriales.* Obtenido de Ecuador Legal: https://drive.google.com/file/d/1noQiABtC9fK72CF8n77GvuuNbZt0sh2h/view

Mistry, A. (24 de octubre de 2017). *User Story In Agile Scrum*. Recuperado el 24 de marzo de 2021, de C# Corner: https://www.c-sharpcorner.com/article/what-is-user-story-in-agile-scrum/

Mogollón Afanador, J., & Esteban Villamizar, L. A. (2010). *El desarrollo Individual de Proyectos de Software: Una Realidad sin Método.* Santander: Universidad de Pamplona.

Molina Montero, B., Vite Cevallos, H., & Dávila Cuesta, J. (Abril de 2018). *Metodologías ágiles frente a las tradicionales en el proceso de desarrollo de software.* Recuperado el 4 de Marzo de 2021, de Espirales: http://revistaespirales.com/index.php/es/article/view/269

Muñoz Muñoz, C. A. (30 de octubre de 2020). *Aplicación de la metodología Mobile-D en el desarrollo de una app móvil para gestionar citas médicas en el Centro JEL Riobamba.*

Recuperado el 10 de marzo de 2021, de Uiversidad Nacional de Chimborazo: http://dspace.unach.edu.ec/handle/51000/7073

Ortega, M. A., & Camacho, E. A. (2019). *Uso de los modelos tradicionales y las metodologías ágiles aplicadas en la industria de software colombiano.* Cali: Universidad Santiago de Cali.

Professional QA. (13 de marzo de 2020). *Installation Testing*. Obtenido de Professional QA: https://www.professionalqa.com/installation-testing

Quintero Vélez, D. P. (2016). *Modelo de gestión del museo virtual para el laboratorio de muestras geológicas de la facultad de ciencias naturales.* Guayaquil: Universidad de Guayaquil.

React. (25 de junio de 2019). *Getting Started*. Recuperado el 2 de abril de 2021, de React Docs: https://reactjs.org/docs/getting-started.html

React Native. (12 de marzo de 2021). *Core Components and Native Components*. Recuperado el 2 de abril de 2021, de React Native Docs: https://reactnative.dev/docs/intro-react-native-components

React Native. (12 de marzo de 2021). *Testing*. Obtenido de React Native Docs: https://reactnative.dev/docs/testing-overview

Real Academia Española. (7 de octubre de 2020). *Metáfora*. Obtenido de Real Academia Española: https://dle.rae.es/metáfora

Red Hat. (4 de marzo de 2019). *¿Qué es una API?* Obtenido de Red Hat: https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces

Roden, M. (30 de junio de 2002). *User Story*. Recuperado el 19 de marzo de 2021, de WikiWikiWeb: http://wiki.c2.com/?UserStory

Rosselló Villán, V. (15 de marzo de 2019). *Las metodologías ágiles más utilizadas y sus ventajas dentro de la empresa*. Obtenido de IEBS: https://www.iebschool.com/blog/que-son-metodologias-agiles-agile-scrum/

Sangama Oñate, A. F. (2020). *Metodologías ágiles Scrum, XP, SLeSS, Scrumban, HME, Mobile-D y MASAN empleadas en la industria de dispositivos móviles: Un contraste en favor de la industria del desarrollo móvil.* Tarapoto: Universidad Peruana Unión.

Sarasty España, H. F. (2015). *Documentación y Análisis de los Principales Frameworks de Arquitectura de Software en Aplicaciones Empresariales.* Buenos Aires: Universidad Nacional de La Plata.

Schwaber, K., & Sutherland, J. (noviembre de 2020). *The Scrum Guide.* Recuperado el 8 de marzo de 2021, de Scrum.org: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf#zoom=100

SCRUMstudy. (4 de octubre de 2013). *Scrum Phases and Processes.* Obtenido de SCRUMstudy: https://www.scrumstudy.com/whyscrum/scrum-phases-and-processes

Sensagent. (19 de diciembre de 2010). *Biblioteca (informática)*. Obtenido de Sensagent: http://diccionario.sensagent.com/Biblioteca%20(informática)/es-es/

Sevilla Arias, A. (8 de octubre de 2015). *Economía*. Recuperado el 29 de mayo de 2021, de Economipedia: https://economipedia.com/definiciones/economia.html

SmartBear. (23 de abril de 2015). *What Is Unit Testing?* Obtenido de SmartBear: https://smartbear.com/learn/automated-testing/what-is-unit-testing/

Software Guru. (1 de julio de 2008). *Expresiones Regulares. Conócelas y Piérdeles el miedo.* Obtenido de Software Guru: https://sg.com.mx/content/view/545

TechTarget Contributor. (Marzo de 2008). *release*. Obtenido de TechTarget: https://searchsoftwarequality.techtarget.com/definition/release

TechTarget Contributor. (2 de abril de 2018). *Release*. Obtenido de TechTarget: https://searchsoftwarequality.techtarget.com/definition/release

Telerik. (4 de febrero de 2013). *Unit Testing for Software Craftsmanship*. Obtenido de Telerik: https://www.telerik.com/products/mocking/unit-testing.aspx

The Pilcrow. (12 de agosto de 2015). *Explaining the basic concepts of Git and how to use GitHub*. Obtenido de The Pilcrow: https://thepilcrow.net/explaining-basic-concepts-git-and-github/

Tumpibamba Borja, E. E. (2016). *Desarrollo de una aplicación móvil que permite a los docentes y estudiantes de la Universidad Central del Ecuador acceder a las bases de datos científicas.* Quito: Universidad Central del Ecuador.

Wells, D. (31 de enero de 2001). *User Stories*. Obtenido de Extreme Programming: http://www.extremeprogramming.org/rules/userstories.html

Wells, D. (31 de marzo de 2002). *Project Velocity*. Obtenido de Extreme Programming: http://www.extremeprogramming.org/rules/velocity.html

Wells, D. (14 de marzo de 2006). *Make frequent small releases*. Obtenido de Extreme Programming : http://www.extremeprogramming.org/rules/releaseoften.html

Wells, D. (1 de marzo de 2006). *The Rules of Extreme Programming*. Obtenido de Extreme Programming: http://www.extremeprogramming.org/rules.html

Wells, D. (19 de septiembre de 2009). *Sequential Integration*. Recuperado el 1 de mayo de 2021, de Extreme Programming: http://www.extremeprogramming.org/rules/sequential.html

Wells, D. (26 de agosto de 2009). *Your host: Don Wells.* Recuperado el 9 de marzo de 2021, de Extreme Programming: http://www.extremeprogramming.org/donwells.html

Wells, D. (8 de octubre de 2013). *Extreme Programming: A gentle introduction.* Obtenido de Extreme Programming: http://www.extremeprogramming.org

Zumba Gamboa, J. P., & León Arreaga, C. A. (2018). *Evolución de las Metodologías y Modelos utilizados en el Desarrollo de Software.* Guayaquil: Universidad de Guayaquil.

9 ANEXOS

ÍNDICE DE ANEXOS

9.1 Anexo 1

Anexo 1 Silverio, V. (2021) Acta de reunión con Juan Carlos Velasco – 001 [Anexo].

ACTA DE REUNIÓN – No. 1
Lugar y fecha: Quito, 18 de marzo del 2021. (Reunión de Zoom)
Participantes: Juan Carlos Velasco, Víctor José Silverio.
Tema de la reunión: Determinación de las historias de usuario
Desarrollo de la reunión: • Explicación del propósito general de la aplicación móvil. • Presentación de antecedentes del proyecto cultural. • Conversación respecto a las necesidades del usuario referente a la aplicación. • Aclaración respecto a qué información es destacable de presentar sobre los festivales, participantes, eventos, talleres, entre otros.
Acuerdos finales: • El equipo de desarrollo debe escribir las historias de usuario a partir de lo hablado en la reunión. • El equipo de desarrollo debe organizar las historias de usuario para poder ser analizadas en la próxima reunión. • El equipo de desarrollo debe comunicarse con Andrea Rodríguez para recibir más información de los anteriores festivales.

Juan Carlos Velasco

Director del Proyecto Cultural "Voces desde la Mitad del Mundo"

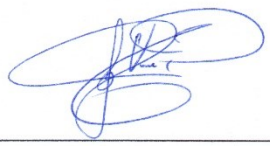
Víctor José Silverio

Estudiante de la Pontificia Universidad Católica del Ecuador

9.2 Anexo 2

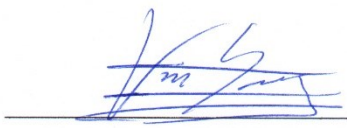
Anexo 2 Silverio, V. (2021) Acta de reunión con Juan Carlos Velasco – 002 [Anexo].

ACTA DE REUNIÓN – No. 2
Lugar y fecha: Quito, 25 de marzo del 2021. (Reunión de Zoom)
Participantes: Juan Carlos Velasco, Víctor José Silverio.
Tema de la reunión: Revisión y asignación de prioridades a las historias de usuario.
Desarrollo de la reunión: • Aclaración respecto a que se pueden aumentar historias de usuario. • Revisión de las 22 historias de usuario definidas. • Asignación de prioridades a las 22 historias de usuario.
Acuerdos finales: • El cliente va a analizar la posibilidad de aumentar nuevas historias de usuario. • El equipo de desarrollo debe organizar las historias de usuario según la prioridad hablada. • El equipo de desarrollo debe asignar estimaciones respecto al tiempo de implementación de las historias de usuario. • El Equipo de desarrollo debe armar una tentativa respecto a las iteraciones para la presentación en la siguiente reunión.



Juan Carlos Velasco

Director del Proyecto Cultural "Voces desde la Mitad del Mundo"



Víctor José Silverio

Estudiante de la Pontificia Universidad Católica del Ecuador

9.3 Anexo 3

Anexo 3 Silverio, V. (2021) Acta de reunión con Juan Carlos Velasco – 003 [Anexo].

ACTA DE REUNIÓN – No. 3
Lugar y fecha: Quito, 8 de abril del 2021. (Reunión de Zoom)
Participantes: Juan Carlos Velasco, Víctor José Silverio.
Tema de la reunión: Presentación de la organización de las iteraciones.
Desarrollo de la reunión: • Discusión respecto a la adición de nuevas historias de usuario. • Presentación del diagrama que resume la arquitectura que se usará en la aplicación móvil. • Conversación respecto a la Metáfora de Negocio. • Exposición de la organización de las entregas e iteraciones. • Explicación del trabajo a realizar por cada requerimiento.
Acuerdos finales: • El equipo de desarrollo debe avanzar en el desarrollo de la aplicación móvil, de manera que se pueda mostrar un avance a la mitad de la iteración 1, con el fin de resolver dudas y discutir cambios antes de la culminación de la iteración.

Juan Carlos Velasco

Director del Proyecto Cultural "Voces desde la Mitad del Mundo"

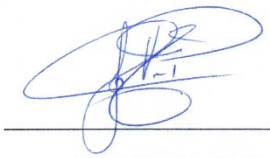
Víctor José Silverio

Estudiante de la Pontificia Universidad Católica del Ecuador

9.4 Anexo 4

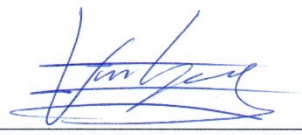
Anexo 4 Silverio, V. (2021) Acta de reunión con Juan Carlos Velasco – 004 [Anexo].

ACTA DE REUNIÓN – No. 4
Lugar y fecha: Quito, 6 de mayo del 2021. (Reunión de Zoom)
Participantes: Juan Carlos Velasco, Víctor José Silverio.
Tema de la reunión: Presentación del avance Iteración 1
Desarrollo de la reunión: • Revisión de las historias de usuario de la iteración 1 que se debieron implementar hasta la presente reunión. • Demostración de los avances en las funcionalidades de la aplicación. • Consulta por comentarios, cambios u opiniones de parte del cliente. • Presentación de las historias de usuario que se implementarán en la segunda iteración.
Acuerdos finales: • El equipo de desarrollo debe enviar el archivo APK de la aplicación para que el cliente la revise con más detenimiento. • El equipo de desarrollo debe avanzar en la implementación de las historias de usuario de la Iteración 2, para presentar los avances en un máximo de dos semanas.



Juan Carlos Velasco

Director del Proyecto Cultural "Voces desde la Mitad del Mundo"



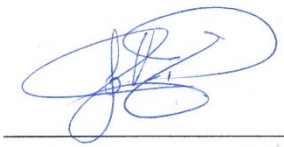
Víctor José Silverio

Estudiante de la Pontificia Universidad Católica del Ecuador

9.5 Anexo 5

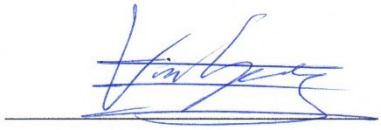
Anexo 5 Silverio, V. (2021) Acta de reunión con Juan Carlos Velasco – 005 [Anexo].

ACTA DE REUNIÓN – No. 5
Lugar y fecha: Quito, 20 de mayo del 2021. (Reunión de Zoom)
Participantes: Juan Carlos Velasco, Víctor José Silverio.
Tema de la reunión: Presentación del avance Iteración 2
Desarrollo de la reunión: • Revisión de las historias de usuario de la iteración 2 que se debieron implementar hasta la presente reunión. • Demostración de los avances en las funcionalidades de la aplicación. • Consulta por comentarios, cambios u opiniones de parte del cliente. • Presentación de las historias de usuario que se implementarán en la tercera iteración.
Acuerdos finales: • El equipo de desarrollo debe avanzar en la implementación de las historias de usuario de la Iteración 3, para presentar los avances en un máximo de una semana.



Juan Carlos Velasco

Director del Proyecto Cultural "Voces desde la Mitad del Mundo"



Víctor José Silverio

Estudiante de la Pontificia Universidad Católica del Ecuador

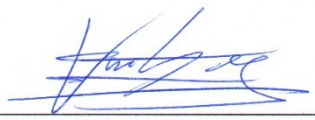
9.6 Anexo 6

Anexo 6 Silverio, V. (2021) Acta de reunión con Juan Carlos Velasco – 006 [Anexo].

ACTA DE REUNIÓN – No. 6
Lugar y fecha: Quito, 3 de junio del 2021. (Reunión de Zoom)
Participantes: Juan Carlos Velasco, Víctor José Silverio.
Tema de la reunión: Presentación del avance Iteración 3
Desarrollo de la reunión: • Revisión de las historias de usuario de la iteración 3 que se debieron implementar hasta la presente reunión, culminando la aplicación móvil. • Demostración de los avances en las funcionalidades de la aplicación. • Consulta por comentarios, cambios u opiniones de parte del cliente. • Cerrado del proceso de construcción de la aplicación móvil.
Acuerdos finales: No hay acuerdos finales.

Juan Carlos Velasco

Director del Proyecto Cultural "Voces desde la Mitad del Mundo"



Víctor José Silverio

Estudiante de la Pontificia Universidad Católica del Ecuador

9.7 Anexo 7

Anexo 7 Silverio, V. (2021) Acta de reunión con Andrea Rodríguez – 001 [Anexo].

ACTA DE REUNIÓN – No. 1
Lugar y fecha: Quito, 12 de abril del 2021. (Reunión de Zoom)
Participantes: Andrea Rodríguez, Víctor José Silverio.
Tema de la reunión: Recopilación de información importante para el desarrollo de la aplicación
Desarrollo de la reunión: • Resumen del objetivo del proyecto. • Visita a la página web oficial del proyecto cultural y análisis de la información pertinente contenida en esta. • Determinación de la información relevante que será expuesta en la aplicación móvil.
Acuerdos finales: • El equipo de desarrollo se comunicará con Freddy Coello, diseñador de la página web, para obtener acceso a los recursos multimedia. • El cliente facilitará los documentos que contienen información de los eventos, coros participantes, organizadores, auspiciantes, así como también el Cronograma General de la Edición 2019.

Andrea Rodríguez

Secretaria General del Proyecto Cultural Festival Internacional de Coros "Voces desde la Mitad del Mundo"

Víctor José Silverio

Estudiante de la Pontificia Universidad Católica del Ecuador